RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Bernd Stelzer

A Read-Out Driver Prototype-1 for the ATLAS Level-1 Calorimeter Trigger Pre-Processor

Diploma Thesis in Physics

HD-KIP-00-46
HD-ASIC-60-0700

KIRCHHOFF-INSTITUT FÜR PHYSIK

A Read-Out Driver Prototype-1 for the ATLAS Level-1 Calorimeter Trigger Pre-Processor

This diploma thesis has been carried out by Bernd Stelzer at the

Kirchhoff-Institut für Physik of the University of Heidelberg

under the supervision of

Prof. Dr. K. Meier

A Read-Out Driver Prototype-1
for the ATLAS Level-1 Calorimeter Trigger Pre-Processor

The subject of this thesis is a prototype-1 Read-Out Driver for the ATLAS Level-1 Calorimeter Trigger Pre-Processor system. This Read-Out Driver interconnects several Pre-Processor Modules via a custom ring-like bus, the PipelineBus. The main task of the Read-Out Driver is to collect read-out data from all connected modules and to send formatted data to the ATLAS data acquisition system.
Using a test system based on VME boards, the PipelineBus has been brought into operation and performance tests have been carried out.
For that purpose, a PipelineBus Master design has been written in the hardware description language Verilog and implemented in a FPGA. For tests, more FPGA designs have been developed, acting as data-spectators and data-sinks between data-processing nodes.
As a pre-stage of the final Read-Out Driver, a CMC daughtercard has been designed and manufactured. This card was successfully commissioned and first tests have been performed. The necessary logic has been developed and implemented on a Virtex FPGA placed on this daughtercard.

Ein Auslesetreiber Prototyp-1
für den ATLAS Level-1 Kalorimeter-Trigger Pre-Prozessor:

Das Thema dieser Arbeit ist ein Prototyp-1 Auslesetreiber für den ATLAS Level-1 Kalorimeter-Trigger Pre-Prozessor. Dieser Auslesetreiber verbindet mehrere Pre-Prozessor-Module über einen speziellen ringförmigen Bus: den PipelineBus. Die Hauptaufgabe des Auslesetreibers ist Auslesedaten aller verbundenen Module einzusammeln und formatierte Daten an die ATLAS Datenerfassung zu verschicken.
Mit einem auf VME-Karten basierenden Testsystem wurde der PipelineBus in Betrieb genommen und Performance-Messungen durchgeführt.
Zu diesem Zweck wurde ein PipelineBus Master Design in der Hardwarebeschreibungssprache Verilog geschrieben und in einem FPGA implementiert. Für Tests wurden weitere FPGA-Designs entwickelt, die als Datenmonitore und Datensenken zwischen Datenverarbeitungsknoten wirken.
Als Vorstufe für einen endgültigen Auslesetreiber wurde eine CMC Tochterkarte entwickelt und hergestellt. Diese Karte konnte erfolgreich in Betrieb genommen werden und erste Tests wurden durchgeführt. Die dafür notwendige Logik wurde entwickelt und in einem Virtex FPGA auf der Tochterkarte implementiert.

Contents

Introduction

Since earliest times people have been trying to answer fundamental question such as: 'what is the world made of and what holds it together?' 'What is the origin of the universe and therefore our existence?' Already as long as 2500 years ago, people have come to realize that the world might be made from a few fundamental elements of nature; elements that are simple and structureless, not made of anything smaller, elements which we define today as fundamental particles.

Empedocles of Acragas
(495-435 B.C.)

Empedocles of Acragas (495-435 B.C.) a Greek philosopher, postulated that earth, air, fire, and water are the basic substances of the world and that the quality of matter depends exclusively on the ratio of these elements. This led other Greek philosophers to ask if matter is infinite, or if it would be a size in quantity that is truly defined. Democritus of Abdera (460-380 B.C.) introduced the concept of 'atomos' (the Greek word atomos means the indivisible). He came up with the idea that if you start cutting matter, splitting stops when it reaches indivisible and indestructible particles.

Only very few new developments occurred in the centuries following the Greek era; the medieval period was mainly dominated and ruled by religion and superstition. However, as the intense intellectual force of the renaissance entered the field of physics, brave scientists like Galileo Galilei began to reject the domination of religious dogmas and started to extend the ancient Greek ideas of nature. However, Galileo's approach was different. As the ancient Greek theories were mainly based on philosophical arguments, Galileo started to base his theories on experimental evidence. He believed that a theory is only valid for as long as it conforms with experimental results. When a theory fails to do so, it has to be discarded or reformulated.

With this concept, physical science started its triumphal procession and knowledge about physics started to grow exponentially.

In 1896, Ernest Rutherford was the first to recognize that much could be learned from the structure of matter and its interaction, by doing scattering experiments. Since his pioneering work, physicists have learned how to lift the secrets of elementary particle physics by studying processes in the microcosm. For this reason huge particle accelerators have been built with high center-of-mass energies, to study matter at the tiniest scales[1].

This method was extremely successful and the great achievements of particle physics have been summarized in the famous Standard Model of particle physics (Chapter 1.1). However, there is one missing particle, predicted by theory, which has not yet been observed in an experiment. The Higgs boson is the last undiscovered particle of the Standard Model. If it turned out that the Higgs does not exist, this would be the strongest indication that the Standard Model is not a sufficient theory and there might be new physics around the corner which could change our understanding of the world completely.

To minimize the costs of new experiments and to boost developments in a joint effort most effectively, particle physicists have organized world wide collaboration of scientists with the same scientific goal.

In a few years from now, the Large Hadron Collider (LHC), will start operation at CERN near Geneva. With 153 participating institutes in 34 countries around the world, it will be the biggest joint venture in the history of physics and will allow the study of physics processes at energy scales which have never been created in the laboratory before. The LHC will therefore have the potential and the infrastructure to open the door to new physics and give new insights into the nature of the universe. The ATLAS experiment is one of the general purpose detectors at the LHC, designed to exploit the full discovery potential of the LHC and to be sensitive to all possible Higgs masses predicted by theory (Chapter 1.3).
A project, very worthwhile to contribute to.

This thesis focuses on the read-out of the ATLAS calorimeter trigger-data. Since the data-rates in high resolution particle detectors have grown to a level which can't possibly be stored on mass media any more ($\approx$ 50 Tbyte/s), the need for selective trigger logic arises (Chapter 2). By the use of dedicated trigger processors, it is possible to read-out among a vast number of background processes, the tiny fraction (10^{-7}) of which are due to interesting, probably new physics.

All trigger data coming from the ATLAS calorimeters have to pass the Level-1 trigger Pre-Processor system before it can be processed by the trigger. The Pre-Processor system performs digitization of analog input data, bunch-crossing identification and energy-calibration, and then sends the data to the succeeding Level-1 trigger processors (Chapter 3).

The Pre-Processor system provides facilities to read out raw trigger data on which the trigger decision was based. This allows the overall trigger performance to be checked, and allows the verification that the trigger tower data and the separated path of full-granularity read-out of the calorimeters are consistent.

A Read-Out Driver, interconnects several Pre-Processor modules via a custom bus (PipelineBus). Collection of read-out data on the board-level is performed by a Read-Out Merger FPGA. In a second step, read-out data is collected by the Read-Out Driver from several modules via the PipelineBus and sent further to the ATLAS data acquisition system (DAQ) (Chapter 5 and 6).

[1] The smaller the scale Δx one wants to explore in a particle accelerator, the higher the required energy E of the probing particles. A relation which can be derived from the Heisenberg uncertainty relation, and can be expressed as: $E \approx \frac{200\ MeV fm}{\Delta x}$

Chapter 1

The LHC and the ATLAS Experiment

1.1 Physical Context

1.1.1 The Standard Model

The Standard Model is a theoretical framework that explains the fundamental particles and the interaction they can undergo. It is basically a summary of the great experimental discoveries and theoretical developments made in the past 50 years in particle physics. It is able to make precise quantitative predictions which have been experimentally tested [1].

Fundamental Particles

According to the Standard Model all matter consists of two fundamental types of particles. The spin-$\frac{1}{2}$ fermions, represented by the quarks and leptons, and the spin-1 gauge bosons, which mediate the interactions between fermions. The up and down quark and the electron are all that are needed to build up observable matter such as human beings, mountains, airplanes etc. For some unknown reason, however there are three generations of quarks and leptons, of which two are just heavier copies of the lightest generation (see Table 1.1). Each generation consists of two quarks, a lepton and a corresponding lepton-neutrino. All

	Quarks			**Leptons**		
Gen.	Flavor	Charge	Mass [MeV]	Flavor	Charge	Mass [MeV]
I	u	$+2/3$	1.5..5	ν_e	0	$< 1.5 \times 10^{-6}$
	d	$-1/3$	3..9	e^-	-1	0.511
II	c	$+2/3$	1100..1400	ν_μ	0	$< 1.9 \times 10^{-4}$
	s	$-1/3$	60..170	μ^-	-1	105.7
III	t	$+2/3$	$170..179 \times 10^3$	ν_τ	0	< 18.2
	b	$-1/3$	4100..4400	τ^-	-1	< 1777.1

Table 1.1: Fundamental fermions of the Standard Model

of these particles are considered to be point-like and structureless.

Symmetries and invariance principles

Symmetries and invariance principles control physics in a most profound way. A symmetry is an invariance of an object or a system to a set of changes (transformation). The way on which space and time are constructed can be described on the basis of a continuous symmetry, i.e. the transformation operations can have arbitrary values (e.g. a rotation of a body under 23°). There are also discrete symmetries, where the symmetry operation comes in distinct steps (e.g. a reflection, which can't be cut in half). The mathematical framework in which symmetries are treated, is group theory. There, the symmetry operations form algebraic systems, called symmetry groups. The continuous group consisting of all rotations in $n-$dimensions for example is known as SO(n), where SO means that all matrices in that group are orthogonal and have determinants equal to one (S =special). The generalization to complex space leads to the SU(n) symmetry groups, where U means unitary. The SO(3) symmetry describes therefore the rotational symmetry of our world (for more information, see [2]).

Emmy Noether derived in 1918 that the (continuous) symmetries of space and time control the dynamics. This theorem, known as the *Noether-Theorem* states, that every symmetry in nature corresponds to a conservation law and therefore qualifies every conservation law an underlying symmetry [3]. The conservation of momentum, angular

symmetry		**conservation law**
space translation	$\leftrightarrow$	momentum
rotation	$\leftrightarrow$	angular momentum
time translation	$\leftrightarrow$	energy
gauge transformation	$\leftrightarrow$	charge

Table 1.2: Symmetries and the corresponding conservation laws

momentum and energy can therefore be considered a consequence of the invariance of the laws of physics under space translations, rotations and time translations, respectively. In a similar way one can prove in a quantum mechanical treatment, that the invariance of the laws of physics under gauge transformations[1] is the symmetry accounting for the conservation of charge (see Table 1.2).

In modern particle physics, discrete symmetry operations play an important role. Until 1957, it was believed that parity were an exact symmetry, i.e. that the operation of reflecting a system in a mirror would not change the law's of physics. However, Lee and Yang proved that parity could be perfectly respected in the strong and electromagnetic interactions, but that the weak force ignores this symmetry. This was a revolutionary idea, that the forces of nature may have different levels of symmetry. Parity violation, however, was explainable due to the fact that e.g. neutrinos (which only interact weak)

[1]In electrodynamics, the potentials are not uniquely defined and shifts in the four vector potential $A_\mu \rightarrow A_\mu + \delta_\mu \chi$ by a scalar field χ (gauge transformation) do not change the physical fields $F_{\mu\nu}$

appear only with 'left-handed' chirality and anti-neutrinos only with 'right-handed' chirality. The determined chirality of neutrinos leads immediately to the violation of another discrete symmetry; the charge conjugation (C). Replacing all left-handed neutrinos by anti-neutrinos would result in left-handed anti-neutrinos, which are not observed, therefore the C-Parity is violated in weak interactions, too.

Today we know that even the combined operation of charge conjugation and parity transformation (CP) is not conserved in the weak interactions (even though it would be a valid operation considering the chirality).

Now it is a necessary condition in quantum mechanics, that the combined operation of charge conjugation, parity transformation and time reversal (CPT) must be an exact symmetry. It turns out that if CPT would fail as an exact symmetry, then probability may not be a conserved quantity over time [4].

Fundamental Interactions

The interaction between fermions is described in the Standard Model as an exchange of gauge bosons. Theoretically, this concept is applied to the electromagnetic, strong and the weak interactions using quantum gauge field theories[2]. In such a theory, complex fields are associated with fundamental particles and are used to construct the Lagrangian of the dynamic system.

It is a basic postulation in theoretical physics that field theories should be invariant under local gauge transformation, i.e. the field theory should comprise an inner symmetry property. This postulation means now, that the Lagrangian of the field theory has to be invariant under transformations like

$$\psi(x) \rightarrow e^{i\theta(x)}\psi(x) \tag{1.1}$$

where $\psi(x)$ corresponds to a particle field, and $\theta(x)$ is a local phase, depending on space-time. The requirement that field theories have to be invariant under local shifts in gauge, causes new terms to appear in the Lagrangian. These terms have to be compensated by introducing gauge field terms to leave the Lagrangian invariant. At the same time, interaction terms between the particle fields and the gauge fields appear.

The requirement that the Lagrangian has to be invariant under local gauge shifts and the existence of an internal symmetry generates the Standard Model gauge bosons.

The gauge boson of the electromagnetic interaction, treated by quantum electrodynamics (QED), is the photon mediating the interaction of particles carrying an electric charge. The underlying symmetry group is U(1), which means, that QED is invariant under (local) phase transformations.

The strong force is described in the Standard Model by quantum chromodynamics (QCD), a gauge field theory which has been developed according to the successful QED but with underlying SU(3) symmetry (QCD is invariant under (local) transformations in color space). The theory introduces a new degree of freedom for the quarks known as color. The interaction between these colored particles is mediated by eight gauge bosons,

[2]The fourth known fundamental force, gravity, has to be excluded due to problems in its formulation as a quantum field theory

the gluons. Quarks can occur in three different colors while gluons always carry color and anti-color. The gluons being colored can interact among themselves. One more feature of the strong force is that at high energy scales, and therefore small distances, the strong interaction becomes weaker, or 'asymptotically free'. On the other hand, at low energies and accordingly larger scales, the interaction between the particles becomes stronger. This behaviour explains why colored objects have not yet been found in isolation, but only confined in color neutral objects.

Force	**gauge bosons**	**symmetry group**	**acts on**	**rel. strength**
strong	gluons (g)	SU(3)	colour	20
electromagnetic	photon (γ)	U(1)	charge	1
weak	$W^{\pm}, Z^0$	SU(2)	flavor	10^{-7}
gravity	graviton[3](G)	-	mass	10^{-36}

Table 1.3: Fundamental Interactions and gauge boson

The weak force is described in the so-called electroweak model. This unified framework of Glashow, Salam and Weinberg shows that the electromagnetic and the weak force can be treated like two aspects of the same unified interaction. Here the underlying symmetry group is the combined SU(2)×U(1) group. Their massive gauge bosons (beside the massless photon), $W^{\pm}$, Z^0 were discovered at CERN in 1983 [5][6].

A stumbling block in the development of an electroweak gauge field theory was that massive gauge bosons break the locale gauge invariance and renormalizability of the field theory.

To get around this problem, Weinberg and Salam applied the so-called Higgs mechanism [7] to the electroweak theory. The Higgs mechanism allows for some of the bosons to aquire masses by a so-called 'spontaneous symmetry breaking' process and lead to the development of an electroweak theory which was again invariant under local shifts in gauge.

The principal idea behind the concept of spontaneous symmetry breaking is that the ground state of the theory may have less symmetry than the equations of motion (or Lagrangian).

A familiar example of this phenomenon is the ferromagnet. Here the equations of motion (Maxwell's equations) are rotational invariant. In a system above the Curie temperature, the magnetic moments of the atoms have no favored orientations and the spins can freely point in any direction. However, as the temperature drops below the Curie temperature, the spins spontaneously align in a random direction. The ferromagnet receives a finite magnetization and it would require a force to redirect the spins again. The original rotational symmetry is therefore broken, and the carrier of the magnetic moments 'look' as they became 'massive' with respect to their rotational degree of freedom. This may be interpreted as a kind of phase transition the ferromagnet experienced [8].

Higgs respectively derived a theory how the massive gauge bosons of the electroweak interaction could have acquired mass. According to his theory, which is today known as

[3] So far, there is no experimental evidence for the existence of a graviton

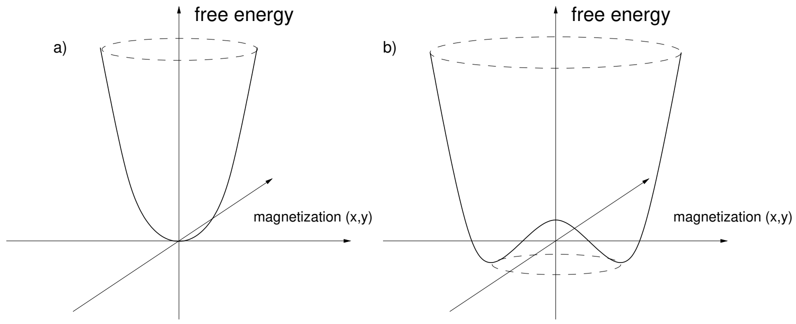


Figure 1.1: The free energy in 2 dim.: a) above the Curie temperature (T_C) b) below T_C (with broken symmetry) as a function of the magnetization M_x and M_z

the Higgs-mechanism, the initial massless W, Z and γ experienced a phase-transition with breaking symmetry, at some early stage of the universe. As a result, the W's and the Z became massive, while the γ remained massless.

Massless gauge bosons result in an infinite reach of the mediated interaction and therefore an $1/r$ potential. For massive gauge bosons of mass m, the potential is shielded by a mass dependent term and given by the Yukawa potential $\sim e^{-\frac{mc}{\hbar}r}/r$. The 'weakness' of the weak interaction results therefore from the mass of the gauge boson.

In the Standard Model, the masses of particles are explained as an interaction of massive particles with a permanent background field, the Higgs-field. In the simplest version of this theory, the Higgs-field is assumed to be a complex doublet with four real components. Three degrees of freedom are used to explain longitudinal degrees of freedom for the spins of the three massive gauge bosons $W^{\pm}$ and the Z. The fourth degree of freedom corresponds to an excited state of the Higgs-field. This excited state should, if the Higgs field exists, be observable as a massive particle, the Higgs boson.

Up to the time of writing this thesis, the Standard Model has been a very successful theory. A major success was certainly the correct prediction of the electroweak vector bosons at CERN 1984 [5, 6]. The top quark was finally discovered in 1995 at Fermilab [9, 10], leaving the Higgs boson the only remaining undiscovered particle of the Standard Model.

1.1.2 Physics beyond the Standard Model

Even though experimental results show a stunning agreement with the predictions of the Standard Model, crucial questions remain:

- What is the origin of the electroweak symmetry breaking mechanism?
- Why can't the Standard Model predict fermions mass?
- What is the reason of having three generations of fermions?
- Why is there more matter than antimatter in the universe?
- What is the reason for CP-violation?

- Is the combined transformation of charge, parity and time (CPT) an exact symmetry?
- What is dark matter in the universe made of and what is its origin?
- There are 18 free parameters in the Standard Model, which must be put in by hand, like the masses of fermions and bosons or the coupling constants of the interactions.

Grand Unification

To obtain a single theory, describing all fundamental interactions in nature, theorists attempt to unify the strong with the electroweak interaction according to the electroweak unification. In such a theory, all particles interactions can be described by only one coupling constant α_G. The two coupling constants of the electroweak interaction α and α', and α_s of the strong interaction, would then be related to α_G. As the coupling strength is a function of the four-momentum transfer Q, all three interactions should have the same coupling strength at some large energy M_p. As predicted by electroweak theory and quantum chromodynamics, M_p is expected to be around 10^{15} GeV (Planck-scale). Figure 1.2 shows the variation of the coupling constants with Q.

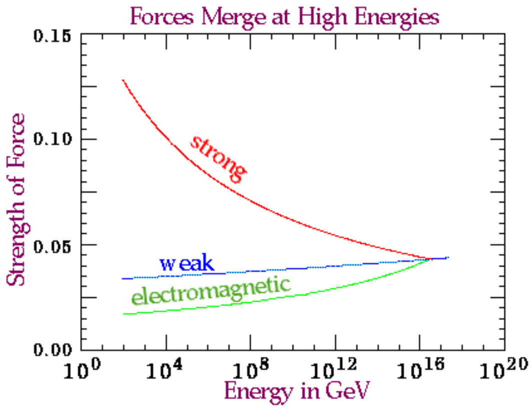


Figure 1.2: The strength of forces merge at high energy [11].

Supersymmetry

All supersymmetric extensions of the Standard Model propose a symmetry between fundamental fermions and bosons. In such a theory it is proposed, that every particle we observe, has a massive supersymmetric partner with a spin differing by $\frac{1}{2}$. The supersymmetric partners of the fermions and bosons would therefore be spin-0 squarks and sleptons and spin-$\frac{1}{2}$ gluinos, charginos, and neutralinos. At some stage, the symmetry between bosons and fermions must have been broken to explain why SUSY particles have not yet been detected.

1.1.3 Proton-Proton Physics

Protons are composed of three valence quarks which account for their quantum numbers. Next to these, there is a sea of virtual quark/anti-quark pairs and gluons which in average do not contribute to the protons quantum numbers. These constituents of the proton are known as partons.

A high energy proton-proton collision may be viewed as an interaction of two partons carrying only a fraction x_1 and x_2 of the total momentum of the proton. The colliding partons may undergo an inelastic scattering or annihilate and produce new particles with a reduced center of mass energy given by $\sqrt{x_1 x_2 s} \approx 0.1\sqrt{s}$.

The other partons, not involved in the interaction, mainly will hadronize into color neutral objects and form so-called hadronic spectator jets (see Figure 1.3).

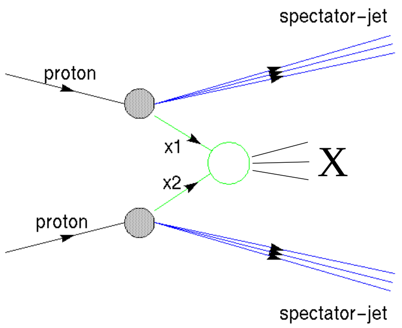


Figure 1.3: Parton parton interaction in high energy collision

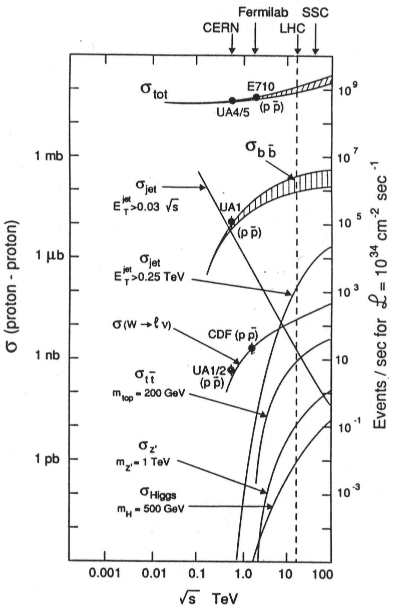


Figure 1.4: Cross-section of proton-proton interactions [12]

The probability of a parton carrying the fraction x[4] of the proton momentum is given by the experimentally measured structure function $F_2(x)$.

However, a fraction of the processes occurring in proton-proton collisions are due to elastic interactions with low momentum transfer between the protons. The cross section for several proton-proton collision final states is given in Figure 1.4 as a function of $\sqrt{s}$. It can be seen that interesting processes, like the production of heavy quarks and bosons are rare. The signal to background ratio for these events is very tiny (e.g. $\sigma_{Higgs}/\sigma_{tot} \approx 10^{-11}$). The need for a selective trigger system on the basis of interesting final state signatures is therefore rather obvious.

[4] x was initially introduced as the Bjorken scale variable, but can be interpreted as the fraction of the proton momentum a parton carries

1.2 The Large Hadron Collider at CERN

The *Large Hadron Collider* (LHC), currently under construction at the *European Laboratory for Particle Physics* (CERN) will accelerate proton beams to almost the speed of light (99.9999991% c) and collide them head on at four dedicated interaction points. This collision will lead to a total center-of-mass energy of $\sqrt{s} = 14$ TeV which will be orders of magnitudes higher than previous accelerator's have reached. This very high center-of-mass energy and the exceptional luminosity[5] achievable at the LHC allows experimentalists not only to study the structure of matter at the tiniest scale ever, but probably also to create new particles which have never been created in the laboratory before. The LHC will therefore have the potential to open the door to new physics and give new insights into the understanding of nature.

In keeping CERN's cost-effective strategy of building on previous investments, it has been decided that the LHC shares the 27 km existing LEP[6] tunnel. In doing so, there will

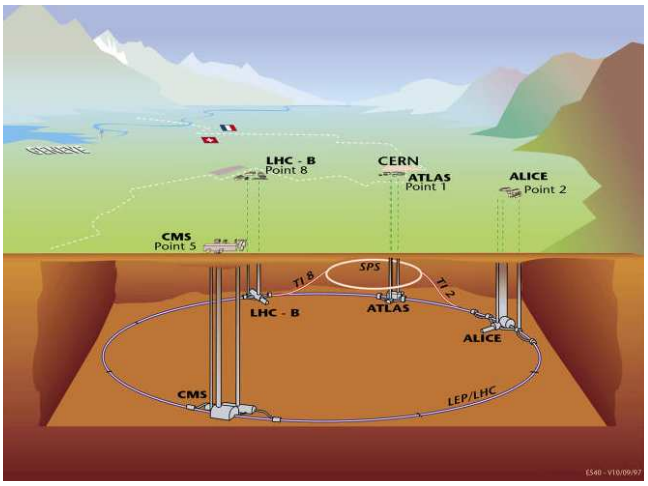


Figure 1.5: Schematic view of the *Large Hadron Collider* (LHC)

be the option to upgrade to a high energy electron-proton Collider in the future.

At a low luminosity stage (first 2-3 years of operation) the luminosity is planned to be $\mathcal{L} = 10^{33} cm^{-2}s^{-1}$. At a later stage, the luminosity will be progressively increased to $\mathcal{L} = 10^{34} cm^{-2}s^{-1}$. This will be achieved by filling each of the two rings with 2835 bunches of 10^{11} particles. The bunch separation is 7.48 m in length and 24.95 ns in time. That corresponds to a bunch-crossing rate of 40.08 MHz, a frequency at which most of the read-out and trigger electronics needs to work. A collection of the LHC performance parameters is given in Table 1.4.

The Collider will also operate for heavy-Ion physics. It will therefore be possible to accelerate Pb-Ions to a center-of-mass energy of $\sqrt{s} = 1250$ TeV at a luminosity of $\mathcal{L} = 10^{27} cm^{-2}s^{-1}$. In contrast to the LEP Collider where the energy-loss through synchrotron radiation was the limiting factor of further beam increase, the major challenge of the

[5] The luminosity $\mathcal{L}$ is the number of interactions per area and time

[6] LEP: Large Electron Positron Collider

LHC performance parameter	
center of mass Energy (pp)	14 TeV
center of mass Energy (PbPb)	1250 TeV
Circumference	27 km
Dipole field	8.4 T
Luminosity	$10^{34}\text{cm}^{-2}\text{s}^{-1}$
Injection energy	450 GeV
Circulating current/beam	0.54 A
Bunch spacing	25 ns
Bunch separation	7.48m
Particles per bunch	10^{11}
Stored beam energy	334 MJ
Beam lifetime	22 h
Luminosity lifetime	10 h
Energy loss per turn	6.7 keV
Total radiated power per beam	3.6 kW

Table 1.4: LHC performance parameters

LHC will be the magnet system. To force the 7 TeV proton-beam onto a circular orbit, a dipole field of 8.3 T is needed. This will only be possible by aligning 1296 state of the art superconducting magnets around the LHC tunnel. The two beam pipes for the counterclockwise accelerated beams of the LHC and the superconducting coils will be hosted in the same superfluid helium cryostat at a temperature of 1.9 K.

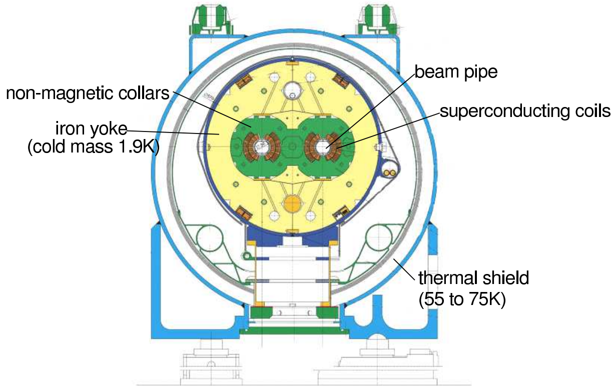


Figure 1.6: Schematic view of an LHC two-in-one dipole [13]

1.2.1 The Experiments and their Discovery Intentions

Because of its high center of mass energy, the LHC will offer a very broad range of opportunities for physics studies. For this reason, four dedicated experiments, with different discovery intentions, have been approved and are under construction.

The **ATLAS**[7] collaboration is building a general purpose detector, which is designed to exploit the full discovery potential of the LHC. The detector optimization is guided by the physics aim to be sensitive to the largest possible Higgs-boson mass range. Moreover, the detector should be capable to discover sparticles predicted by supersymmetric theories, heavy W -and Z like objects, compositness of fundamental fermions and, in general, any other physics beyond the Standard Model.

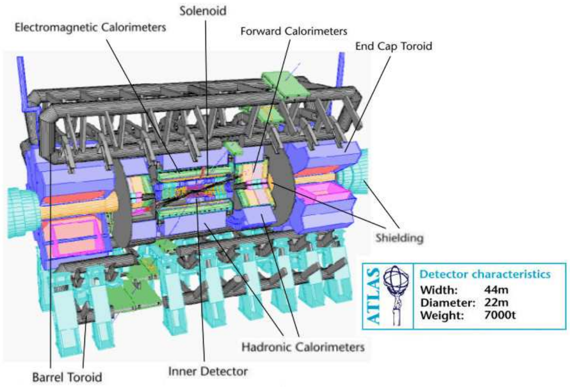


Figure 1.7: The ATLAS detector [14]

People working at **ALICE**[8] will build a dedicated heavy-ion detector to exploit the nucleus-nucleus interactions at the very high LHC energies. These kind of experiments are often referred to as 'mini big bang' experiments and the aim is to study strongly interacting matter at very extreme energy densities, similar to the conditions right after the natural big bang. There, the formation of a new phase of matter, the quark-gluon plasma, is expected. The existence of such a phase and its properties are a key issue in QCD for the understanding of confinement and of chiral-symmetry restoration.

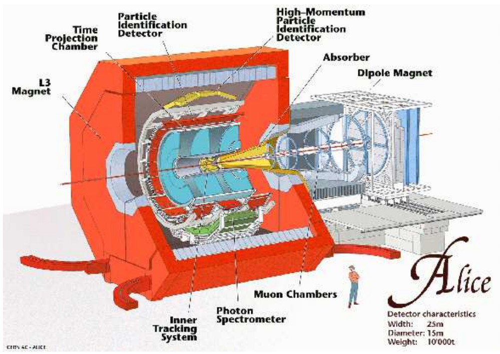


Figure 1.8: The ALICE detector [15]

[7] **ATLAS: A T**oroidal **L**HC **A**paratu**S**
[8] **ALICE: A L**arge **I**on **C**ollider **E**xperiment

The **CMS**[9] detector will, similar to ATLAS, be a general purpose detector with the same physics aims. It is also designed to cover important aspects of the heavy ion physics program.

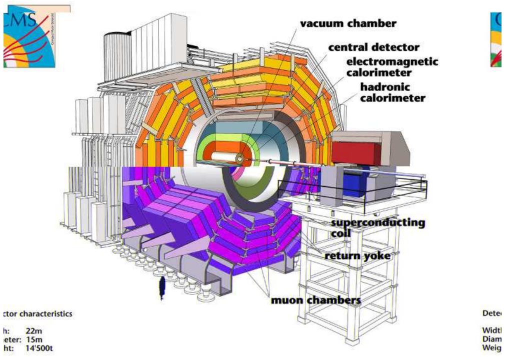


Figure 1.9: The CMS detector [16]

LHC-B[10] is a single-arm spectrometer focusing on the forward direction of the interaction point. The choice of the detector geometry is motivated by the fact that at high energies both the b and $\bar{b}$-hadrons are predominantly produced in the same forward cone. LHC-B will therefore be the ideal detector to study CP-violation in weak interactions, as these kind of processes are predicted to occur very frequently in the B-system.

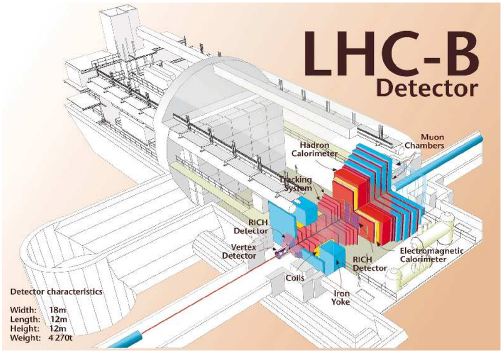


Figure 1.10: The LHC-B detector [17]

1.2.2 Higgs Physics at the LHC

Lifting the secret of the origin of the electroweak symmetry breaking mechanism is currently one of the most interesting issues in high energy physics and also one main goal for the construction of the Large Hadron Collider. The discovery of the Standard Model Higgs boson would be another great confirmation of the correct predictions of the Standard Model. According to the theory, the Higgs couples to particle anti-particle pairs with

[9] **CMS: C**ompact **M**uon **S**olenoid
[10] **LHC-B: LHC-B**ottom-quarks experiment

strength proportional to the particle masses. For fermions of a mass m_f the coupling is of strength $\sim \frac{m_f}{2M_W}$. For the massive gauge boson $W^{\pm}$ the coupling is $\sim m_W$ and for the Z it is $\sim \frac{m_Z}{cos\theta_W}$, where θ_W is the Weinberg angle.

The coupling to stable matter is therefore very small while the coupling to the *top* quark and the $W^{\pm}$ and Z boson is significant.

Higgs production

For all possible Higgs masses, the overall dominating production mechanism at the LHC will be gluon fusion. Figure 1.11 shows one possible process. The loop is largely dominated by the heavy top quark. At highest masses a significant part of the cross section would also come from vector boson fusion [18].

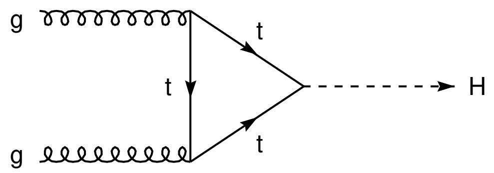


Figure 1.11: Higgs production at the LHC

Higgs decay and identification

The field of Higgs physics is rather large, as the Standard Model Higgs can decay directly into all massive particle/antiparticle pairs. Furthermore it can decay through loop diagrams into pairs of massless gluons and photons. The branching ratio of decays into various final states is shown in Figure 1.12. For the decay into ZZ and WW, the value indicated is the rate of decays into ZZ^* and WW^* if $M_H < 2M_Z$ and $M_H < 2M_W$ (W^* and Z^* mean a virtual Z and W).

The choice of which decay channel is used for the Higgs discovery, depends on the signal to noise ratio of the signature in the various mass range. Most decay channels suffer from small production rate and large QCD backgrounds. Table 1.5 lists the most promising decay channels for different Higgs mass ranges.

decay channel	**Higgs mass [GeV]**
H $\rightarrow bb$	$80 < m_H < 110$
H $\rightarrow \gamma\gamma$	$95 < m_H < 130$
H $\rightarrow ZZ^* \rightarrow$ 4l	$130 < m_H < 700$
H $\rightarrow ZZ \rightarrow$ ll$\nu\nu$, H $\rightarrow W^{\pm}W^{\pm} \rightarrow$ lνjj	$600 < m_H < 1000$

Table 1.5: Most promising decay channels for the Higgs discovery

However, recent results from LEP indicate that if there is a Standard Model Higgs boson, it could be light. An analysis of indirect electroweak precission measurements

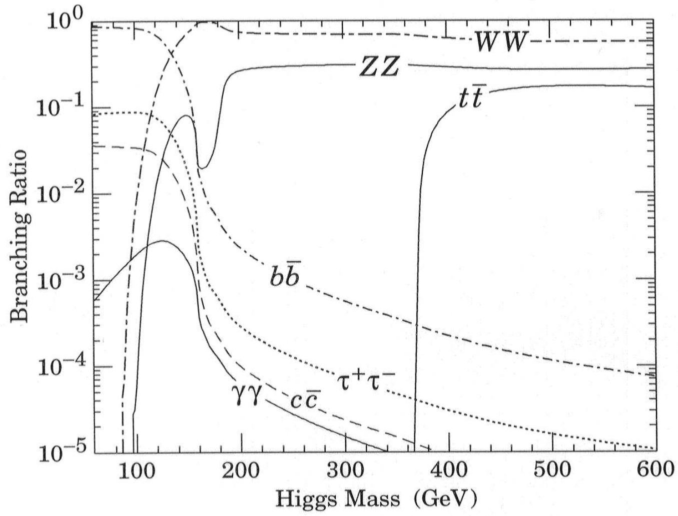


Figure 1.12: The branching ratio of the Higgs boson into $\gamma\gamma, \tau\tau$, b$\bar{\text{b}}$, t$\bar{\text{t}}$, c$\bar{\text{c}}$, ZZ and WW as a function of the Higgs mass [19]

performed by the LEP Collaborations on combined data from ALEPH, DELPHI, L3 and OPAL show, that the Higgs mass is most probable around 100 GeV. In Figure 1.13 the Higgs mass is plotted against $\Delta\chi^2$. The most probable Higgs mass lies in the minimum of this curve. The values with an 1σ deviation can be found at $\Delta\chi^2 = 1$. Direct searches from LEP rule out a Standard Model Higgs with mass less than 106 GeV. This was about the the upper limit, LEP could possibly explore. Active Higgs searches at LEP will now stop, in order to start the installation of the LHC into the existing tunnel.

1.3 The ATLAS Detector

The ATLAS detector will be one of the general purpose detectors at the LHC. The single detector parts have been designed by a worldwide collaboration of 34 member states, organized in 153 institutes around the world. ATLAS has been designed to operate at very high luminosities, enabling it to exploit the full discovery potential of the LHC and to be sensitive to all possible Higgs masses predicted by theory.

It is practical to define a coordinate system which makes use of the detector geometry. A spherical coordinate system is used with its origin at the interaction point. The beam axis is denoted z, the azimuthal angle ϕ is measured around the beam axis, and the polar angle θ is the angle from the beam axis. It is convenient to use instead of θ an additional angel called *pseudorapidity* $\eta = -\ln\tan\frac{\theta}{2}$. Differences in η are Lorentz invariant variables which consider the highly relativistic particle energies at the LHC.

The variety of final state signatures to which ATLAS has to be sensitive, has led to accentuation placed in the detector design on the detection of created particles and jets, over a big energy range (≈ 800 GeV). In addition, the detector should have a large acceptance in pseudorapidity and should be able to measure tracks, momenta and energy of particles and

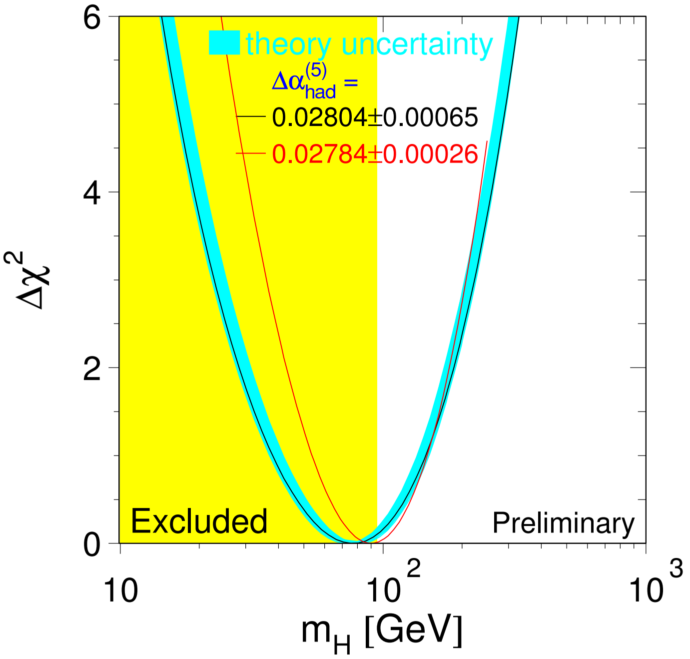


Figure 1.13: Result of LEP measurements indicate a 'light' Standard Model Higgs mass in the 100 GeV range [20]

jets very precisely and with a high resolution. For this reason, very high resolution and radiation hard detector components are under construction. An electromagnetic calorimeter for electron and photon identification and energy measurements. A full-coverage hadronic calorimeter for accurate jet and missing E_T measurements. An inner tracker system for measuring lepton-momentum and to identify and tag heavy quark flavors or τ leptons. And a muon detector, with the capability of measuring muon momenta at highest luminosities. Finally, a highly efficient trigger system is under development to filter among the vast number of physical events the tiny fraction of which are due to interesting physics (see Chapter 2).

In the following Sections only a brief description is given on the actual detector parts which are labeled in Figure 1.14. A more detailed description can be found in [21].

1.3.1 The Inner Detector

The Inner Detector is located inside a cylindrical cavity with a radius of 1.15m and a length of 7m surrounding the LHC beam-pipe. Its task is to track charged particles from the proton-proton interaction point to the electromagnetic calorimeter. It consists of three subsystems. The Pixel Detector and the Strip Detector (SCT), which are build from semiconductor strips in two and one dimensional segmentation respectively, and the Transition Radiation Tracker (TRT) (see Figure 1.15).

The whole parts are immersed into a solenoidal magnetic field of 2 T parallel to the beam axis. This allows momenta of charged particles to be measured from the deflection of

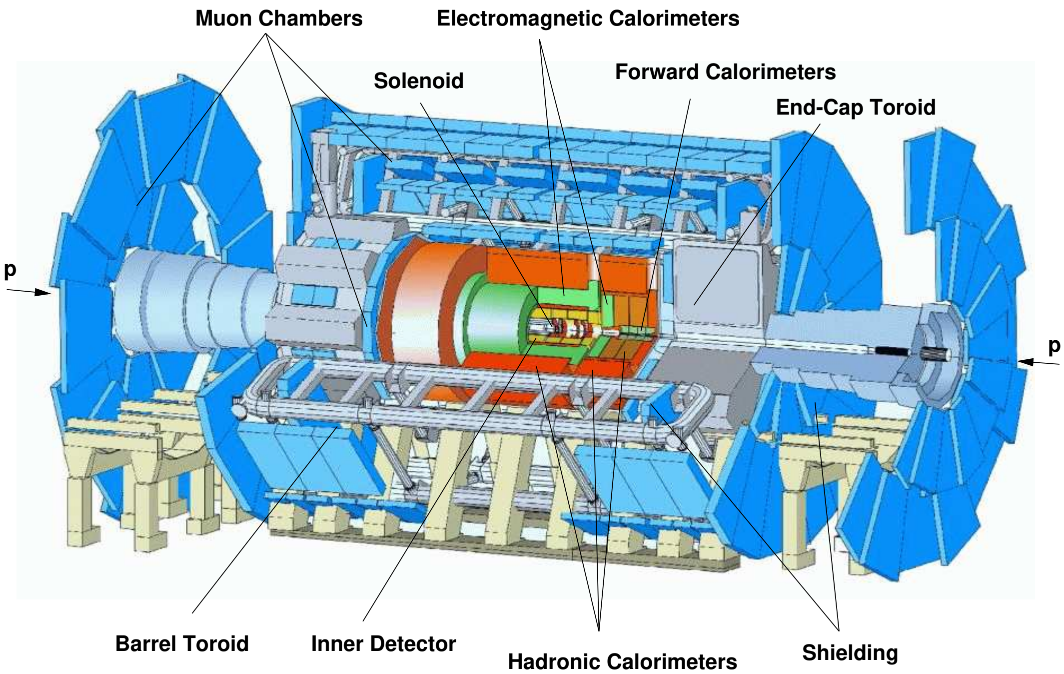


Figure 1.14: Schematic view of the ATLAS detector [22]

the trajectories in that field. The Pixel Detector is situated closest to the collision point. The design ensures the highest granularity to be able to accurately determine vertexes which are precise enough to tell whether a particle originated at the proton-proton collision point, or a few millimeters apart from it, due to a decay process.

The particle trajectories are tracked further with less granularity by silicon strip detectors in the inner parts and straw tube tracking detectors in the outer parts. This allows additional position measurements, to be done, further away from the vertex. The total number of high-precision layers used in the pixel detector, has been limited because of the high cost and the additional material they introduce. To many massive components in the tracker parts would degrade the particles momenta and spoil the energy measurements in the calorimeter.

For the straw tube detector, 70 % xenon gas has been employed to be able to identify electrons from transition radiation they produce - thus its name the Transition Radiation Tracker. Typically, three pixel layers, eight strip layers and 36 straw tube tracking points are crossed per particle track, resulting in a very precise trajectory reconstruction and momentum measurement.

1.3.2 The Calorimeter

The calorimeters will play an important role at ATLAS, as they will be the leading detectors for determining decay channels of prime physical interest.

They will be used to measure the energy and direction of photons, electrons, hadrons and jets as well as the missing transverse energy with a high accuracy over a range of

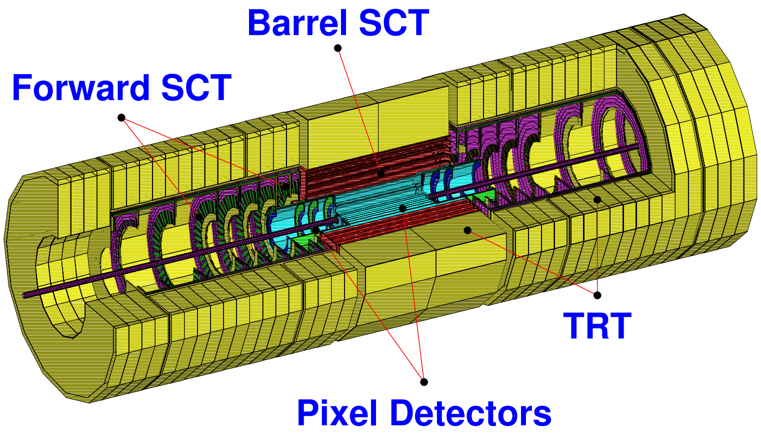


Figure 1.15: Schematic view of the Inner Detector [23]

$\eta < 5$. For this purpose the calorimeter is divided into an electromagnetic calorimeter, an hadronic calorimeter and a forward calorimeter.

It is important to note that the calorimeters measure only the transverse component of the energy depositions $E_T = E \sin\theta$. The longitudinal component does not carry any usable physical information, as the center-of-mass momentum of the colliding partons is not determined in longitudinal direction at the collision. However, the transversal component of the proton momentum can be neglected in comparison to its longitudinal component. The transversal component of the center-of-mass momentum of the colliding partons can therefore be assumed to be zero.

In general, calorimeters consist of a sandwich structure with layers of absorber material interspersed with layers of sensing material. In the absorber parts, the particles lose a fraction of their total energy through successive absorption processes and pair production of secondary particles. This is known as a shower, and as the shower develops the number of particles increase exponentially with depth, until the incident particle has deposited its total energy.

Different particles correspond to characteristic showers of secondary particles and can therefore be distinguished. The showers are detected by ionization processes in the sensing layers. A pulse shaper in the detector electronics transfers the ionization into a current which is proportional to the deposited energy and therefore to the incident particles energy.

The calorimeters are separated into a large matrix of cells, which allow to locate the energy deposition in the calorimeter. These cells are often referred to as channels and determine the resolution of the detector.

A major challenge of the calorimeter design is to build radiation hard detector parts, operating at the very high LHC luminosity, over a period of at least ten years.

The Electromagnetic Calorimeter

The Electromagnetic Calorimeter is a lead/liquid argon detector with a granularity of 200,000 channels. The combination of calorimeter and inner tracker is used for electron and photon identification.

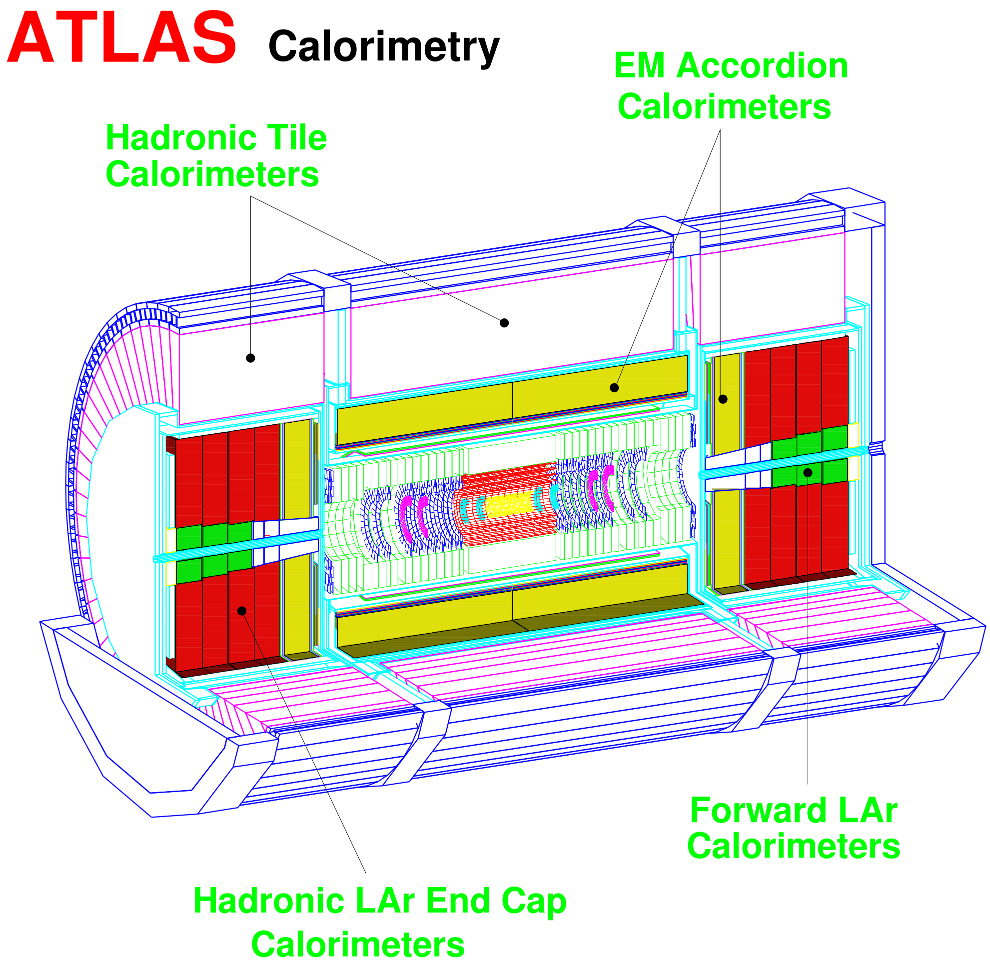


Figure 1.16: Cutaway view of the ATLAS calorimeter [22]

Lead acts as the absorber material where bremsstrahlung of the electrons and photons is the major contribution to the total absorption cross-section. The lead plates are bent into an accordion shape. This geometry allows the read-out towers to point towards the interaction region and read-out can be done with minimal space wasted, resulting in a hermetic design [24].

Liquid argon was chosen, because it is a relatively inexpensive Noble liquid, which have the appropriate properties for a sensing element in a calorimeter. It is tolerant to intensive radiation, is a good insulator, is relatively dense, and has the ability to let free electrons drift the required distances without getting captured and in this way does not spoil the energy measurement. The total thickness of the electromagnetic calorimeter is $> 24\ X_0$ (radiation lengths[11]) in the barrel and $> 26\ X_0$ in the end-caps.

[11] The radiation length X_0 is the absorber thickness over which the electron or photon energy is reduced by a factor e

The Hadronic Calorimeter

Hadrons pass through the Electromagnetic Calorimeter without losing much energy due to bremsstrahlung because of their relative high mass. For high energy hadrons, strong interaction with the absorber nuclei is the absorption process with the biggest contribution. The cross-section for this process differs for different materials and is larger for copper, tungsten and iron in comparison to lead. Steel absorbers are therefore used for the hadronic calorimeter in the outer parts, while copper and tungsten are used in the forward sections. These are always separated by tiles of scintillating plastic, as sensing material for the tile calorimeter and liquid argon for the hadronic forward calorimeter. The characteristic radiation lengths λ_0 for hadronic calorimeters is greater than that for their electromagnetic counterparts due to a less efficient absorption process. This is the reason why the hadronic parts are bigger in size.

1.3.3 The Muon Spectrometer

High momentum muons are among others very promising signatures for interesting physics occurring at the LHC. They have to be detected at the outer most parts of the ATLAS detector as they pass the tracker and calorimeter without interacting much. This is due to the fact that muons only interact electroweak and have a relatively high mass (table 1.1). Therefore they do not lose much energy through bremsstrahlung, making them very penetrating particles.

The Muon Spectrometer consists of multiple layers of high precision tracking chambers immersed into a system of large superconducting air-core toroid (the T in ATLAS) magnets. This allows precise determination of track coordinates and momentum to be calculated from the bending of the tracks in the magnetic field.

Over a large part of the η acceptance, three layers of *Monitored Drift Tube* (MDT) chambers are used. They consist of two multilayer of pressurized thin-wall aluminum drift tubes. For muons moving at very small angles to the beam pipe, cathode strip chambers (CSC) are used. These are multiwire proportional chambers and provide a finer granularity, to cope with the higher background conditions at small η.

Moreover, the Muon Spectrometer is instrumented with a separate trigger system. The muon trigger receives it's signals from dedicated trigger chambers located on the outside of the precision chambers. Namely these trigger chambers are the *Resistive Plate Chambers* (RPC) in the barrel and *Thin Gap Chambers* (TGC) in the end-cap region of the spectrometer. The RPC is a gaseous parallel plate detector while the TGC is a multiwire gas chamber. These trigger chambers cover a total area of 6550 m^2.

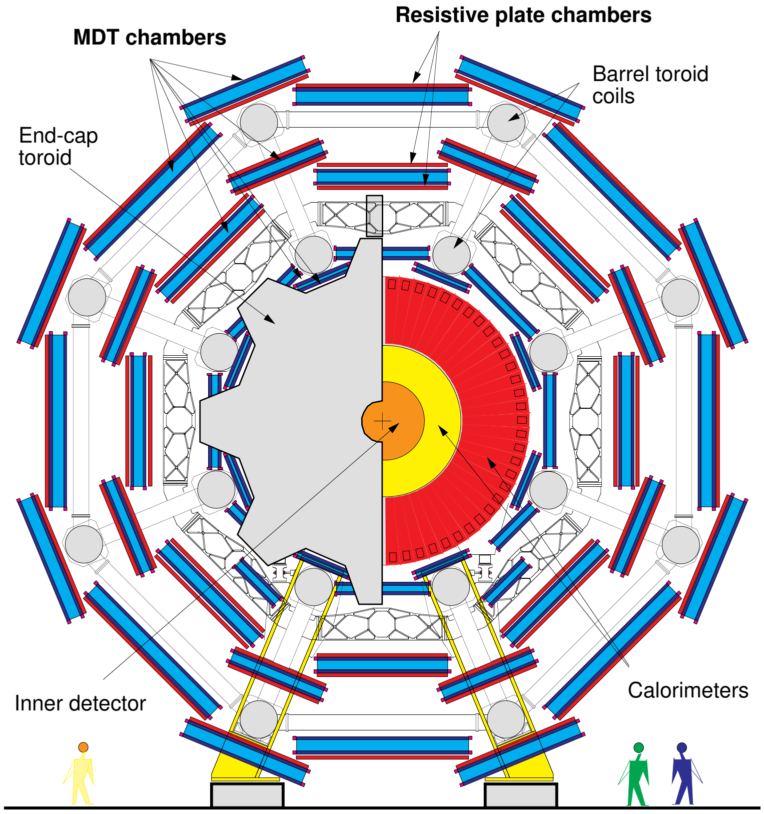


Figure 1.17: Side view of the ATLAS muon spectrometer [25]

Chapter 2

The ATLAS Trigger System

The high interaction rate at the LHC and the vast amount of detector channels, place tremendous requirements on the data acquisition system. To enable a complete event reconstruction, the entire detector has to be read out per bunch-crossing. In Table 2.1 the amount of data produced per bunch-crossing is listed for several detector parts. The data rate produced by the detector, at the LHC bunch-crossing rate of 40 MHz, sums up to 1.28 MB $\times$ $40 \cdot 10^6 = 50$ Tbyte/s. This data rate is roughly equivalent to twenty simultaneous telephone conversations by every person on earth! [11]. Obviously only a tiny fraction of the occurring events can therefore be recorded for off-line analysis.

Detector part	number of channels	amount of data [Kbytes]
Pixel detector	$1,4 \cdot 10^8$	50
Tracker	$5,6 \cdot 10^6$	850
Calorimeter	$2,3 \cdot 10^5$	180
muon spectrometer	$1,3 \cdot 10^6$	200
Total amount		1280

Table 2.1: Amount of data produced in the ATLAS detector parts for every bunch-crossing [26]

Fortunately not every proton-proton collision will produce physics of particular interest. That is the reason why a selective trigger system can be applied, choosing among the enormous amount of events the small fraction which is of physical interest. Since such a trigger decision is impossible to make within two succeeding bunch-crossings (25ns), the full detector read-out data has to be held in huge pipeline memories until the trigger has performed a decision whether the event, coresponding to a bunch-crossing, is of interest or can be discarded. The time required for the trigger to do so, is called the latency. A very fast trigger decision is desired in order to reduce the size, and corresponding costs, of necessary pipeline memories.

The ATLAS trigger system has been divided into three logic levels of event selection; the Level-1 and Level-2 Trigger, and the Event filter (see Figure 2.1). Each level performs a rate reduction on selected events and therefore reduces the amount of read-out data

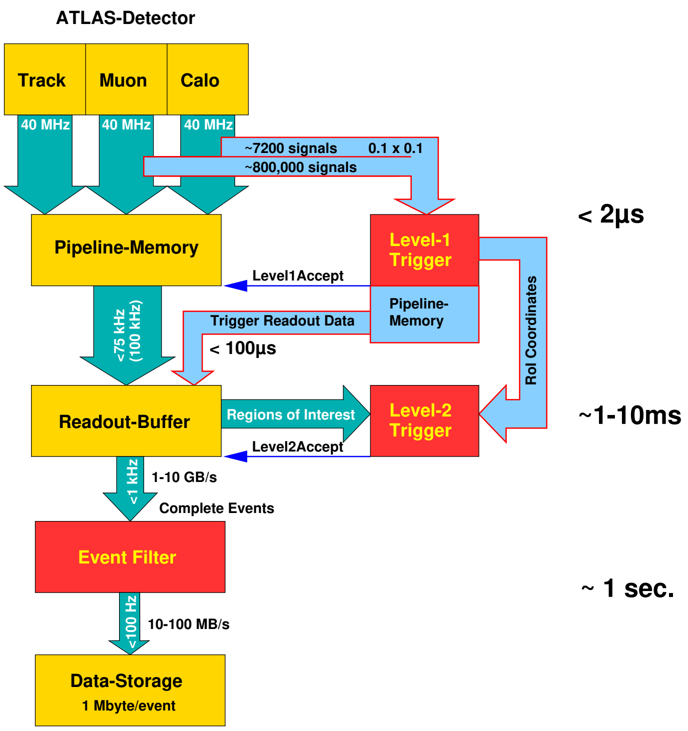


Figure 2.1: Detector read-out and trigger system [27]

passed to the following trigger level. In that way, higher trigger level will have more time (due to less data to process) in reaching a trigger decisions.

The Level-1 Trigger must accept the full LHC bunch crossing rate. This rate can only be handled by specialized electronics operating on data of lower granularity. At each bunch-crossing, read-out data from all detector parts of ATLAS, are shifted into pipeline-memories. After a fixed latency time (which determines the pipeline depth), the Level-1 Trigger indicates if the bunch-crossing was of physical interest or not. The accepted records are then shifted to the Read-Out Buffer memories and are hold there for the time the Level-2 Trigger needs to reach its decision. Accepted datasets are finally transferred to the Event Filter which reconstructs and builds full events. Only events which pass this final trigger level are passed onto permanent storage. In that way, the ATLAS Level-1 Trigger is able to reduce the data rate from the LHC (40 MHz bunch crossing rate, with an interaction rate of up to 1 GHz at a design luminosity of $\mathcal{L} = 10^{34} cm^{-2} s^{-1}$) by a factor of 10^7, to a level, which can be recorded for off-line analysis ($\leq$100 Hz). As can be seen from Figure 2.1 this will lead to a permanent data stream of 10 - 100 Mbyte/s for permanent storage on mass media.

2.1 The Level-1 Trigger

The Level-1 Trigger has to select bunch-crossings with interesting signatures from others and indicate them with a Level-1 Accept signal. It has been designed to reduce the event

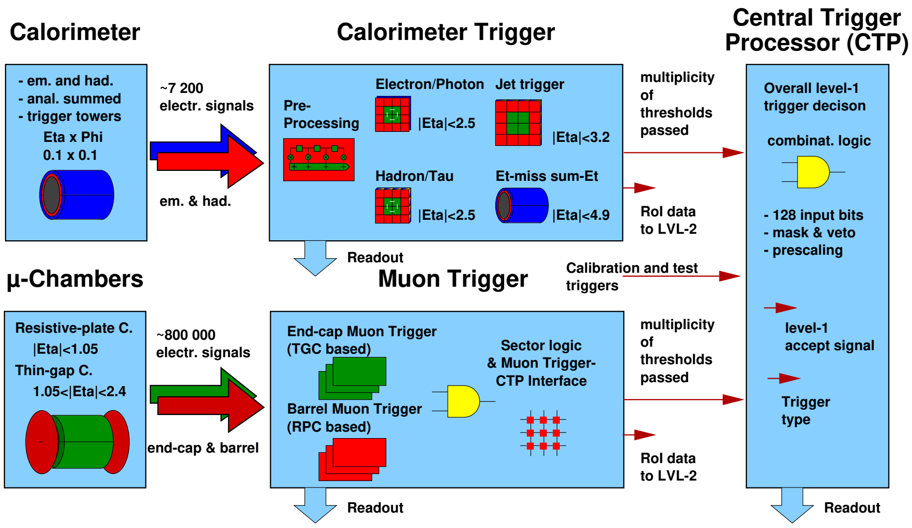


Figure 2.2: Level-1 calorimeter trigger architecture [27]

rate by a factor of 10^4 to 100 kHz which will be the input rate of the Level-2 Trigger.

The trigger algorithm searches for signatures involving isolated electrons, isolated photons, jets, muons and calculates the calorimeter global energy sum, to be sensitive to missing transversal energy (E_T) produced by particles which did not interact with the detector (e.g. neutrinos).

The trigger decision has to be made within a fixed latency of 2μs (80 bunch crossings) including cable delay. To achieve this, the Level-1 trigger is acting only on reduced-granularity data of the calorimeters and the muon spectrometer, to speed up the performance. In addition, all trigger algorithms are implemented in dedicated hardware chips where both, application-specific-integrated circuits (ASICs) and field programmable gate arrays (FPGAs) are used.

The Level-1 Trigger is divided into three subsystems, namely the Calorimeter Trigger, the Muon Trigger and the Central Trigger Processor (CTP). Figure 2.2 shows the overall architecture.

2.1.1 The Calorimeter Trigger

The input to the Calorimeter Trigger are 7200 pre-summed analog channels from the electromagnetic and hadronic calorimeter parts which are referred to as trigger towers. Each trigger tower signal corresponds to the transverse energy depositions in a calorimeter cell with a granularity of 0.1 $\times$ 0.1 in the η and ϕ direction. The signals are transmitted electrically via $\approx$ 60 m of twisted-pair cables from the detector parts to the trigger electronics, located in a trigger cavern. The Calorimeter Trigger will perform the following tasks:

- pre-processing of input signals
- trigger on electrons/photons
- trigger on single hadrons/taus

- trigger on jets
- calculate global energy sum in the calorimeter to be able to trigger on missing transversal energy (E_T)

All trigger algorithms identify so called trigger objects, according to their transverse energy deposition (E_T) in several trigger towers. The number of objects, passing a programmable energy threshold are counted. The multiplicity together with its threshold, are sent to the Central Trigger Processor. The coordinates and multiplicity of identified objects are sent to the Level-2 Trigger, which are referred to as regions-of-interest (RoI).

The pre-processing of the input signals is performed by the Pre-Processor System, which is currently under development at the Kirchhoff-Institut für Physik of the University of Heidelberg. For a description, see Chapter 3.

The electron/photon and hadron/tau algorithm will be implemented in the Cluster Processor (CP). It will identify isolated photons and electrons as objects of interest and increase the corresponding multiplicity. The algorithm is searching for trigger-tower cells with a transverse energy deposition which is greater than a given threshold. The trigger-towers are large enough to fully contain an electromagnetic shower produced by a single photon or an electron, but are small compared to the size of a jet core. An isolation criteria is therefore applied looking at a ring of neighboring trigger-tower cells, if E_T there is smaller than a given threshold one can be sure that the object is actually isolated and is most probable no jet.

The jet and E_T-miss algorithm will be implemented in the Jet/Energy-sum Processor. It allows the identification of jets and will be sensitive to events which will produce a missing E_T signature in the calorimeters. The jet algorithm is looking for energy depositions in the electromagnetic and hadronic calorimeter, which are greater than a certain threshold value. As jets are extended objects, the algorithm is applied to quadratic windows of several jet elements. A Jet element is formed by adding the electromagnetic trigger towers to the coresponding hadronic trigger tower.

The size of the window is a crucial parameter and influences strongly the performance of the jet-algorithm. If the window is chosen to small, it will not contain the full jet E_T and resolution will be lost, whereas choosing the window to big will result to large summation of noise and pile-up over the increased area which will also decrease the resolution.

The missing E_T-trigger processes trigger towers from all calorimeters in a pseudorapidity range of $\eta < 5$. This large acceptance is essential for avoiding mismeasurements of missing E_T due to jets falling outside the acceptance of the trigger. The E_T-miss algorithm sums over all missing transversal energy depositions of the calorimeter in x and y-direction and calculates the total missing E_T-value. The algorithm then searches for E_T values which are greater than a certain threshold value.

2.1.2 The Muon Trigger

The Level-1 Muon Trigger measures the trajectory for high and low transverse momentum (p_T) muons and provides data for the central trigger processor (CTP) and the Level-2 Trigger. It has to process hit information from dedicated finely segmented muon trigger chambers, organized in 3 stations, with a total number of $\approx$800,000 trigger channels.

These are the Resistive-Plate Chambers (RPCs) in the barrel and the Thin-Gap Chambers (TGCs) in the end-caps. For high-p_T muons, a coincidence in three stations of the trigger chambers is required to identify the muon track. For a low-p_T muon, a 2-fold coincidence is sufficient. The muon trigger algorithm looks at muons with an p_T-value exceeding a programmable threshold. The data to be sent to the CTP are the muon-candidate multiplicity values for several p_T thresholds. Data to be sent to the Level-2 Trigger include the multiplicity values and information of the candidates track (p_T range, position in η and ϕ).

2.1.3 The Central Trigger-Processor

The task of the Central Trigger-Processor (CTP) is to combine results from all different regions of the calorimeter and muon trigger to make a final Level-1 Trigger decision. This is done by combining the threshold multiplicities of the respective trigger algorithms into so-called *trigger objects*. From these trigger objects, eventually 96 *items* are formed by performing combinatorial logic on the trigger objects. Events can therefore be selected on the basis of menu items. An event is selected if it satisfies the criteria of one or more menu items (e.g. two muons and two electrons with p_T >20 GeV). The CTP distributes the Level-1 Accept, trigger type information, bunch-crossing number and event number via the Timing, Trigger and Control system (TTC) to the read-out and front-end electronics.

2.2 The Level-2 Trigger

The Level-1 Trigger passes its information on the regions-of-interests (RoI), including their position in η and ϕ to the Level-2 system. The Level-2 Trigger uses these information, to decide which data from each sub-detector are associated with each of the RoI and selectively copies data with the full granularity from the Read-Out Buffers. This selection of data for the Level-2 Trigger reduces the load on the Level-2 system and its complex algorithms. This trigger level has been designed to reduce the input data rate of $\leq$100 kHz by a factor of 10^3, to a data rate of $\leq$1 kHz, which will be the input rate of the Event Filter.

The Level-2 Trigger has to perform event processing tasks which are referred to as *feature extraction*, *object building* and *trigger type selection*. In the feature extraction phase, the data corresponding to the RoI is processed to give physics-like quantities (e.g. for the tracker, the basic hit information is converted to track-segment parameters). The object builder then takes these features and determines the particles parameters and, if possible, the particle type. In the trigger type selection phase, finally all the objects found in the event are combined, and compared with the topology (particles, momenta, missing mass) for a menu of physics processes.

2.3 The Event Filter

The final level of event selection will be carried out by the Event Filter, a farm of commercial available processors. The Event Filter should provide enough processing power to fully reconstruct the event and execute filtering algorithms which will allow it, to take the decision of accepting or rejecting the event on the basis of certain well-defined physics criteria. The Event Filter has to reduce the data rate to $\leq$100 Hz resulting in a storage rate of $\leq$100 Mbyte/s.

In contrast to the Level-2 Trigger, which processes copies of relevant event fragments, the Event Filter is a functionally part of the main data acquisition stream. The relevant data-stream is therefore processed in real-time by event filter programs, having the events in their processor memories. The ATLAS strategy for the event filtering algorithms is to use the offline reconstruction and analysis programs to a maximum possible extend. Exact selection criteria to be used are not yet chosen, since the final optimization will most likely be performed during the initial operation and data taking period.

In the LHC environment, the boundary between off-line and online computing will be less clearly defined than in previous experiments. Traditional off-line algorithms will be applied in real-time for event filtering and with an improved understanding of the detector, more and more tasks may migrate from the off-line to the online environment.

Chapter 3

The Pre-Processor System

The Pre-Processor links the ATLAS calorimeters with the Level-1 Calorimeter Trigger. It is the first unit, performing processing tasks, in the ATLAS Level-1 Calorimeter Trigger. Altogether the Pre-Processor receives 7296 analog trigger tower signals from all the ATLAS calorimeters. It has to provide the succeeding trigger processors with digitized and BC-identified data, containing the transverse energy deposition in the trigger towers. Moreover, read-out of raw trigger data is performed to allow monitoring of raw trigger data and cross-check of calorimeter read-out and trigger data. This allows to investigate the overall trigger performance.

3.1 Tasks of the Pre-Processor System

The Pre-Processor is of primary importance to the ATLAS experiment, as it is located at a very responsible position. All trigger input data, arriving with the full LHC bunch-crossing rate from the calorimeter, have to go through the system and have to be linked further to the succeeding trigger processors. Moreover, read-out and monitoring of raw trigger data will be performed by the Pre-Processor, enabling it to check the overall trigger performance. The pre-processing tasks can be summarized as follow:

- **Phase adjustment of input signals:** The time position of the sampling strobe, with respect to the analog input signals, can be adjusted in steps of 1 ns within a range of 25 ns. This is done by a dedicated ASIC (Phos4) developed by the CERN microelectronics group.

- **Digitization of analog trigger tower signals:** Every analog trigger tower signal will be digitized by an analog-to-digital converter (ADC) with 10-bit resolution which is commercially available from Analog Devices (AD9042).

- **Bunch-crossing identification (BCID):** Two algorithms are applied to identify to a trigger tower signal, the corresponding bunch-crossing in time. One algorithm has been developed for saturated signals and one for non-saturated signals. The logic to perform this task is implemented in a dedicated ASIC, the Pre-Processor Asic (PPr-Asic).

- **Calibration of digitized data:** The transverse energy value of 10-bit resolution is mapped onto an 8-bit value (after the BCID algorithm) and an inter-channel energy calibration is performed (PPr-Asic).

- **Formation of jet elements:** Four trigger towers are combined to coarser jet elements with size 0.2×0.2 in η and ϕ which serve as input to the Jet/$\not{E}_T$-miss trigger. The summing is done separately for the electromagnetic and the hadronic calorimeter, leaving the option to apply separate jet thresholds for the trigger algorithms (PPr-Asic).

- **Bunch-crossing multiplexing:** To double the effective bandwidth of the serial links to the Cluster Processor, a multiplexing scheme is applied. This schemes makes use of the nature of the non-saturated BCID algorithm, which produces empty time slices after it has identified a bunch-crossing. This empty time slice can therefore be used to transmit a value of a neighboring trigger tower (PPr-Asic).

- **Serial data transmission:** Trigger data on which pre-processing has been applied, is sent to the remaining Level-1 Trigger processors via high-speed serial links. This technology is used to keep the number of required links and hence costs to an acceptable amount. For this task commercial available LVDS links from National Semiconductor are used.

- **Read-out of raw trigger data:** Read-out and monitoring of raw trigger data allows to check the overall trigger performance. Checking the raw data, on which the trigger decision was based is a key feature in order to understand the trigger system. Moreover it is desirable to cross-check the calorimeter read-out and trigger data. A dedicated FPGA, the RemFPGA collects data on the board-level and a Read-Out Driver module collects data from connected RemFPGA's, via a custom pipelined bus (PipelineBus) on the crate-level.

3.2 System Architecture

The Level-1 Trigger decision has to be made within a fixed latency of 80 bunch-crossings including cable delay. This time has to be shared by all Level-1 Trigger processors and connection cables. The Pre-processor is allowed to use 18 bunch-crossings to perform the tasks listed above. A high integration density of the used components is needed to meet this stringent requirement.

A multi-chip-module (MCM) technology is therefore applied. All hardware chips required to perform the Pre-processor tasks for a limited amount of channels, are placed on one small printed-circuit-board (MCM). This compact design reduces signal delays between components and allows to have small exchangeable components. On the Pre-Processor Multi-Chip Module (PPr-MCM) 9 ASICs are placed as 'dies' which are capable to perform the pre-processing tasks for 4 trigger-tower channels. In detail these are 4 ADCs to perform the digitization task, one phase adjusting ASIC (Phos4) and 3 high-speed serializing ASICs (LVDS). The digital logic performed on four trigger towers is implemented

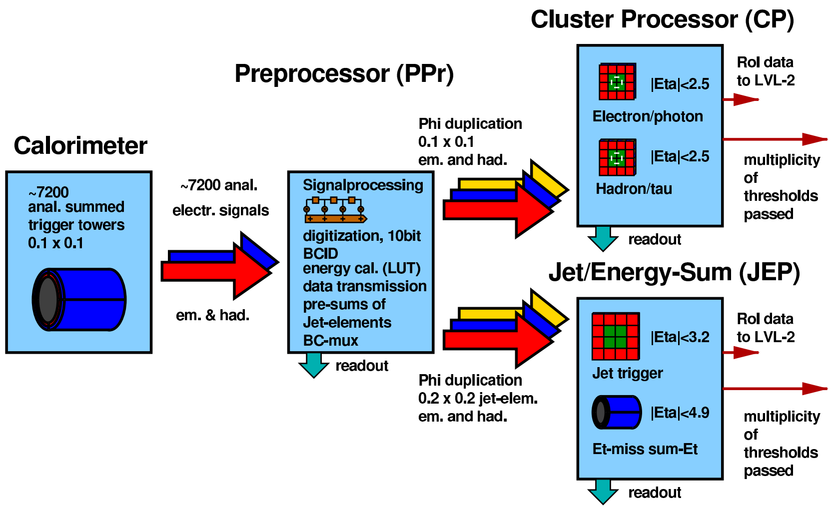


Figure 3.1: The Pre-Processor System within the Level-1 Calorimeter Trigger environment [27]

in a dedicated ASIC, the Pre-Processor ASIC. Its tasks include, bunch-crossing identification (BCID), synchronization of digital data from the ADCs, bunch-crossing multiplexing, calibration of digitized data and providing a serial interface for pipelined event-data read-out.

In order to be capable to process 7296 input channels, the Pre-processor system is organized in 128 identical so-called Pre-Processor Modules (PPM) which represent the main electronics of the system. Each PPM carries 16 PPr-MCM and one Read-Out Merger FPGA, therefore it can process 64 trigger-tower channels. Eight electronic crates are filled with 16 Pre-Processor Modules and 2 Read-Out Driver Modules, each.

The Read-Out Driver (ROD) collects the read-out data of 8 PPMs via a ring-like bus (PipelineBus) on the backplane of the VME crates, and sends the read-out data via a S-Link[1], to the Read-Out Buffers (ROB's) of the ATLAS data acquisition system (DAQ).

3.3 Read-out of Trigger Data

Read-out and monitoring of raw trigger data allows to check the overall trigger performance. Checking the raw data, on which the trigger decision was based, is a key feature in order to understand the trigger system. Moreover, it is necessary to check, if the trigger-data are consistent with data from the separated path of full-granularity read-out of the calorimeters.

In the Pre-Processor system, collection of read-out data is done in two steps. First it is collected on the board-level by the so-called RemFPGA using serial links to the PPr-Asics. A Read-Out Driver module then collects data from several modules via a custom bus (PipelineBus) system on the crate-level.

[1]S-Link: Standard Link, specified by CERN to connect RODs with the ROBs. For a more detailed

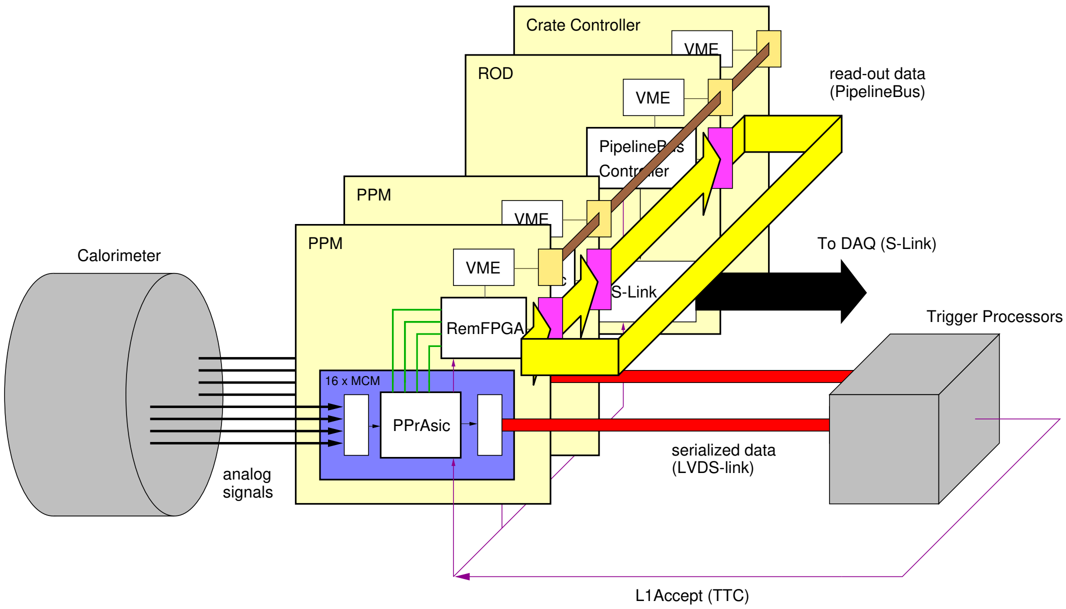


Figure 3.2: The PipelineBus of the Pre-Processor system [28]

3.3.1 The RemFPGA: Read-out on the board-level

One RemFPGA will be placed on each Pre-Processor Module (PPM). It will connect via serial interfaces to all 16 PPr-Asics on every PPM. If a Level-1 Accept signal is received, distributed by the Central Trigger Processor, the RemFPGA collects all event data from the PPr-Asics, and stores it as event block's in an internal buffer memory. On that data, data compression is performed which achieves a typical data reduction factor of 2.2-2.8. After compression, the data is made available to the PipelineBus interface.

For configuration, the RemFPGA receives data from the PipelineBus or from the local VME-bus interface. The PPr-Asic can be configured by data received from the same sources and the RemFPGA sends it further to the PPr-Asics.

3.3.2 The PipelineBus: Read-out on the crate-level

The PipelineBus interconnects 8 Pre-Processor Modules to one Read-Out Driver Module. The Read-Out Driver collects read-out data from connected nodes and sends them via a standard S-Link to the ATLAS data acquisition system (DAQ).

The PipelineBus can not only be used for read-out but also for sending configuration data to the connected modules. The Read-Out Driver takes then the role as a source of these data and sends them to all connected PPM's.

description see Chapter 6

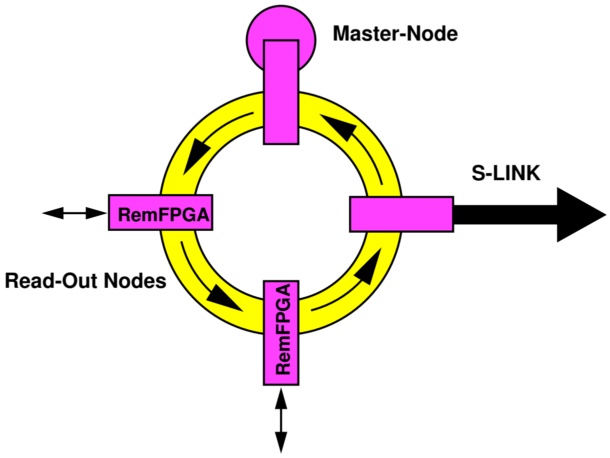


Figure 3.3: The PipelineBus Topology [28]

PipelineBus Topology

The bus itself is built from several nodes, connected by a 35-bit wide point-to-point connection in a ring-like fashion. All nodes receive a common bus clock signal, which is used to latch input data into the node and make output data available for the next node. Each node has to process the input data and to give out the same, or an appropriate response to the data at the next clock cycle. In this way, data are shifted from node to node through the pipeline formed by the nodes and their connections - thus the name PipelineBus.

Four different type of nodes exist. A *Master node*, *Read-out nodes*, a *S-Link node* and a *Monitoring node*. The Master controls the PipelineBus and can react to responses of other nodes by injecting commands. The Read-out node is the source of read-out data, or the sink of control data. The S-Link node will transfer read-out data to the ATLAS DAQ system via a standard S-Link read-out link and the Monitoring node accesses event samples to spy on read-out data.

PipelineBus Protocol

The 35 bus lines include one parity bit for transmission error checks and 2 control bits, identifying the bus word which consists of the remaining 32 bits of data. Three different bus words are distinguished: idle words, command words and data words. The fourth possible control bit combination ('b11) is unused and is considered an illegal bus state.

The idle word is the default state of the bus and means that the bus is currently unoccupied. Any idle word can be overwritten with a command or a data word. Commands are used to control the bus. They are injected into the pipeline by the master node and processed by all nodes that the command addresses. The command word is therefore split into bit-fields (see Figure 3.4). It contains an eight bit token-field, which identifies the command. A sixteen bit-field of argument data for commands which require parameters. This field is also used to contain return values, generated by a read-out node as a response to a command.

Two additional bits are available in a command word. An accept bit is set by a read-out node, after it has successfully processed a command. An error bit is set, if an error occurred. If a node detects an error, e.g. a parity error, this bit is set and an error code is put into the argument field. The bus master has to identify the error word and take appropriate actions.

Altogether 15 different commands are defined. There are commands for address configuration, control of read-out and input, status information etc. For a list of all available PipelineBus commands see Appendix A.

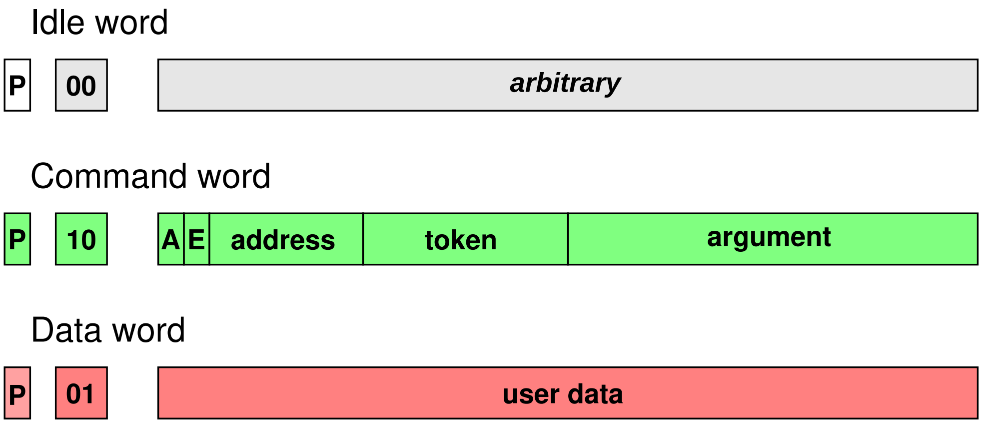


Figure 3.4: The PipelineBus protocol [28]

PipelineBus Operation

As an example for a typical PipelineBus operation, a read-out operation of two nodes is illustrated in Figure 3.5. Each step shown represents several bus clock cycles. The big horizontal arrow corresponds to the pipeline and indicates the propagation direction of the bus words. The two framed boxes represent two read-out nodes which have initially read-out data available. In step one, the bus master has injected the command `StartReadout` (S). The command propagates through the pipeline and when it reaches the first read-out node, the node enters read-out mode, which is indicated by the little arrow above the node. In read-out mode the node waits for the command `BeginOfData` (B), which activates the actual readout.

When the node receives the `BeginOfData` command, it looks for idle words following the command and overwrites them with its read-out data (step two). When a complete data block is written to the pipeline, the node terminates the process with an `EndOfData` (E) command. The argument of this command contains the number of the node, the data block belongs to.

In step three, the `BeginOfData` command has passed the second node. This node also looks for idle words to put its read-out data onto the bus. In this way, the second node lets pass the data block of the first node and then fills the idle words with its own data. It eventually terminates the block with a `EndOfData` command and the read-out operation is finished. The event block witch was built during this operation is now propagating

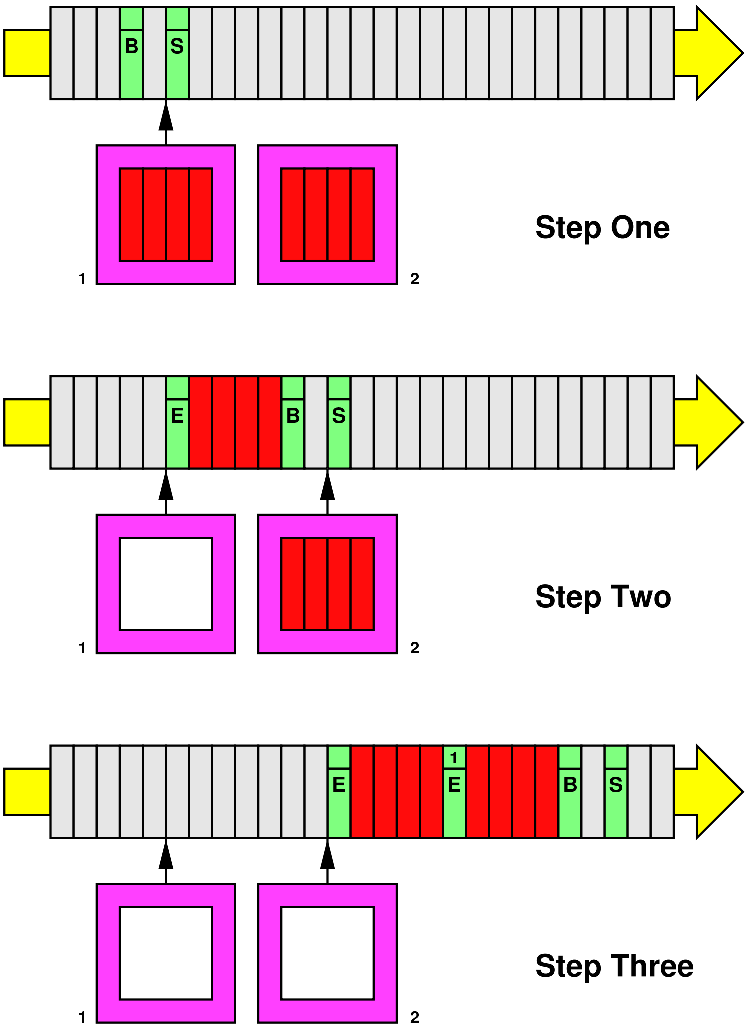


Figure 3.5: A PipelineBus operation [28]

through the pipeline to the S-Link node. This node adds header and trailer information to the event data and sends this data, which is now an ATLAS DAQ fragment, to the data acquisition system.

The structure of the PipelineBus fits well to the S-Link interface, and can be seen as a kind of extension to it.

PipelineBus Bandwidth

At a bus-clock frequency of 40 MHz the theoretical limit for the bandwidth of the Pipeline-Bus including protocol, results in 40 MHz $\times$ 32 bit = 152.6 Mbyte/s. This theoretical bandwidth is slightly reduced by idle cycles between successive read-out operations, and separator/trailer information generated by each Read-out node. The theoretical maximum bandwidth of the S-Link is basically the same as for the PipelineBus at 40 MHz.

The amount of data, which has to be read out, depends on the rate of Level-1 Trigger accepts and the number of raw trigger data on every Level-1 Accept. Currently it is desired to read out 5 raw trigger-tower signals with a 10 bit resolution and 1 BCID result with an 8 bit resolution, for every Level-1 Accept. The Level-1 Accept rate is planned to be between 75 to 100 kHz at most. The total amount of data produced is listed in Table 3.1 where it is assumed that one PipelineBus ring interconnects 8 Pre-Processor Modules with 64 trigger tower inputs each. Every Read-Out Driver in one PipelineBus ring, collects therefore data from $64 \times 8 = 512$ channels. The first row in the list, labeled with single event, is the amount of data produced from 8 Pre-Processor Modules for every single Level-1 Accept.

Level-1 Accept rate:	**single event**	**75 kHz**	**100 kHz**
BCID result (8 bit)	4096 bit	36.62 Mbyte/s	48.83 Mbyte/s
5 raw trigger towers (10 bit)	25600 bit	228.88 Mbyte/s	305.18 Mbyte/s
Event format overhead	64 bit	0.57 Mbyte/s	0.76 Mbyte/s
Idle cycle between read-out	256 bit	2.29 Mbyte/s	3.05 Mbyte/s
Total amount	30016 bit	268.36 Mbyte/s	357.82 Mbyte/s
Compressed by a factor 2.5	12198 bit	109.06 Mbyte/s	145.42 Mbyte/s

Table 3.1: Data volume to be read out and resulting data rates

At a 100 kHz Level-1 Accept rate, this sums up to a data rate of 357.82 Mbyte/s. This clearly shows that in order to cope with this data-rate, either the bus frequency has to be increased or data compression has to be performed on read-out data.

Tests have been performed on a prototype of the Read-Out Merger FPGA (RemFPGA). This prototype is an implementation of the read-out merger tasks into an ASIC, the RemAsic. Implemented compression algorithms in that ASIC have achieved compression rates of roughly 2.5 [29]. As can be seen from Table 3.1, this would be the factor which is needed to be able to read out the desired amount of 5 raw trigger towers with 10 bit resolution and one BCID result with 8 bit resolution. The resulting data rate would sum up to 145.42 Mbyte/s which can be processed by the PipelineBus.

If more data are needed to be read out, the bus frequency might be increased.

Chapter 4

Field Programmable Gate Arrays (FPGA)

The logic for several PipelineBus-nodes has been developed in the hardware description language Verilog. The designs have then been implemented on FPGA's from Xilinx Inc. The design process and the architecture of FPGA's is described in this Section from a very general perspective.

4.1 Introduction to FPGA

A *Field Programmable Gate Array* (FPGA) is a programmable logic device that can be configured for a wide variety of applications. In contrast to an ASIC, a FPGA does not get a specified functionality during manufacturing, but can be configured and even reconfigured by the user. It is therefore possible to change implemented designs very quickly without the need to change the actual hardware.

However, next to the number of components which account for the implemented logic, hardware components are necessary which make the flexible programmability available. FPGA devices therefore are usually bigger, slower and more expensive in bigger quantities than their ASIC counterparts. For applications, where flexibility is a major issue and the amount of needed devices is rather small, FPGAs are the ideal solution. In the Pre-Processor system, where the Read-Out Driver functionality is needed only on 16 equal Read-Out Driver Modules, the implementation in a FPGA is the desired solution.

4.1.1 Conceptional Superstructure of FPGAs

Basically, FPGA's consist of three major building blocks:

- **Logic blocks:** provide the functional elements for constructing logic and the possibility to store logic values in flip-flops

- **Input/Output blocks:** provide the interface between internal logic and the outside world (package pins)

- **Routing network:** provide interconnection of the logic blocks and the distribution of global clock and reset signals

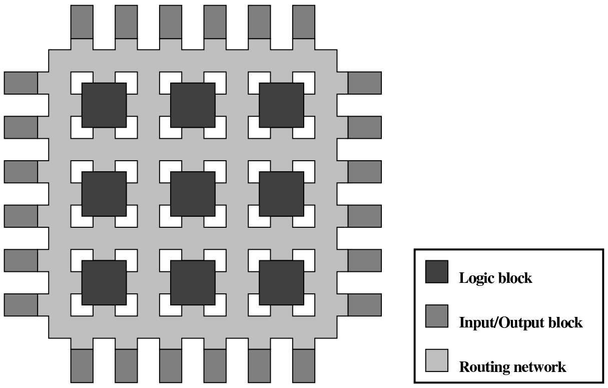


Figure 4.1: General structure of a FPGA [27]

A major problem of programming FPGAs next to the mapping of the logic circuit onto the logic blocks, is the routing of the logic blocks. In principal all logic blocks can be connected among each other. However, the amount of routing resources are limited and at some stage timing delay through more complex routing becomes a problem. It is the longest delay path that limits the speed of every design. It is therefore necessary to find a clever fragmentation of the logic blocks which allow short interconnections while all connections are established.

This task is done by complex software. The software follows an iterative approach to place logic blocks and to try to minimize path delays to yield best system performance. To minimize slack between user-design constraints and estimated timing is therefore the primary goal. It might happen that designs, according to the logic resources, should be able to be implemented in a certain FPGA but can't be routed to meet the given constraints. That is the reason why the available resources of FPGA's usually can't be used completely. However, the user can influence the place and routing process by applying several user constraints, which usually result in a better performance.

4.1.2 The Virtex FPGA from Xilinx Inc.

Xilinx Inc., introduced the first FPGA in 1985, holds patents on FPGA architecture and technology, and continues to be the leading supplier to this market. With the Virtex product family, they offer fast and complex devices that contain up to 1 million logic gates and several auxiliary features.

The general architecture of the Virtex FPGA follows the conceptional superstructure described in the last Section. Here, the logic blocks are called *Configurable Logic Blocks* (CLB), and the *Input/Output Blocks* (IOB) are named the same. The CLBs interconnect through a broad selection of routing resources throughout the device. A so-called Versa Ring interfaces to the IOB's to provide additional routing resources around the periphery of the device to adapt to constraints of I/O placement.

In addition to a dedicated global clock distribution network, the Virtex device offers a very powerful feature to generate multiples or fractions of the system clock. This is done by *Delay Locked Loops* (DLL) situated at every corner of the FPGA.

To meet the requirements of many customers dedicated block ram memories of 4096 bits each have been added to the basic component parts. All block ram cells may be individually be used either synchronous with the Clock or asynchronous.

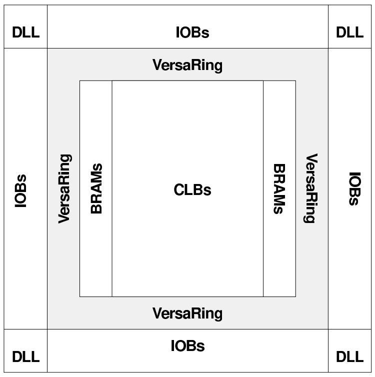


Figure 4.2: Virtex architecture [28]

The device is customized by loading configuration data into internal static memory cells. Values stored in these cells control the configurable logic elements and interconnect resources. The device may be reloaded at any time, to change its functionality.

Configurable Logic Block (CLB)

The CLB provides the FPGA with the basic flexibility and functionality. The basic building block of the Virtex CLB is the logic cell (LC). An LC includes a 4-input function generator, carry logic, and a storage element. Each Virtex CLB contains four LCs, organized in two similar slices, as shown in Figure 4.3. The function generator may embody any boolean operation with 4 inputs. The generator is implemented as a static RAM in which the truth table of the desired boolean operation is stored. The argument of the function is used as address of the RAM which returns the solution value to the boolean operation. That is often referred to as a lookup table (LUT). Via the carry and control logic, two or more function generators may be combined to realize operations with more input signals. The output from the function generator in each LC drives both the CLB output and the D input of the flip-flop. The function generators may be configured to serve as static rams to store 16 bits of data, each.

The flip-flops in the Virtex CLB's can be configured either as edge-triggered D-type flip-flops or as level-sensitive latches. The D inputs can be driven either by the function generators within the slice or directly from slice inputs, bypassing the function generators.

In addition to Clock (RC) and Clock Enable (EC) signals, each Slice has synchronous set and reset signals. These signals force a flip-flop into the initialization state specified for it during configuration.

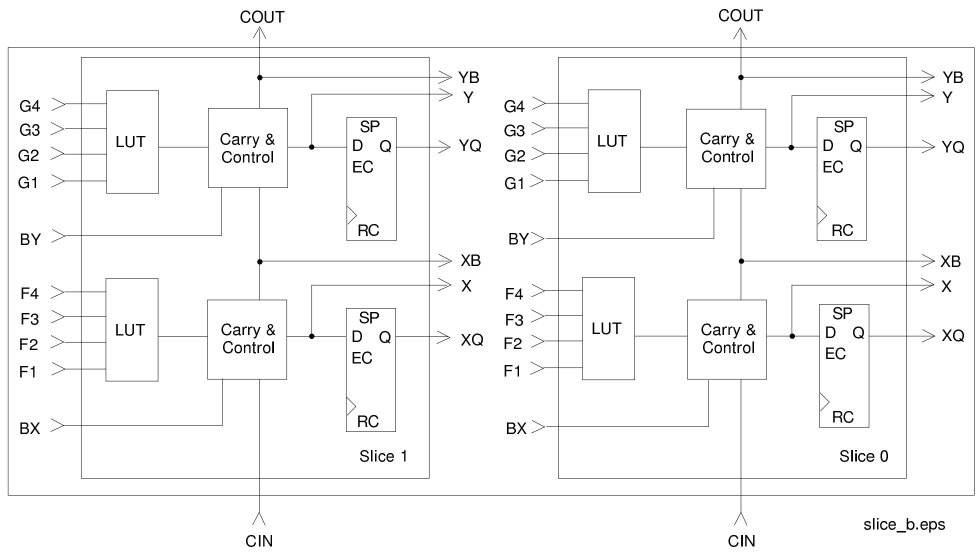


Figure 4.3: Configurable Logic Block (CLB) of a Virtex device [28]

Input/Output Block (IOB)

The IOBs provide the interface between the package pins and the CLBs. Three IOB storage elements function either as edge-triggered flip-flops or as level sensitive latches. The pins can therefore be used as Inputs, Outputs or bidirectional I/O's.

The Virtex IOB features a so-called SelectI/O input and output that support a wide variety of I/O signaling standards.

The input signal can either be routed directly to internal logic or through an optional input flip-flop. An optional delay element at the D-input of this flip-flop can be configured to adjust the input-timing.

The output signal can be routed to the buffer directly, or through an optional output flip-flop. Each output buffer can source up to 24mA.

An optional weak-keeper monitors, when selected, the voltage on the pad and weakly drives the pin to logic High or Low to match the input signal. Maintaining a valid logic level in this way eliminates signal uncertainties.

Block SelectRAM

All Virtex devices contain two columns of Block SelectRAM memories (see Figure 4.2). Each memory block has the height of four CLBs. It is a fully synchronous dual-ported 4096 bit RAM with independent control signals for each port. The data ports can be configured independently with different widths. If the width of the block RAM is increased, the depth is automatically decreased. The Virtex Block SelectRAM also include dedicated routing to provide an efficient interface with CLBs and other Block SelectRAMs.

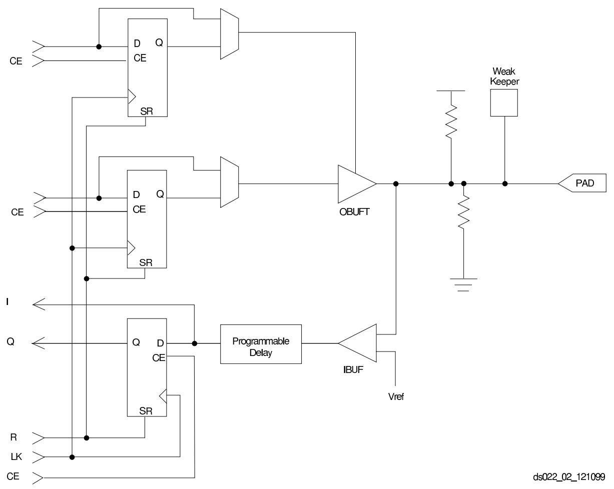


Figure 4.4: Input/Output Block (IOB) of a Virtex device [28]

Routing resources

To ensure best performance, the Virtex device offers several different routing resources to minimize signal delays. Between the CLBs, the most important ones are summarized:

- **General Routing Matrix (GRM):** provides adjacent to each CLB a switch matrix through which horizontal and vertical routing resources connect
- **Single-length lines:** provide routing from GRM signals to the four adjacent GRMs
- **Buffered Hex lines:** routing of GRM signals to GRMs which are six-blocks away. They can be accessed either at the endpoints or at the midpoints
- **Longlines:** these are bidirectional wires which distribute signals across the whole device
- **VersaRing:** provides a very fast interface between the periphery of the CLB array and the IOBs. Input/Output pins can therefore be swapped arbitrary.
- **Global routing:** provide four high-fanout clock and one reset signal distribution with minimal skew

Delay Locked Loop (DLL)

The Virtex FPGA provides four dedicated on-chip DLL circuits. Each DLL can drive up to two global clock routing networks within the device. By monitoring a sample of the DLL output clock, the DLL can compensate for the delay on the routing network and effectively eliminate the delay. In addition to providing zero delay with respect to a user source clock, the DLL can provide multiple phases of the source clock. Moreover, the DLL can also act as a clock doubler or it can divide the user clock by up to 16.

A DLL works by inserting delay between the input clock (CLKIN) and the feedback clock (CLKFB). The delay is performed by a clock delay element and adjusted by a control logic until the two rising clock edges align. Then the clocks are in phase (effectively they are 360 out of phase) and the DLL 'locks'. In that way, the DLL output clock (CLKOUT) compensates for the delay in the clock distribution network, effectively removing the delay between the source clock and its loads.

In order to guarantee that the system clock is operating correctly after the FPGA configuration, the DLL can delay the completion of the configuration process until after the DLL has 'locked'.

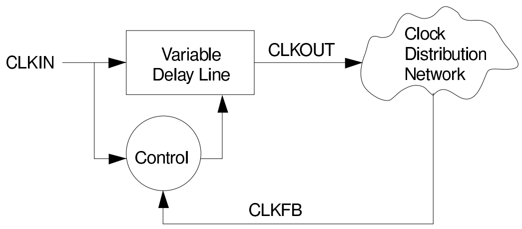


Figure 4.5: Delay Locked Loop (DLL) [28]

Configuration

Virtex devices are configured by loading configuration data into the internal configuration memory. After removing the supply voltage the device looses its configuration and has to be reloaded.

Configuration can be done in a serial mode or in parallel, where byte-wide data is written into the FPGA.

4.2 The Design Flow

The development of a digital design can be divided into three major steps (see Figure 4.6). At start-up the design behaviour has to be formulated in a hardware description language (HDL). The behaviour of the circuit is specified in an abstract fashion. In the second step, the so-called synthesis, the behaviour described in the HDL is translated into a so-called netlist. The netlist is a description of the circuit as a set of interconnection of logic gates (e.g. AND, OR, NAND, XOR). It has the same functionality as the design formulated in the HDL, but is just another representation. One has to ensure, that during design entry, only language constructs were used, which the synthesizer is able to translated. The generated netlist is finally mapped onto the configurable logic blocks of the FPGA. This is done by the so-called 'place and route' software of the FPGA-vendor. As a result, a 'bit-stream' is generated, holding the required information to configure the FPGA.

Within each step, it is advisable to perform a complete simulation of the design functionality. If an error occurs, either the design has to be changed or the step has to be repeated with different parameters.

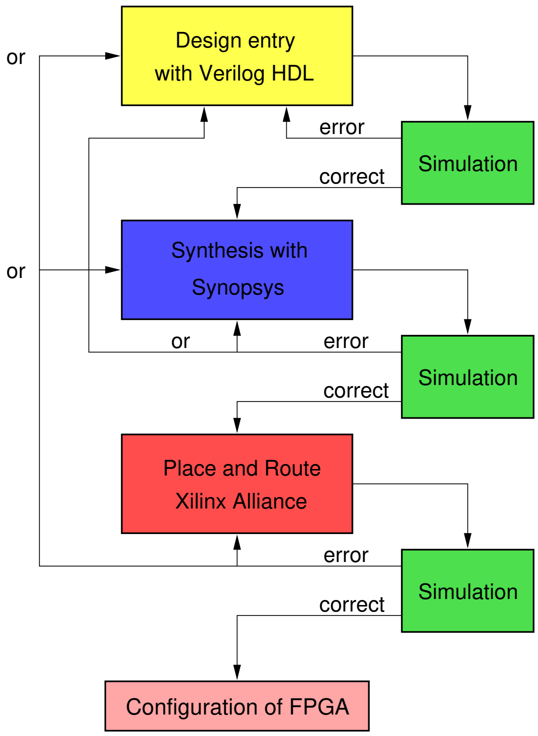


Figure 4.6: Design of FPFA, flow chart

4.2.1 Design Entry with Verilog HDL

Just ten years ago, integrated circuits were developed on 'the gate-level'. Logic gates (e.g. AND, OR, NAND, XOR) and flip-flops served as the basic building blocks to draw 'logic schematics' resembling circuits with a special functionality.

As electronic designs got larger and more complex, gate level descriptions became unmanageable and incomprehensible. This made it necessary to express designs in more abstract ways. This was the motivation to use hardware description languages for developing digital circuits. Just as in the 1970's, when high level programming languages replaced assembly languages, in the 1990's, hardware description languages (HDL) replaced gate level schematics.

Logic synthesis tools have to perform the gate-level translation. This transformation however, can still reach enormous complexity, depending on the abstraction level of the HDL. While the translation of a boolean operation is a fairly simple task, the translation of Finite-State Machines[1] (FSM) may confront the synthesis software with unresolvable problems. That's the reason why one has to check the synthesizablity of HDL constructs prior to the design entry.

[1] A Finite-State Machine is an abstract machine consisting of a set of states (including the initial state), a set of input/output events and a state transition function. The function takes the current state and an input event and returns the new set of output events and the next state. The state machine can also be viewed as a function which maps an ordered sequence of input events into a corresponding sequence of output events

At present there are two dominant hardware description languages, Verilog and VHDL. Both have become international standards for digital design.

The Verilog HDL

The Verilog HDL was developed in 1984 by Phil Moorby of Gateway Design Automation. It was developed to be used for hardware description and simulation of digitally based hardware systems. Almost every digital circuit can be described and simulated inclusive timing behaviour. Verilog HDL made its mark by allowing designers to represent their designs in structural descriptions as well as behavioral. A structural description explains the physical makeup of the circuit, detailing gates and the connections between them (this description is closer to the traditional way of drawing gate level schematics). A behavioral description describes what the circuits does in a more abstract way. The design entry, however may be a mix of the two.

The philosophy of the Verilog HDL is related to traditional procedural and sequential programming languages.

The language describes a digital system as a set of `modules`. Variables can be declared (`reg, wire`) and assigned (`assign, =`), arithmetic and combinatorial logic operations may be performed (`+,*,!`) and conditional statements are available (`if, case`). The main difference to programming languages is, that in contrast to software programs, where variables are always sequentially manipulated, signals in hardware are processed in parallel. Tasks are performed independent from others but with the freedom to exchange signals, as all tasks are performed with respect to a common clock signal. This parallelism has to be emulated in the HDL. Additional language elements are therefore available (`always@ Clk`) to make the execution of parallel events available.

4.2.2 Synthesis of the Design

The Synthesis tool translates and optimizes a HDL description of a circuit into a gate-level netlist build from library elements. The properties of the library elements consist of a set of logic gates and logic devices (e.g. DLL's, BlockRAM) which correspond to what can be implemented in the target device.

The functionality of this netlist is equivalent to the Verilog description of design, but is just another representation.

To achieve satisfying results, one has to consider the limits of the synthesis software and may give several guidelines to improve the performance. One example which certainly affects the speed of a given design is the way 'Finite-State machines' are implemented. The traditional methods used to generate state machine logic was to assign to every state one binary number, stored in flip-flops. These kind of implementation typically have a minimum number of flip-flops and wide combinatorial functions which result in highly-encoded states. However, as todays FPGAs have many flip-flops and comparably narrow function generators, highly-encoded state variables can result in inefficient implementation in terms of speed and density. A so-called 'one-hot' encoding reserves for each state of the state machine one flip-flop. This allows to create state machine implementations that are more efficient for FPGA architectures with a decreased width of combinatorial logic.

The synthesis of the FPGA designs, have been performed with *fpga_analyzer* and *FpgaCompiler2*, which are tools from Synopsys Inc. [32]. To check the functionality of the design, the netlist should be simulated. The behaviour has to be the same as with the Verilog simulation. This simulation already includes timing behaviour. This is possible, as the software adds averaged information of signal delays due to latency. The real timing behaviour can only be determined by the place and route software of the vendor.

4.2.3 Implementation of the Design on the FPGA

The translation of the netlist of the synthesis software into a configuration file for a FPGA device is done by a 'place and route' software. For Xilinx FPGAs this software tool is 'Xilinx Alliance2.1i' [33].

In the first step the software maps the library elements and ports of the netlist onto configurable logic blocks (CLB) and Input/Output ports (IOB) of the FPGA device. Then, the CLBs are placed and distributed on the CLB array. The software tries in an iterative approach, to most effectively interconnect the separated logic blocks with the available routing resources. The goal is to achieve minimal signal delays to meet the 'timing constraints' omitted by the user.

After completion, the timing behaviour can be analyzed and a maximum clock frequency is available. All routing delays are summarized in a file which is called 'Standard Delay Format' (SDF) file. A final simulation can therefore be performed including the SDF-file, allowing a very realistic scenario.

4.2.4 Simulation and Verification

The simulation is the only way to really visualize what is happening inside a digital device. It is therefore a very essential tool in the development of complex digital designs. To check the functionality at every step of the design flow, ensures the designer of the correctness of the internal logic.

However, it doesn't tell anything about the timing constraints to be met with the outside world. In order to test a developed FPGA-design, it is therefore most convenient to simulate the surrounding electronic of the hardware device, too.

For this reason so-called 'test benches' have to be written which simulate the behaviour of that part of the connected electronic, which communicates with the FPGA. In that way the simulation is not restricted to check only the response of the design to a sequence of applied test vectors at its inputs. In fact one can simulate the performance of real communication of the design with other hardware components like memories, processors or data links.

4.2.5 An Example Module: Generation of pseudo-random-numbers

As an example of a Verilog description of a digital circuit, a simple example module will be presented.

Because of its importance in Bit Error Rate Tests (BERT), a hardware generator of 32-bit pseudo-random-numbers with a asynchronous Reset, synchronous Load and Clock-Enable has been chosen.

Such a generator serves as a data source for Bit Error Rate Tests (BERT). In such tests, received signal patterns are compared to transmitted patterns in order to determine the rate at which the received signal bits differ from the transmitted signal bits. The bit error rate (BER) is characterized by a ratio of error bits to total bits, usually as a power of ten.

In the example, a received data stream of pseudo-random-numbers (e.g. transmitted via a data link) is compared to a reference pseudo-random-number generator. In this way, transmission errors can be determined by comparing the two pseudo-random-sequences (which should be equal) and increase an 'error-counter' when a mismatch is found.

This method has also been used for testing the PipelineBus long term reliability within the Pre-Processor Test System (see next Chapter).

A few words about pseudo-random-numbers

The need for a simple method of generating sequences with random properties arises in a wide variety of situations in electronics, digital computer work and cryptography. In a sense, however no finite sequence is ever truly random. The best that can be done is to single out certain properties as being associated with randomness, and to accept any sequence which has these properties as a random sequence (or a pseudo random sequence). The following properties should be associated with a random binary sequence.

- The number of Logic ONs (1) should be approximately equal the number of Logic OFFs (0)
- Runs of consecutive 1's or of consecutive 0's frequently occur, with short runs being more frequent than long runs. About 1/2 of the runs should have length 1, 1/4 should have length 2, 1/4 should have length 3 etc. Moreover, for each of these lengths, there are equally many runs of 1's and of 0's.
- Random sequences possess a special kind of auto-correlation function, which is a measure of the amount of similarities between the sequence and its phase shift. This should peak in the middle (no phase shift) and tapering off rapidly at the ends.

Possible generators of such sequences are shift registers with a linear feedback function (LFSR[2]). A shift register is an arrangement of $n-$tubes in a row, each tube is either 'ON' (1) or 'OFF' (0), which shifts the contents of each tube to the next tube, in time with a clock pulse. If no new signal is introduced into the first tube during this process, then all tubes will be 'OFF' at some stage.

To keep the shift register active, the states of certain of the $n-$tubes are fed back into the first tube. A modulo-2 sum is usually performed on the states before, on the next shift of the register, the result is transferred to the first tube (see Figure 4.7). It's not hard to show that the succession of states in a LFSR is periodic with a period $p \leq 2^n - 1$ (see Appendix B). But it turns out, that the case of a 'maximum length' LFSR with a period $p = 2^n - 1$, satisfies all three 'randomness properties' mentioned above, and there exists always at least one appropriate feedback function for every shift register length [29].

[2]LSFR: Linear Feedback Shift Register

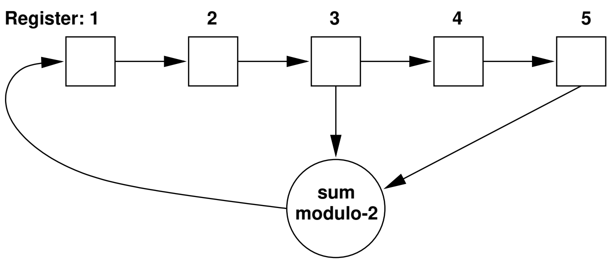


Figure 4.7: Shift register with feedback

The task is therefore to find a feedback function for which the sequence has a maximum length (A list of feedback functions for many 'maximum-length LFSR' can be found in the Appendix B).

The Verilog description of a Bit Error Test (BERT) on the basis of pseudo-random-numbers, realized with a LFSR, is shown in Figure 4.9. The period for a 32-bit LFSR is $2^{32} - 1 = 4294967295$. This means, that any bit combination only occurs again after completion of this period. A qualitative proof that the generated sequences are truly '(pseudo) random' can therefore be established by plotting a histogram of the generated data sequences. If the plotted data set is smaller than the length of one period, the histogram should be flat, with one entry for each bit combination.

A histogram for the first 950,000 data words of a 32-bit LFSR is plotted in figure 4.8 for a range of 1,000,000 possible combinations.

Figure 4.8: Histogram of the first 950,000 bit patterns of a 32-bit LFSR

The Verilog example starts with the declaration of the module and all its input/output ports. Multi-bit ports are declared using brackets, with the numbers defining the bus-width. The following lines declare the data types of the error flag, the error-counter, the shift-register and its parallel loadable input (to be able to change the initial value of the shift register).

```verilog
module BERT (ErrorCounter,
             Clk, Enable, Load, RandomIn, Reset, ShiftRegLoad);

  output [7:0] ErrorCounter;  // declaration of the outputs

  input Clk;                  // declaration of the inputs
  input Enable;
  input Load;
  input [31:0] RandomIn;
  input Reset;
  input [31:0] ShiftRegLoad;

  reg ErrorDetected;           // declaration of ErrorFlag
  reg [7:0] ErrorCounter;      // declaration of the error counter
  reg [31:0] RandomRef;         // declaration of the shift register

  wire [31:0] ShiftRegLoad;

  always @(posedge Clk or posedge Reset) // synchronous shift register
    if(Reset)                            // with Load and Enable strobe,
      RandomRef = 0;                     // asynchronous reset and
    else if (Load)                       // a linear feedback
      RandomRef <= ShiftRegLoad;
    else if (Enable)
      RandomRef <= {RandomRef[30:0], ~^{RandomRef[31], RandomRef[21],
                                        RandomRef[1], RandomRef[0]}};
    else
      RandomRef <= RandomRef;

  always @(RandomIn or RandomRef)
    if (RandomIn!=RandomRef)
      ErrorDetected=1;
    else
      ErrorDetected=0;

  always @(posedge Clk or posedge Reset)
    if(Reset)
      ErrorCounter <= 0;
    else if (Enable)
      begin
        if (ErrorDetected)
          ErrorCounter <= ErrorCounter + 1;
        else
          ErrorCounter <= ErrorCounter;
      end
    else
      ErrorCounter <= ErrorCounter;

endmodule
```

Figure 4.9: Verilog example of a Bit Error Test (BERT) module on the basis of pseudo-random-numbers

The first always block implements the shift register with linear feedback. A synchronous Load and Enable strobe is available as well as an asynchronous reset. The shift operation is realized by passing the last 31 bits of the previous word over to the first 31 bits of the new word. The remaining last bit is generated by a logic XNOR operation of bit 31, 21, 1 and 0 of the previous word. In that way, shift register sequences with random properties are achieved.

The next always block shows an example of a so-called combinatorial logic. Two digital values are compared and if the result is true a bit is set to Logic 1, otherwise to Logic 0.

The last always block implements a counter. The counter value is increased by one, if the ErrorDetected flag has a Logic 1 value (and the Enable flag is high), at each rising edge of the clock.

A part of gate-level schematics, generated by the synthesis tool *Synopsys* is shown in Figure 4.10. It shows 3 flip-flops out of the shift-register chain. This module has been

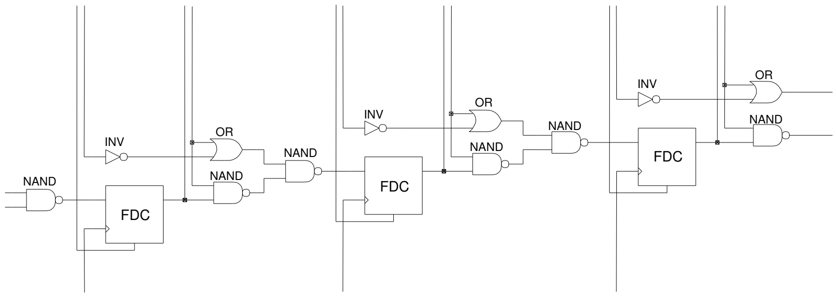


Figure 4.10: Schematic view of 3 out of the 32 flip-flops of the LFSR

simulated using *Verilog-XL* as the simulator [30] and the waveform viewer *SignalScan* [31], both tools are from Cadence Inc. Appropriate test vectors have been generated using a 'test bench' written in Verilog HDL. The simulation result is shown in Figure 4.11. Any

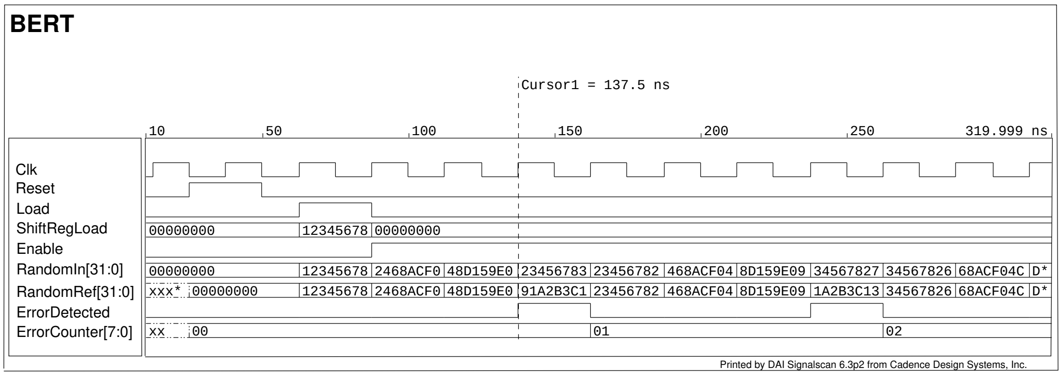


Figure 4.11: Simulation result of the Bit Error Test (BERT) module

time the reference data sequence 'RandomRef' is different to the 'RandomIn' data stream, the 'error counter' is increased by one. A cyclic error source with a period of four clock cycles has been generated in the test bench.

Chapter 5

PipelineBus Implementation

To test the functionality of the PipelineBus explained in Section 3.3.2, a modular test and development system was used.

To bring the PipelineBus into operation, two FPGA designs had to be written. The first was a Master-design, providing the logic to control the bus at 40 MHz and a so-called Spectator-design, which allows one to spy on bus data and monitor the bus activity. Timing between the bus nodes needed adjustment to ensure sufficient 'setup' time for the bus nodes, resulting in a very reliable configuration.

5.1 The Pre-Processor Test System

The philosophy of the Test System is to have a down-scaled version of the Pre-Processor system available, based on existing prototype components. In this way, it is possible to test and develop all prototype components of the Pre-Processor system. The flexibility of the system allows either separate tests of the components, or the operation in interconnection with other modules.

It is based on a general purpose motherboard, which provides commonly used resources. Special functionality is added by application specific daughtercards which can be plugged onto the motherboard. All daughtercards follow the Common Mezzanine Card (CMC) specifications [34], formulated by the IEEE[1].

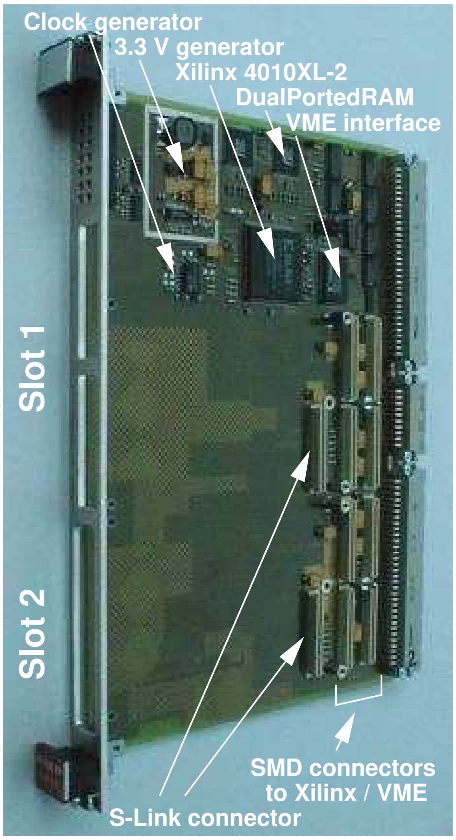
Eight daughtercards are available: A Master-CMC, which has the task to control the PipelineBus, a General-I/O-CMC, used to fanout common used signals, needed by all PipelineBus nodes. A RemAsic-CMC holds a prototype Read-Out Merger ASIC for data collection tasks on the board-level. The PPr-dMCM-CMC holds a Pre-Processor demonstrator MCM which performs most of the Pre-Processor task. An ECMC is used to evaluate design properties of a final Pre-Processor MCM. A pair of LVDS-CMC cards is available to test the serial high-speed links. Finally a Virtex-CMC and a S-Link-CMC are available for developments towards a final Read-Out Driver.

For every configuration with application specific daughtercards, the motherboard gets a different identity (see Section 5.1.2).

[1]IEEE: Institute of Electrical and Electronics Engineers

5.1.1 A general purpose Motherboard

The general purpose motherboard of the Test-System is a 6U VME[2]-module which provides two CMC[3] slots for inserting CMC-daughtercards. The CMC-interfaces to the daughtercards can be controlled in a flexible way by a Xilinx FPGA. For each CMC-card configuration an associated FPGA design is required.



An on-board 32 bit×8K Dual-Ported RAM is available to store 32 Kbytes of data. One side is connected to the Xilinx FPGA and one side to the VME-interface. It can therefore be read and written from both sides. Communication between both sides is triggered by an 'interrupt logic'. The interrupt is set on one side, when the other side writes to a special memory location and the opposite around.

The VME-interface, implemented inside an Xilinx CPLD[4], is used to configure and access the FPGA and to map the motherboard address space to VME.

A clock generator on-board can be selected and provides two independent clocks in a range from 0.4 - 120 MHz.

The motherboard requires a 5 V supply voltage from the backplane and generates a 3.3 V supply voltage on-board, which is available for the daughtercards via the SMD-connectors.

Eight SMD-connectors conform to the CMC-standard. They are partly linked to the motherboard FPGA, to the VME-interface and among each other. The remaining two connectors are compatible with the CERN Standard-Link (S-Link) specification and can therefore carry a S-Link daughtercard (see Chapter 6.3).

5.1.2 VME-Board Units

For each assembly of daughtercards on the motherboard (MB), the motherboard gets a different identity. As the MB has only two free slots available, combinations of 2 daughtercards at most are possible. These board configurations will be referred to as VME-board units. All available VME units are summarized in Table 5.1

The LVDS unit was used to test the high speed serial links from the Pre-Processor System to the Cluster and Jet/Energy Sum Processors of the Level-1 Calorimeter Trigger. The ECMC unit was used to evaluate layout properties of the final Pre-Processor MCM and to make performance measurements of the analog to digital converter (ADC). Both aren't discussed here but information can be found at [35].

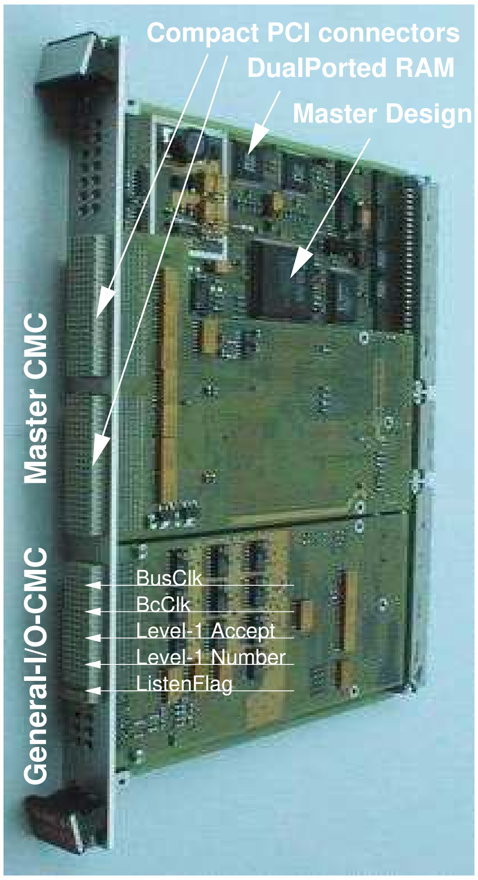
VME-board unit	Slot 1 (bus slot)	Slot 2
Master unit	Master-CMC	General-I/O-CMC
Pre-Processor unit	RemAsic-CMC	PPr-dMCM-CMC
Spectator unit	Master-CMC	—
Read-Out Driver unit	Virtex-CMC	S-Link-CMC
LVDS unit	LVDS-CMC (tx/rx)	—
ECMC unit	ECMC	—

Table 5.1: Test System VME-board units

The ROD Unit will be discussed in Section 6.4.

5.1.3 The PipelineBus Master Unit

The PipelineBus Master unit consists of the Master-CMC daughtercard plugged into the first slot of the Test System motherboard (MB) and the General-I/O-CMC, plugged into the second one.



The unit gets its functionality from the Master design implemented in the MB FPGA. The design can be accessed from VME via the Dual Ported RAM.

The PipelineBus ring is realized with point-to-point connections, formed by small printed circuit boards. They are plugged into the Compact PCI connectors available on each front panel of a bus node (in the final system, the PipelineBus will be accommodated on the backplane). Two Compact PCI connectors with 110 pins each, are used for all bus nodes. This allows every signal pin to be surrounded by multiple ground pins resulting in a very good shielding against disturbances from external sources. All outputs of the CMC card are buffered to provide enough current for a fast signal rise-time of the 5V TTL bus signals. The connection from the front-panel to the motherboard FPGA is performed by the MB SMD-connectors.

Control signals distributed to all PipelineBus nodes are generated by the bus master. The Master-CMC has a connection to the General-I/O-CMC which carries an Xilinx CPLD and fans-out up to 40 generated signals. To ensure safe signal transmission, the outputs are differential ECL signals. The control signals include the 40 MHz bunch-crossing clock, the

[2]VME: Versa Module Eurocard

[3]CMC: Common Mezzanine Cards

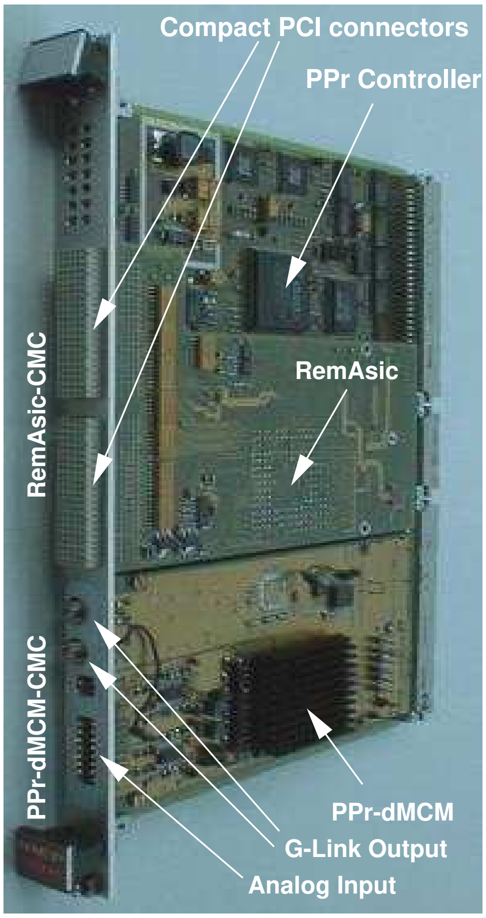
[4]CPLD: Complex Programmable Logic Device, these are devices similar to FPGA's but with less resources and defined timing

common bus clock, a Level-1 Accept signal and Level-1 numbers, which are the counted lower 4 bits of distributed Level-1 Accepts.

The motherboard FPGA can also be loaded with a PipelineBus Spectator design. In this case, the unit forms a PipelineBus Spectator unit which can be used to spy on the bus data and monitor bus activity between other nodes.

5.1.4 The Pre-Processor Unit

The Pre-Processor unit carries in the first slot a RemAsic-CMC and a PPr-dMCM-CMC in the second one. This unit has the functionality of a Pre-Processor Module. It performs all pre-processing and read-out tasks, but with a reduced number of four channels.



A FPGA design, the Pre-Processor Controller, implemented in the MB FPGA allows configuration of devices on the PPr-dMCM-CMC and the RemAsic-CMC which are not accessible via the PipelineBus interface of the RemAsic. An alternative configuration of the RemAsic is possible via a serial implementation of the PipelineBus interface. This is particularly useful for stand-alone tests of the Pre-Processor unit, without a PipelineBus Master unit present. For a detailed description of the Pre-Processor Controller, see [29].

The analog input of four trigger towers are received from the PPr-dMCM-CMC and processed by ASICs placed on the PPr-dMCM. The main tasks are performed by the Front-End ASIC (FeAsic) which is the predecessor of the Pre-Processor ASIC. After processing, the data is available at the high-speed serial links (G-Link[5]) which provide the succeeding trigger processors with data. This processing has been successfully carried out, and results can be found in [27].

The Front-End ASIC makes read-out data available via a serial interface connected to the RemAsic on the RemAsic-CMC. This link is established by the MB SMD connectors. The RemAsic performs compression on the data and makes them available to the PipelineBus interface (Compact PCI connectors). Unfortunately, the interface between the RemAsic-CMC and the FeAsic (on the PPr-dMCM) is not working properly. Results on performed investigations can be found in [29]. The PPr-dMCM-CMC could therefore not be used for planned tests. The Pre-Processor unit with or without a PPr-dMCM-CMC will be referred to as a Read-out Node from now on, being used as a data source for the PipelineBus.

[5]The G-Link technology was the prototype technology for the serial links, for the final system LVDS links are used

5.2 FPGA Design Specifications

Bringing the PipelineBus into operation required two FPGA designs to be written. One Master design, able to control the bus at 40MHz, and a Spectator design allowing to spy on bus data and monitor bus activity for debugging and testing reasons. A brief description of the designs is given here.

5.2.1 The PipelineBus Master

The task of the PipelineBus Master is to control the PipelineBus at the bus clock frequency of 40MHz. With every clock cycle one command has to be processed in order to support the full bandwidth. The design is accessible through the VME-interface. Commands are written to the interrupt register of the Dual Ported RAM from VME side. A received interrupt signal indicates the Master to read the register and to execute the command. Accordingly, this register is referred to as the command register. The command format is shown in Figure 5.1.

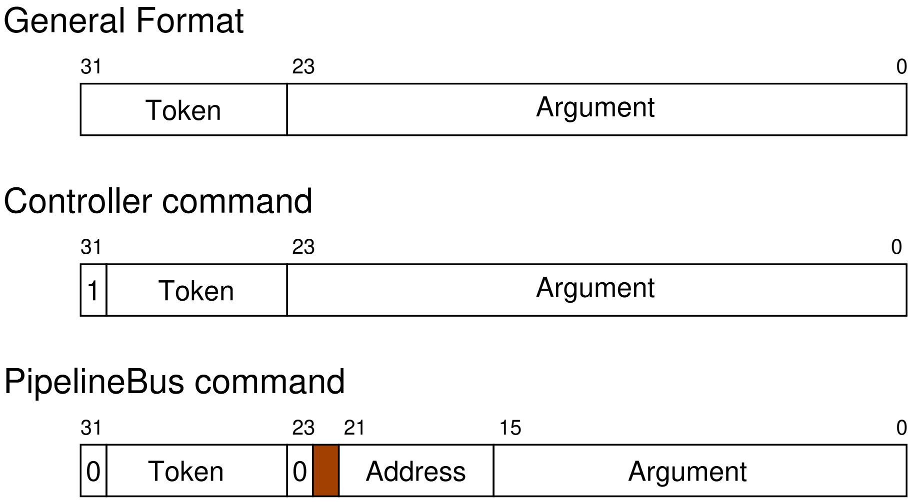


Figure 5.1: Format of the PipelineBus Master commands

Two types of commands exist, distinguished by the most significant bit of the command word. PipelineBus commands start with a zero value. Supported commands by the PipelineBus can be found in the appendix. These commands include configuration of Read-out Nodes, requesting status information and initiating read-out. The remaining class of commands are called Controller commands. The format is chosen in that way, that they are compatible with the Pre-Processor Controller design. Some Pre-Processor commands concerning configuration of some special Pre-Processor board devices (e.g. Phos4), are not available, as the PipelineBus Master does not have a physical connection to these devices. The command value, however has been left free to assure compatibility of common programs.

Commands written to the command register are executed immediately but they may also execute a program stored in the Dual Ported memory. The command `Execute` starts execution of a program at a given address by its argument. The program ends with a `Stop` command which stops execution. The program, may however contain further `Execute` commands so that the program pointer jumps to different memory locations. If the command register is written during program execution, the command from the register is executed and the previous one is aborted. A program may always be aborted with the `Stop` command.

The main function of the Master is to control the PipelineBus and the connected nodes. To do so, it transmits commands and data to the connected RemAsics at the 40 MHz bus clock via the PipelineBus connectors. As the Dual Ported Memory on the motherboard can either be read or written at this frequency, the response to the commands are monitored by the Spectator design to support the full bandwidth of the bus.

Command	Binary Value	Hex. Value	Description
Nop	1000 0000	80	No operation
Reset	1000 0011	83	Reset error counter
Execute	1000 0100	84	Execute RAM program
Stop	1000 0101	85	Stop program execution
Wait	1000 0110	86	Wait some clock cycles
CtrlReg	1000 0111	87	Set control register
PipeIdle	1000 1011	8B	Send some idle words
Lvl1Accept	1001 0000	90	Generate a Level-1 Accept
BeginOfData	0000 1111	0F	Mark begin of data block
PipelineBus[6]	0xxx xxxx	0-7F	Pipeline Bus commands

Table 5.2: List of PipelineBus Master command tokens

For configuration of Read-out nodes, the RemAsics first have to be switched into a state where they can receive input data. This is done with the PipelineBus command `StartInput`. The following command `BeginOfData` contains the address of the target node (the Master may transmit on the broadcast address, which is zero, then all nodes are addressed) and the target sub-device number to be configured in the argument field [36]. For the Master, the parsed command initiates a transmission of a user data block and has to contain the amount of data words to be sent. This is necessary for the design, to set the PipelineBus control lines correctly for data words and to change them again at the right time.

If no command is present in the command register to be executed and no program is running, either idle words are transmitted as default or `NoOperation` commands. This depends on a bit set in the control register. This is particularly useful, as in some cases it would be desirable to force the Read-out Nodes to no operation for a special amount of time. Two Controller commands are designated for this feature. The `PipeWait` command waits for the amount of clock cycles specified in the argument, before executing following commands. In the meanwhile default bus words, specified in the control register are sent.

[6] A list of supported PipelineBus commands and their meaning, can be found in appendix A

`PipeIdle` does the same, with the difference, to ignore the setting in the control register and to send Idle words in any case.

The control register has another bit, the 'EnableErrorCheck' which can be turned on. This activates the 'transmission quality check' protocol of the PipelineBus. In that mode, every Idle word, which is transmitted on the bus as default, is filled by the master with pseudo-random-numbers. These random numbers propagate through the pipeline and might be overwritten by Read-out Nodes. But the amount which hasn't been overwritten is received back by the master and compared to a reference data-stream. If a mismatch occurs, a dedicated error counter is increased. The current value of the error counter can always be read by an eight bit register, accessible independently from other bus activity. The `Reset` command with a selected ErrorCounterReset bit sets the error counter back to zero.

The amount of connected nodes may vary for different PipelineBus setups. For that reason, the Master design determines the number of connected nodes automatically at start-up. This is done by sending a synchronization word, encoded as a data word, at the global reset, which is activated after the design has been loaded into the FPGA. The fact that this word is encoded as a data word, ensures that no Read-out Node is processing this word. After reception of this sync word, the counted pipeline steps since transmission, is the number of connected nodes.

For testing reasons, a `Lvl1Accept` command has been implemented. This command triggers the General-I/O-CMC (plugged into the second slot of the Master unit) to distribute a 'Level-1 Accept' signal, with corresponding Level-1 number, to all connected nodes via a differential cable. In that way the General-I/O-CMC acts as a precursor of the Trigger Timing and Control system (TTCrx) within the Test-System (see Chapter 6).

5.2.2 The PipelineBus Spectator

The PipelineBus Spectator is used to spy on bus data and monitor bus activity between any nodes. The node is designed to accept the full bus bandwidth and to write any PipelineBus commands or data into the motherboard Dual Ported RAM. From there, the spectator data is accessible via the VME bus and can be copied to a single board computer.

Sequences of idle words or `NoOperation` commands between data-words or commands are counted to use the memory space in a more sufficient way. The summed values are written to the RAM as one word, just before a data or command word would be written. The memory address pointer can be asynchronously set back to zero.

5.3 PipelineBus Performance

The PipelineBus interconnects several Pre-Processor Modules to one Master node. It is used as a parallel link to send configuration data to connected Modules and to request read-out data from connected nodes. Tests have been performed which verified the functionality of the PipelineBus. These tests involved configuration operations of Read-out Nodes and requests of read-out data.

However, to check the overall performance and reliability of this data bus, long time tests were desirable. In an experiment like ATLAS, the read-out electronic has to operate reliably over years. Long time tests have therefore been performed on the basis of pseudo-random-data as data source.

The advantage of using pseudo-random-data is, that data sequences are reproducible. This means that transmitted data sequences can be received back and the receiver 'knows' already about the data it has to expect. If there is a mismatch on the received data sequence to the one transmitted, a transmission error has occurred.

The ring-like architecture of the PipelineBus is an ideal setup for such 'Bit Error Rate Tests' (BERT). The Bit Error Rate, which is the fraction of bit errors to the total amount of transmitted bits, is then a measure of the link performance.

The test setup is shown in Figure 5.2. Access to all VME boards was established from a terminal running a dedicated software (HDMC) which connects via ethernet to the TCP/IP interface of the VME crate controller.

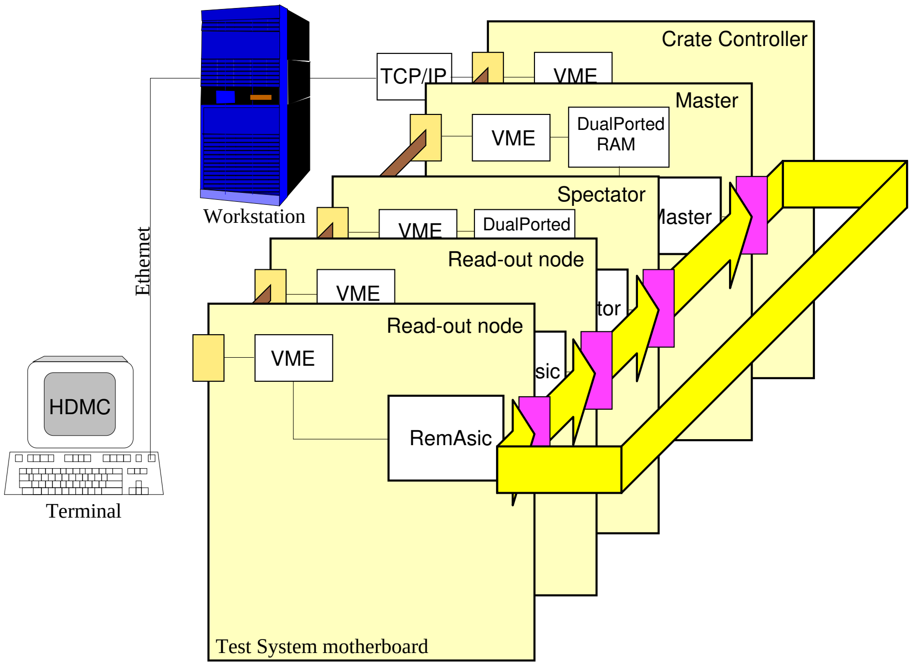


Figure 5.2: Test setup for Bit Error Tests with the PipelineBus

Two Read-Out Nodes were available for the tests, mounted with one RemAsic-CMC, each. The Spectator module served as a monitoring node to check from time to time if the PipelineBus operations are working properly.

In an indirect way, not only data words were checked for transmission errors but also command words and control bits. If a command word or a control bit were be transferred wrongly, the data sequences would be out of phase, with the consequence that every following word would cause an error.

To achieve a high data rate, an automated test was desirable. Moreover, data should be transmitted via the PipelineBus for a configuration process and a read-out operation. A repetitive program has therefore been written, executed by the PipelineBus Master, which sends blocks of random data successively to all connected nodes. If the nodes have received their (unique) pattern, a read-out operation is started (as described in Section 3.3.2). This operation then reads out the previously sent data blocks. Data is shifted through the Pipeline and is finally received back by the PipelineBus Master, which uses a reference random data generator, to count bit errors.

The used program, written in the HDMC assembler code, is shown in Figure 5.3, together with the meaning of the commands. For more information about HDMC (Hardware Diagnostics, Monitoring and Control Software), see [37].

After configuration, the RemAsics were reset and have been configured with continuous addresses (1,2). For the test, a modified Master design has been used, which identifies the command BeginOfData, as a request to send random data words of the amount, specified in the argument field.

```
PipelineBus # StartInput,1,0        // Start Input Operation for first node
PipelineBus # BeginOfData,1,0x302   // Send 48 data words to sub-device 2
$Inline 0x80                        // Set base address of sub-device 2
PipelineBus # EndOfData,1,0         // end of data block
Controller # PipeIdle, 1360         // wait the required processing time
... Same for second part of subdevice with base address 0x60
... Same for second node
PipelineBus # ReadSubDevice,1,2     // Start read-out mode for node 1
PipelineBus # ReadSubDevice,2,2     // Start read-out mode for node 2
Controller # PipeIdle, 210          // Time RemAsic needs to make data available
Controller # CtrlReg,0,1...         // Switch master to BERT mode ON
PipelineBus # BeginOfData,0,2       // Start PipelineBus read-out operation
Controller # PipeIdle, 120          // Wait until BERT is finished
Controller # CtrlReg,0,0...         // Switch master to BERT mode OFF
Controller # Execute,0x0            // Restart program
```

Figure 5.3: HDMC script of a repetitive program, initiating configuration and read-out of connected nodes

Initially, the measured Bit Error Rate (BER) was very high. Timing between the PipelineBus nodes was obviously a problem and needed adjustment.

5.3.1 Timing between Read-out Nodes

To enable fast response time to commands, initiated by the Master node, every PipelineBus node processes a PipelineBus word within one clock cycle. This is possible by employing one register and one multiplexer per node. The register latches the received data with every low to high transition of the bus clock. The multiplexer then allows to respond to commands and modify bus data, or simply leave the bus word unchanged. The node architecture is shown in Figure 5.4. The response to a bus word is therefore not determined by a fixed latency time, but given by the combinatorial logic implemented inside the RemAsic. This response time may therefore have arbitrary values.

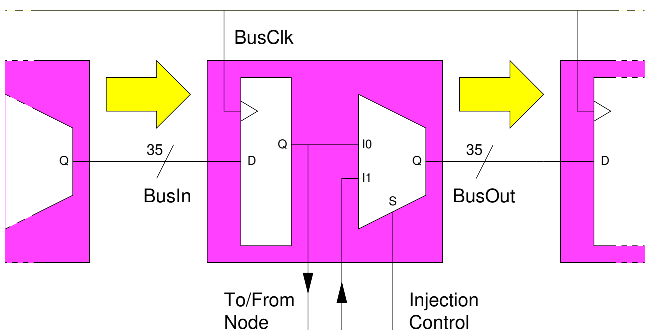


Figure 5.4: A PipelineBus node simply consists of a register and a multiplexer [28]

As an example a bit of the RemAsic output is plotted together with the bus clock (BusClk). This bit is a response to the Bit Error Test described above. This signal serves as input to the next connected node, which latches the data with the next positive edge of the bus clock. As this time is determined (by the bus clock), one has to make sure, that the data has reached a stable value before it is read by the next node. As the bus clock was measured at the bus connectors, one has to take into consideration, signal delays through electronic devices on the data and clock path.

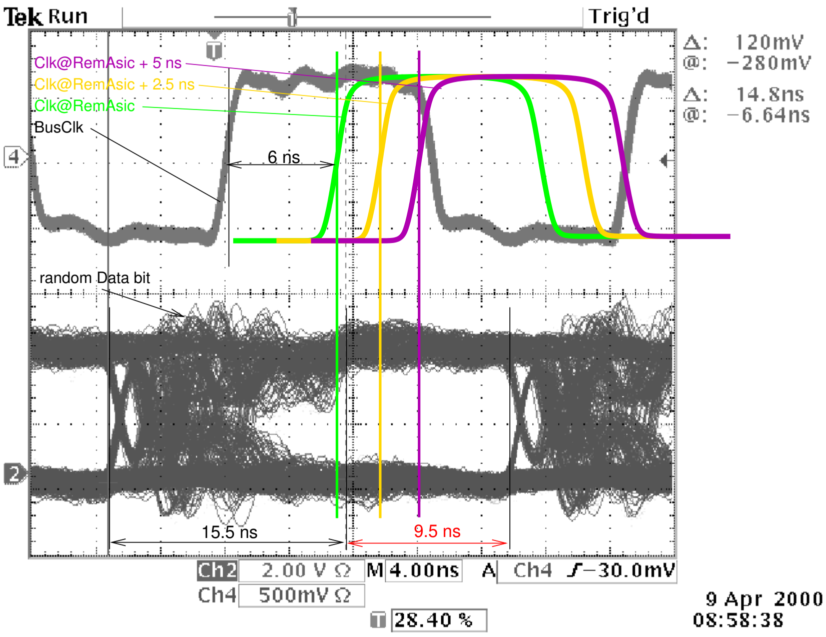


Figure 5.5: Response of a Read-out Node to a random test-pattern

It has been measured, that the bus clock path takes 6 ns more time than the data path, due to an additional multiplexer device on the board[7], a logic converter (ECL→

[7]The multiplexer is used to switch the clock of the RemAsic from the bus clock to a motherboard clock, which is used in stand-alone operation

TTL), buffers and routing delays on that path.

Three important things can be seen from Figure 5.5. First, the range of undetermined data, due to the combinatorial output of the RemAsic, is ≈15.5 ns. Second, in order to latch data, the bus clock of the following node has to be delayed by at least 2.5 ns to ensure enough setup time for latching stable data. And finally, the range of 'valid' data (i.e. data which is stable and doesn't change its value anymore) is ≈9.5 ns. The next positive edge of the bus clock may only migrate ≈7 ns to the left, to ensure reliable data transmission.

The maximum clock period which is manageable for successive connected RemAsics, is therefore ≈ 18 ns. That corresponds to a maximum bus clock frequency of ≤55 MHz, if RemAsics would be used for the PipelineBus interfaces of the final Pre-Processor Modules.

5.3.2 Bit Error Rate Tests

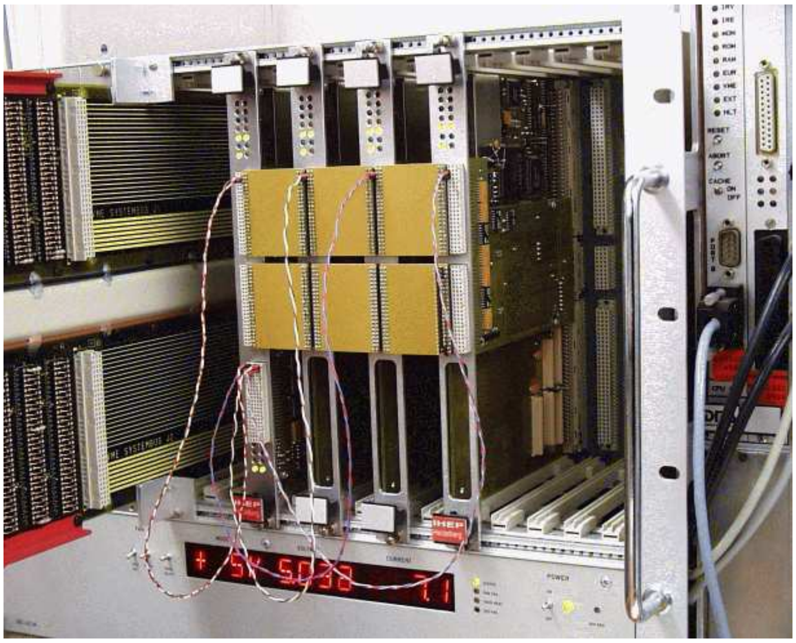


Figure 5.6: PipelineBus test-setup for Bit Error Rate Tests

With an adjusted timing, the PipelineBus showed a very reliable performance. The Bit Error Rate Test, described above, has been performed over a total time of 262 hours. During that time no bit error occurred. At a bus clock frequency of 40 MHz this corresponds to an upper limit of the Bit Error Rate (BER) of:

$$\text{BER} < 1.47 \times 10^{-14}$$

A picture of the test setup is shown in Figure 5.6. It shows a Master unit, two Read-out Nodes and a Spectator unit. The PipelineBus operations were checked from time to time by accessing the Spectator node via the VME interface. The PipelineBus ring was closed by a flat ribbon cable, which is not shown in this picture. The bus clock is distributed by the General-I/O-CMC via differential cables.

The main advantages of the PipelineBus can be summarized as follows:

First of all, the bus provides high bandwidth with a relatively low complexity. The bus is built from short parallel point-to-point links which are electrically and mechanical simple and minimize the capacitive load on the nodes. These short interconnections result in a very fast signal rise-time, compared to conventional bus implementations.

The PipelineBus also features a high flexibility (the number of connected nodes can easily be extended or reduced) and a fast response time to bus commands (every node processes data within one clock cycle).

Finally, the PipelineBus fits very well into the architecture of the Pre-Processor system and the overall ATLAS read-out. The bus can easily be accommodated on the backplane of the final Pre-Processor crates, and collected data is send via S-Link to the ATLAS DAQ system. The protocol of the S-Link is very similar to the PipelineBus protocol and the PipelineBus might be seen as a kind of an extension to the S-Link and the ATLAS data acquisition system.

Chapter 6

A Read-Out Driver Prototype-1

For developments towards a Read-Out Driver (ROD) prototype for the Pre-Processor System, a new CMC daughtercard has been designed and manufactured. The necessary logic has been implemented in a Virtex FPGA from Xilinx Inc. which has been placed on the daughtercard.

6.1 ROD Requirements

The main tasks of a Read-Out Driver (ROD) module in the final Pre-Processor System is to collect data from eight connected Pre-Processor modules (PPM) and to send formatted data to the ATLAS data acquisition system (DAQ). However, in order to realize a Read-Out Driver Module, several requirements have to be met [38]. A Read-Out Driver has to:

- collect data from all connected Pre-Processor Modules
- control the PipelineBus (bus master)
- operate at the Level-1 Accept rate
- format event data to ATLAS DAQ fragments
- buffer data and interface to the read-out link (S-Link)
- perform error detection on received data
- provide an event buffer to spy on read-out data
- interface to the ATLAS Trigger Timing and Control System (TTC)
- interface to the ATLAS Central Trigger Processor (CTP)

As it was decided initially not to interface to the TTC and the CTP system, the ROD prototype will be referred to as ROD prototype-1 from now on. However, for the most important signals, the interface to the TTC system is emulated using the General-I/O-CMC described in Section 5.1.2. More information on the TTC system can be found at

[46]. The interface to the CTP involves only one bit (ROD Busy), which will be considered with an accessible pin on the board. Planned further developments are briefly discussed in the Summary and Outlook. All other requirements can be met within the Test System environment. A schematic view of the new Virtex-CMC daughtercard plugged onto a Test System motherboard, together with a S-Link-CMC can be seen in Figure 6.1. The Figure

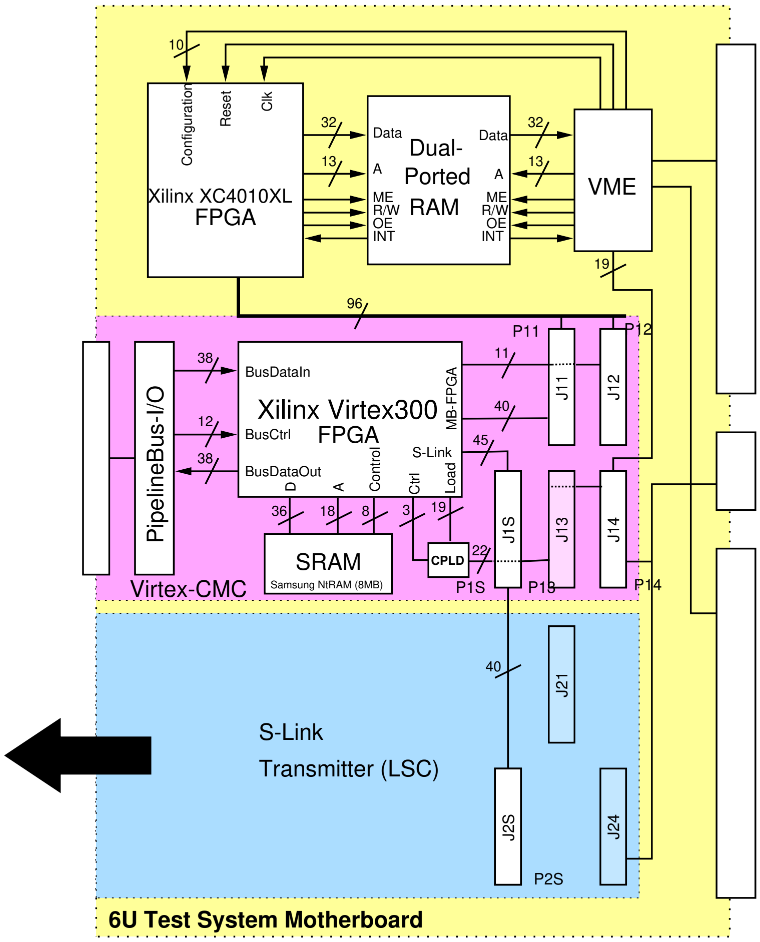


Figure 6.1: Schematic view of the new Virtex-CMC plugged onto a Test System motherboard together with a S-Link-CMC daughtercard

shows the Virtex FPGA on the new CMC which holds the necessary logic to perform the ROD prototype-1 tasks. The FPGA interfaces to the PipelineBus, to the S-Link transmitter card and the motherboard Dual Ported Ram. The SRAM on the Virtex-CMC will be used to perform the buffering of the data to be send to the DAQ system via S-Link.

The flexibility and modularity of the Test System, introduced in Chapter 5, allows several configurations of daughtercards which give the underlying motherboard a new identity. The design of the Virtex-CMC was therefore guided by the aim to use the daughtercard also for the development of the Read-Out Merger FPGA (RemFPGA). In that case, a PPr-MCM-CMC would be plugged onto the second slot of the motherboard to form a 'next generation' Pre-Processor unit to be tested in the Test System.

6.2 Design of the Virtex-CMC Daughtercard

The design of a printed circuited board (PCB) is basically done in three major steps.

The design is started with the schematic layout. This layout consists of the logical wiring of the signals and the power lines between the used components. For this, libraries of the used components have to be created. The libraries contain the layout information like the pin counts and the pin name. The used Virtex FPGA for example, has in total 352 pins, from which 260 are so-called user Input/Outputs and have been used to realize the necessary interfaces to the PipelineBus, the SRAM and the Dual Ported Ram. The tool *Concept* from Cadence Design Systems Inc. has been used for the schematic entry of the PCB design [39].

Once the logical wiring in the schematic layout is done, and all necessary components have been chosen, the physical routing of wires and the geographical placement of the devices has to be performed. Moreover the physical layout requires the true package dimensions and footprints of all used devices. In complex PCB designs, a so-called multilayer technique is used, where the necessary wire-routing, to connect all devices on the board, is done on several interconnected routing layers. For the Virtex-CMC eight routing layers were needed.

In the final step, the physical layout is translated into a manufacturing file, which holds all the information needed for the PCB manufacturer to build the board.

6.2.1 Design Considerations

In order to achieve optimal performance, some innovations have been put into the design process of the CMC-daughtercard.

- **Pin assignment:** First of all attention has been placed on the pin assignment of the interconnected devices on the PCB. The pin assignment of the PipelineBus interface and the SRAM interface of the Virtex FPGA have been chosen so that wires connecting the devices do not cross each other. As a result, all signal wires have more or less equivalent routing lengths, so that signal delays between neighboring data paths can not play a significant role.

  Moreover, the Virtex device has four dedicated so-called global clock input pins, which have to be connected to clock sources in order to use the internal DLLs and global routing resources most effectively.

- **Differential input signals:** A differential stripline of 100 Ohm impedance has been designed to match the impedance of the differential input cables (to avoid signal reflections on the input). These inputs are used for high frequency clock and control signals.

- **Powering of the Virtex device:** The Virtex device operates at a core voltage of 2.5 V. The Input/Output voltage has been made switchable between 3.3 V and 2.5

V to be able to support different I/O standards. As the Test System motherboard does not support a 2.5 voltage supply, an on-board 2.5V voltage regulator has been added.

- **Clock distribution to the SRAM:** The synchronous SRAM receives its clock from the Virtex device. A clock DLL is used for that task which is fed back into the Virtex again. This effectively de-skews the clock driving the SRAM and eliminates the routing delay through the IOB of the FPGA. This helps to meet the required setup time of the memory without any major problems.

6.2.2 Key Components

The manufactured and assembled daughtercard can be seen in Figure 6.2. The labeled components used on the Virtex-CMC are described here.

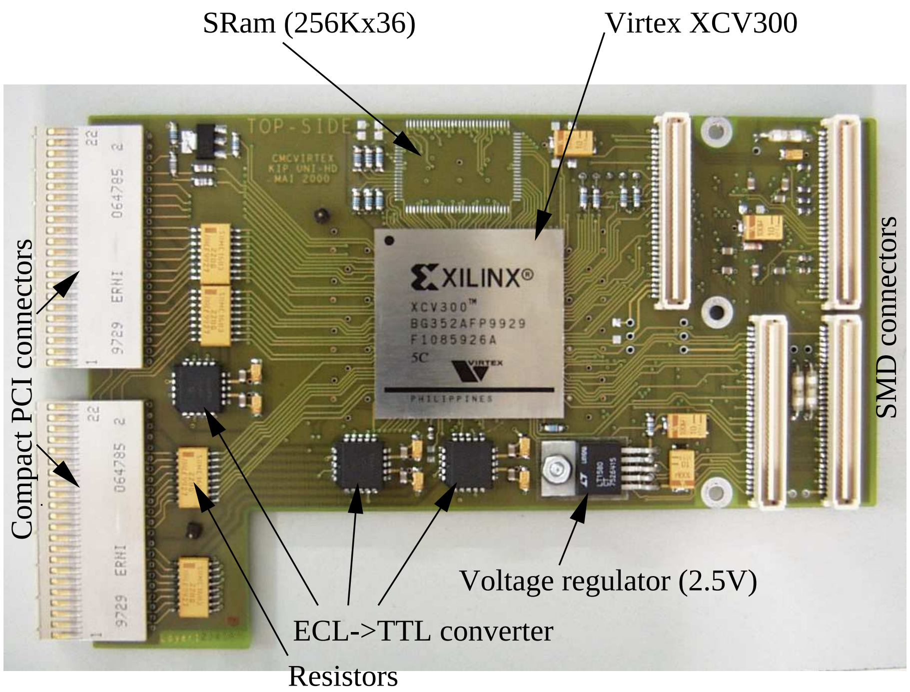


Figure 6.2: The Virtex-CMC daughtercard from the Top view

Virtex FPGA

The Virtex family from Xilinx Inc. has been already introduced in Chapter 4. Because of its large number of user Input/Output pins (IOBs) and its large amount of logic resources (CLBs) and BlockRAMs at a relatively small price, a Virtex XCV300 device in a 352 pin Ball Grid Array (BGA) package has been chosen. Both the Read-Out Driver prototype-1 and the RemFPGA will be implementable in that device. Figure 6.3 shows a photo of the BGA from a side view which was taken right after the BGA was assembled.

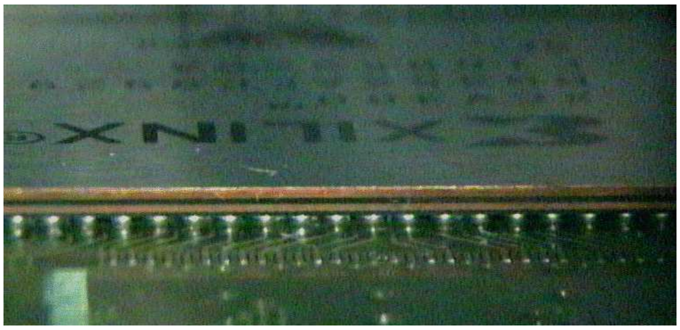


Figure 6.3: Virtex Ball Grid Array (BGA) from a side view

Samsung 256K×36 SRAM

The chosen SRAM from Samsung Electronics is a so-called synchronous non turn- around static RAM (NtRAM). Which means that a change from a read to a write operation can be performed on every logic low to high transition of the applied clock, without any latency restrictions. The maximum operating frequency is 120 MHz and the maximum amount of storable data is 256K×36 bit.

In order to use the RAM as a buffer memory, the device is clocked with double the frequency of the bus clock. By doing so, a read and write operation can be performed within one PipelineBus clock cycle. This permanently provides the S-Link interface with data while the buffer is not empty. The state diagram for the SRAM is shown in Figure 6.4. However, at the time of writing this thesis, the ordered SRAM has not been delivered

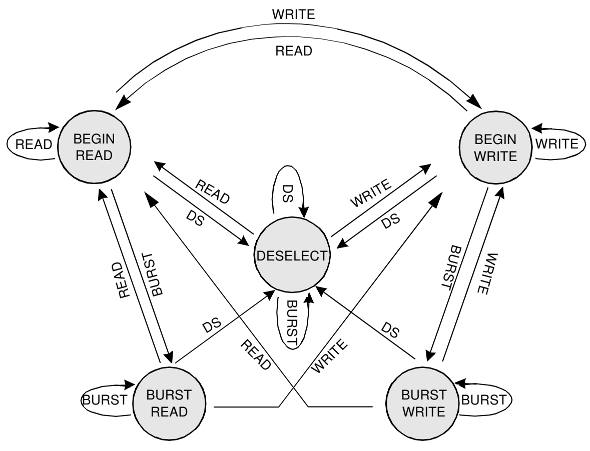


Figure 6.4: State diagram for the Samsung SRAM [43]

and could therefore not be used. As can be seen in Figure 6.2, the dedicated space on the board is left free.

The CPLD

An on-board CPLD was added to the daughtercard to allow a flexible configuration via the VME interface. From there, the Virtex FPGA is configured. For the loading sequence, the so-called SelectMAP mode has been chosen [28]. This mode allows the FPGA to be loaded with eight bit parallel data. The configuration bytes are shifted into device with every low to high transition of the configuration clock. This clock may have a frequency of up to 50 MHz. As the configuration files of Virtex devices are rather big (≈230 Kbyte) this allows a very short configuration time.

ECL → TTL converter

Three logic converter with four channels each, from Motorola have been employed. They allow 12 differential input signals (ECL[1]) to be transferred into unipolar signals (TTL[2]) on-board. The differential input channels are used to receive signals from the General-I/O-CMC, which provides all the PipelineBus nodes with commonly used signals like the bus clock and emulated TTC signals.

6.3 The S-Link

The S-Link, specified by CERN [40], is a data-link between (sub) detector specific electronics and standard read-out electronics (ROBs). In the LHC environment, there is need for several thousand of such data links.

The specification follows a new concept to provide standardization without limitations. The specification defines at both ends of the link a simple FIFO[3]-like user interface which remains independent of the physical link. This is particularly useful as link technologies are changing too rapidly and different environments in the LHC experiments are seeking different properties of the link, like radiation hardness, long or short distance coverage, etc.

In the case of the Pre-Processor, the link is used to transfer data from the Read-Out Driver Module to the Read-Out Buffers (ROBs) which are part of the ATLAS DAQ data flow.

The specification determines only the interface between the ROD and the Link Source Card (LSC) and the interface between the Link Destination Card (LDC) and the ROB. But it does not describe the physical link itself. This concept is shown in Figure 6.5

In addition to simple data movement, the S-Link includes some more features which depend on the actual S-Link implementation. S-Links are available commercially [40].

For the ROD prototype, we obtained a pair of S-Link cards which use a standard SCSI-II cable and LVDS driver/receiver chips from National Semiconductors as a physical layer. The main features of this implementation are:

[1]ECL: Emitter Coupled Logic

[2]TTL: Transistor Transistor Logic

[3]FIFO: First In, First Out. A queuing discipline in which entities in a queue leave the queue in the same order in which they arrive. The first data written to a pipeline or a data storage element is the first data that is read from it.

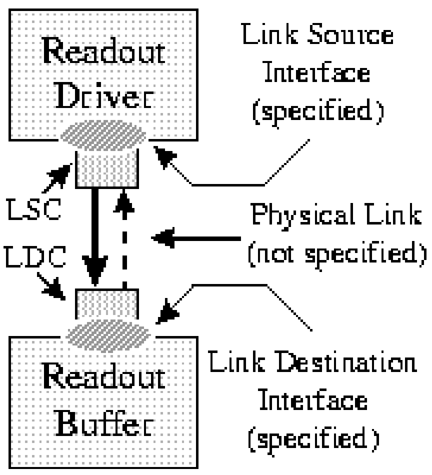


Figure 6.5: Concept of the S-Link specification [41]

- 32 bit of user data to be transferred on low to high transitions of the user clock
- 40 MHz max user clock
- 160 Mbyte/s max transfer rate over a length of 10 m
- word-by-word basis parity checking
- self-test function to be initiated by the LSC to check the Link functionality

A picture of the Link Source Card (LSC) is shown in Figure 6.6. The S-Link is basically controlled by two signals. A write enable signal (#WEN) enables data to be transferred

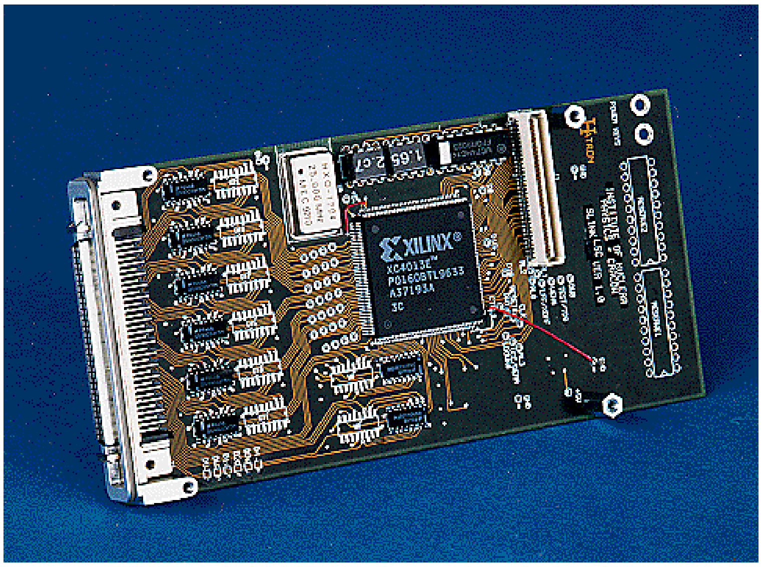


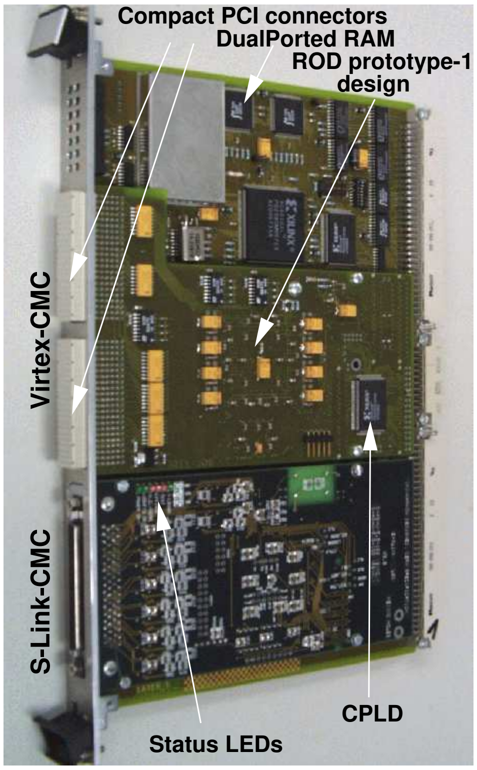
Figure 6.6: The parallel electrical S-Link implementation [44]

via the S-Link with every low to high transition of the applied clock. A control signal allows labeled words to be sent as control words. The Link destination card (LDC) may apply a so-called link full flag (#LFF) indicating that no more words should be sent via the link due to problems appearing on the receiver. A more detailed description can be found in the S-Link specification document, available at [40].

6.4 The Read-Out Driver Prototype-1 Unit

As mentioned already in Section 5.1.2 the configuration of the Test System motherboard, forming a Read-Out Driver prototype-1 unit (ROD-1), will briefly be described now.

The ROD-1 unit consists of a Virtex-CMC plugged onto the first slot of the Test System motherboard and a S-Link-CMC in the second slot. The configuration data is loaded via VME and a small interface, implemented in the CPLD of the Virtex CMC.



The unit gets all its functionality from the ROD prototype-1 design implemented in the Virtex FPGA on the Virtex-CMC. The design can be accessed from VME via the Dual Ported RAM in a similar fashion as the Master design. The only difference is that now the signals have to be routed through the motherboard Xilinx.

Corresponding to the PipelineBus topology introduced in Section 3.3.2, the ROD prototype-1 unit combines three logic PipelineBus nodes: the Master node; the S-Link node; and the Monitoring node;

The Master node has to control the PipelineBus, which is done via the PipelineBus interface. Physically this interface is realized by two compact PCI connectors which connect neighboring bus nodes through small printed circuit boards.

The S-Link node gets its functionality also through the Virtex FPGA and the electrical connection is established by SMD connectors on the motherboard. Several LEDs on the S-Link daughtercard indicate the current status of the link.

The Monitoring or spy node writes copies of read-out data to the motherboard Dual Ported RAM whenever there is no activity. This can be realized using FIFOs inside the Virtex, buffering data for the time the Dual Ported RAM interface is active. The data samples can then be accessed via the VME port of the RAM.

6.5 The Event Format in the ATLAS DAQ

The event format to be used in the ATLAS data acquisition (DAQ) system is not yet absolutely defined. However, a first prototype-1 specification is available at [42] and has been used to format read-out data collected by the ROD prototype-1 .

The event format defines the structure of the data as a set of fragments at various stages within the DAQ and allows elements of the data flow and processing tasks to access the data without resorting to use of other resources like databases etc. For a more detailed description see [42].

According to this specification, it is believed that data to be transferred from a ROD to a ROB should have both a Header and a Trailer as shown in Figure 6.7. The Header contains all the event formatting information and is followed by either the data elements or status elements. The relative order of these two blocks may be switched but has to be indicated in the Trailer which is the last part of a fragment.

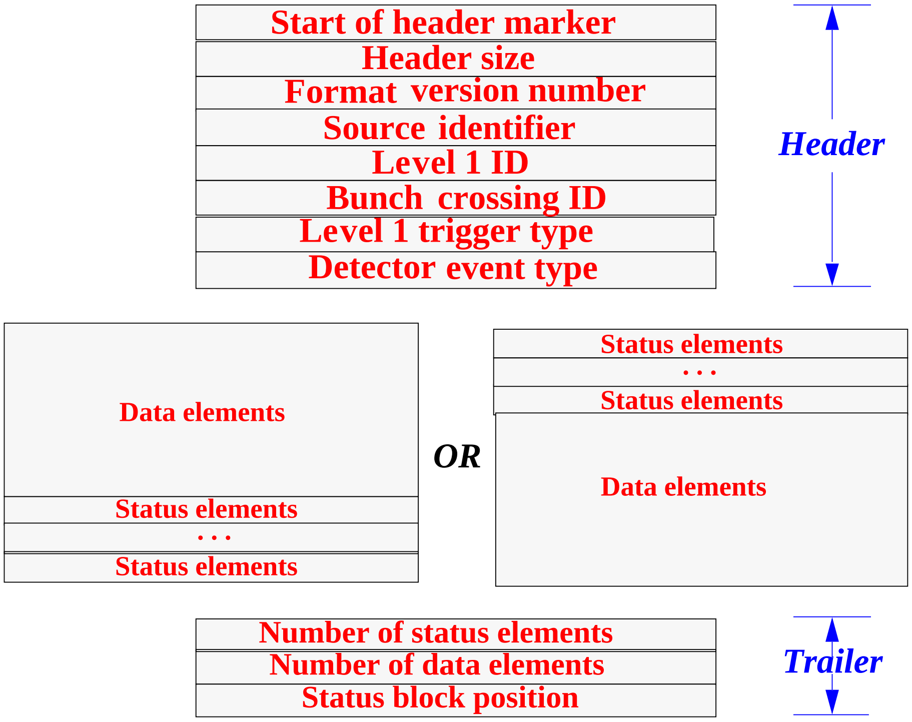


Figure 6.7: The ATLAS DAQ/EF prototype-1 event format [42]

- **Start of header marker:** hex EEEEEEEE
  This marker indicates the start of a ROD fragment header and is part of the header format.

- **Header Size:** hex 00000008
  This element indicates the total size of the Header, which is currently eight words.

- **Format version number:** hex 00000001
This element gives the format version of the ROD fragment which is currently 1. The existence of this element allows the ROD fragment to change independently of the format version of other sub-detectors.

- **Source identifier:**
This element identifies the source of the fragment. The bit field should allow a unique source identification across the whole ATLAS experiment. For the Pre-Processor System, this element will identify each of the 16 PipelineBus rings in a consistent way.

- **Level1 ID:**
This word contains the event identifier generated by the Level-1 Central Trigger Processor. It is a 24 bit value of successive counted Level-1 Accept signals.

- **Bunch crossing ID:**
This word contains the bunch crossing identifier generated by the Level-1 Central Trigger Processor. It is a 12 bit value of successive counted LHC bunch crossings.

- **Level1 Trigger Type:**
This element identifies the event type, transmitted by the Level-1 Central Trigger Processor and is coded as a 8 bit value.

- **Detector Specific Event Type:**
This element identifies an event, which may have been generated by a sub detector, independent of other sub-detectors and the trigger system. This might happen when signals for calibration are introduced into the system.

- **Data Elements:**
The data block contains read-out data from the connected Pre-Processor Modules. Every Module made data from 64 trigger tower channels available to the Pipeline-Bus. Eight Pre-Processor Modules are interconnected to one Read-Out Driver which performs the data formatting.

- **Status Elements:**
The Status Elements contain an error protocol, generated by the ROD-1. If no error occurred during one full read-out operation, this field is left free and is indicated by a value of 0 in the appropriate Trailer information.

- **Number of status elements:**
Holds the number of status elements following the data block.

- **Number of data elements:**
Holds the number of data elements contained in the data block

- **Status block position:**
This element solely defines the relative order of Data and Status elements. A value of zero for this element indicates that the status block precedes the data block and a value of one indicates that the status block follows the data block.

Each ROD fragment will be preceded and terminated by a single S-Link control word. These control words are referred to as `Begin of Fragment` (hex B0F00000) and `End of Fragment` (hex E0F00000). The S-Link uses the lowest two bits (which have to be initially set to zero), to perform a parity check on the transmitted data. The least significant bit indicates the status of the previous data fragment, the second bit indicates the status of the control word itself. A bit value of zero indicates no parity errors, a value of one means that a parity error occurred. All other data transferred via the S-Link interface are labeled as data words.

6.6 ROD Prototype-1 Features and Functionality

An FPGA design has been written to implement the necessary logic required for the ROD prototype-1. Figure 6.9 shows a logic block diagram of the implemented functionality. The framed part to the right corresponds to the functionality of the ROD prototype-1 Unit. The logic blocks on the darker layer are implemented in the Virtex device and correspond to modules in the Verilog code. The blocks on the brighter layer correspond to devices which have interfaces to the Virtex FPGA and indicate the interface logic, implemented in the XCV300 (S-Link, Dual Ported RAM, SRAM and the VME interface). The PipelineBus interconnects Read-out Nodes (RemAsic/RemFPGA) and is indicated by the big arrows outside the ROD part. The arrows in general symbolizes the data stream. The performed tasks will be briefly described.

6.6.1 Node Configuration

During configuration, the ROD prototype-1 acts as a data source to all connected Read-out Nodes. Data is downloaded to all RemFPGAs/RemAsics which make configuration data available to the whole Pre-Processor Module (PPM).

The ROD initiates a configuration process by injecting the PipelineBus command `StartInput` into the pipeline. The command propagates through the pipeline and requests each addressed node to switch to a state in which it is ready to receive data. The actual configuration data is sent after a `BeginOfData` which holds the address of nodes to be configured and the destination of the data. This is done by specifying a so-called sub-device as the target of configuration data. In that way, the RemFPGA/RemAsic, as the initial receiver of configuration data, can even initiate further forwarding of data to connected devices. For more information on the RemAsic and configuration processes, see [36].

6.6.2 Read-Out Tasks

During read-out operation, the ROD prototype-1 has to collect data from all connected Read-out Nodes and send formatted data to the ATLAS DAQ.

The ROD prototype-1 is switched into 'Auto-Read-Out mode' by applying the command `StartPBReadOut`. In that mode the Master first injects the command `StartReadout` into the pipeline and all Read-out Nodes switch to a state where they can make data available to the PipelineBus. Read-out operations are then only triggered by the Level-1 Accept signal. If a Level-1 Accept is received, the Read-out Nodes start to collect read-out data on the board-level via their serial interfaces. If all Read-out Nodes have a complete data block available for read-out, the node asserts a signal which is called 'ReadDataReady'. This signal is connected by a wired-AND logic to all Read-out Nodes and the ROD. The ROD prototype-1 parses the state of this signal. As soon the state changes, a read-out operation is started as described in Section 3.3.2. Data is shifted through the pipeline with every clock cycle and eventually received by the ROD-1.

ReadDataReady and timeout

However, if the ReadDataReady signal isn't asserted at all due to an unknown failure but the Read-out Nodes would actually have data available, a read-out operation would still be initiated. The ROD prototype-1 will only wait a specified amount of time for the ReadDataReady signal to become active. After this timeout, a read-out operation will be initiated on the basis of the PipelineBus command `GetEventCounter`. The response of the Read-out Nodes to that command is the amount of full events which they have available for read-out. If every node responses with a value greater than 1, a read-out operation is started by the ROD-1.

Parsing received data

The data received by the Read-Out Driver is checked for parity errors and node errors. Every RemFPGA/RemAsic on a PPM has added a header word to its data block. This header word contains the bunch-crossing number and the Level-1 Accept number corresponding to the current data block. The ROD prototype-1 can therefore check if these numbers differ from reference values. For every error or mismatch found during the parsing of received read-out data, an error word is written, which is stored in a an error FIFO. The used error protocol and its meaning is shown in Table 6.1. It is basically an extension to the existing node error protocol of the RemAsic. The used bit field is shown in Figure 6.8. The first eleven bits identify the error message, the following six bit identify the node producing the error, and the final eight bits contain the PipelineBus Token. This field is used in the case of the Level-1 and bunch number mismatches to hold the values of the last four bits of the received Level-1 and bunch number in contradiction to the expected value. In the case of a ROD parity error, the eight bit value holds the number of parity errors counted during the reception of one full event.

Figure 6.8: Bit field of the error protocol

Formatting of fragments and S-Link interface

The successively received Level-1 Accept signals and bunch crossing numbers are stored in FIFOs as well. All FIFO data is made available when event formatting is performed. The

	Error message (11 bit)	Brief description
0	Node Parity Error	Parity check performed by a Read-out Node failed
1	Writeout active	RemAsic is busy with internal data transmission
2	Readout active	While read-out is active configuration is not allowed
3	DataToFullBuffer	An input data block was oversized
4	BufferNotEmpty	Internal buffer memory needs to be empty
5	DataDirFromPipe	Data direction is wrong
6	DeviceBusy	Subdevice is busy with read-out
7	InputActive	RemAsic is in configuration mode
8	ROD Parity Error	An parity error was detected on received data
9	BC number mismatch	A BC number did not match with ROD BC number
10	Level-1 number mismatch	A Level-1 number did not match with ROD Level-1 ID

Table 6.1: Table of the ROD prototype-1 error protocol

preliminary ATLAS DAQ/EF event format has been described above. To guarantee the right order of read-out data, the raw event data is written to the SRAM buffer memory first. Header and Trailer is added afterwards, when all error checks have been performed. After a full fragment has been written to the SRAM buffer memory, the S-Link interface logic is enabled to send the data via the link to the Read-Out Buffers (ROB).

However, as mentioned above, the SRAM was not available by the time of writing this thesis. The ROD prototype-1 FPGA design needed therefore to be changed. Instead of the external SRAM, 5 Block SelectRAMs inside the Virtex have been applied to perform the tasks of the SRAM. These internal RAMs however have comparably little buffer space (256 words instead of 256,000 words of the SRAM). But fortunately, the S-Link was usable even with this little amount of buffer space.

ROD Busy

Next to the 'ReadDataReady' signal, there exists another Read-Out control signal which can be set by the Read-out Nodes. This signal is called 'Level1Throttle' and is asserted when the Read-out Node can't process further Level-1 Accepts.

In a similar way, the S-Link indicates that it can't process any more data by asserting a so-called 'LinkFullFlag' (#LFF). If this happens, the ROD-1 can buffer 256,000 more words in the dedicated buffer SRAM. If the buffer memory becomes full[4], a 'SRAM-FullFlag' is asserted. If either the 'SRAMFullFlag', or the 'Level-1Throttle' of any Read-out Node is asserted, the ROD-1 indicates with a ROD Busy signal that the Level-1 trigger has to stop the generation of further Level-1 Accept signals.

In the current implementation, the ROD Busy signal is just a pin on the board, but in the final system, this pin has to be connected to the Central Trigger Processor (CTP).

[4]This case is very improbable and should only happen if the S-Link is faulty

The PipelineBus Master

The PipelineBus Master functionality is basically the same as explained in Section 5.2.1. The design is also accessed via the command register of the Dual Ported Memory. The difference is that in auto-read-out mode, the PipelineBus is not accessible by the user. However, if a command is written to the command register, the Read-Out controller, is asserting a user interrupt and the command is executed between two succeeding read-out operations.

Access to Monitor or Spy Data

Monitor Data or Spy Data are accessible in a similar way to executing interrupt commands. By writing the command `ReadMonitorData` to the command register during read-out mode, an arbitrary number (specified in the command argument) of full event fragments are written to the motherboard Dual Ported Memory.

Read-Out Controller

Some Read-Out Driver specific commands have been added to the list of PipelineBus commands and Master controller commands, which are listed in Table 6.2. The command

Command	Binary Value	Hex. Value	Description
StartPBReadOut	1001 0000	90	Start auto-read-out
StopPBReadOut	1001 0001	91	Stop auto-read-out
ReadMonitorData	1001 0010	92	Read full fragment samples
SetMonitorAddr	1001 0011	93	Set the spy data address space
SetRDRTimeOut	1001 0100	94	Set ReadDataRdy timeout time
SetReadOutNodeNr	1001 0101	95	Set the Read-out Node number
RodCtrlReg	1001 0110	96	Set the ROD-1 control register
StartSLinkTest	1001 0111	97	Start S-Link test cycles
StartSLinkTest	1001 1000	98	Stop S-Link test cycles

Table 6.2: List of Read-Out Driver prototype-1 command tokens

`SetMonitorAddr` specifies the range of address space used for storing monitor data.

With the command `SetRDRTimeOut` the ReadDataReady timeout period can be changed individually. The value set in the argument of this command corresponds then to the amount of clock cycles, the ROD-1 will wait for the signal 'ReadDataReady' to be asserted by the Read-out Nodes.

As explained in Section 5.2.1, the PipelineBus Master determines the number of connected nodes on start-up. The Read-Out Driver takes this value as the number of interconnected Read-out Nodes. This is usually fine, however if a Spectator node is connected, one would like to set the Read-out Node number manually. This can be done with the command `SetReadOutNodeNr`.

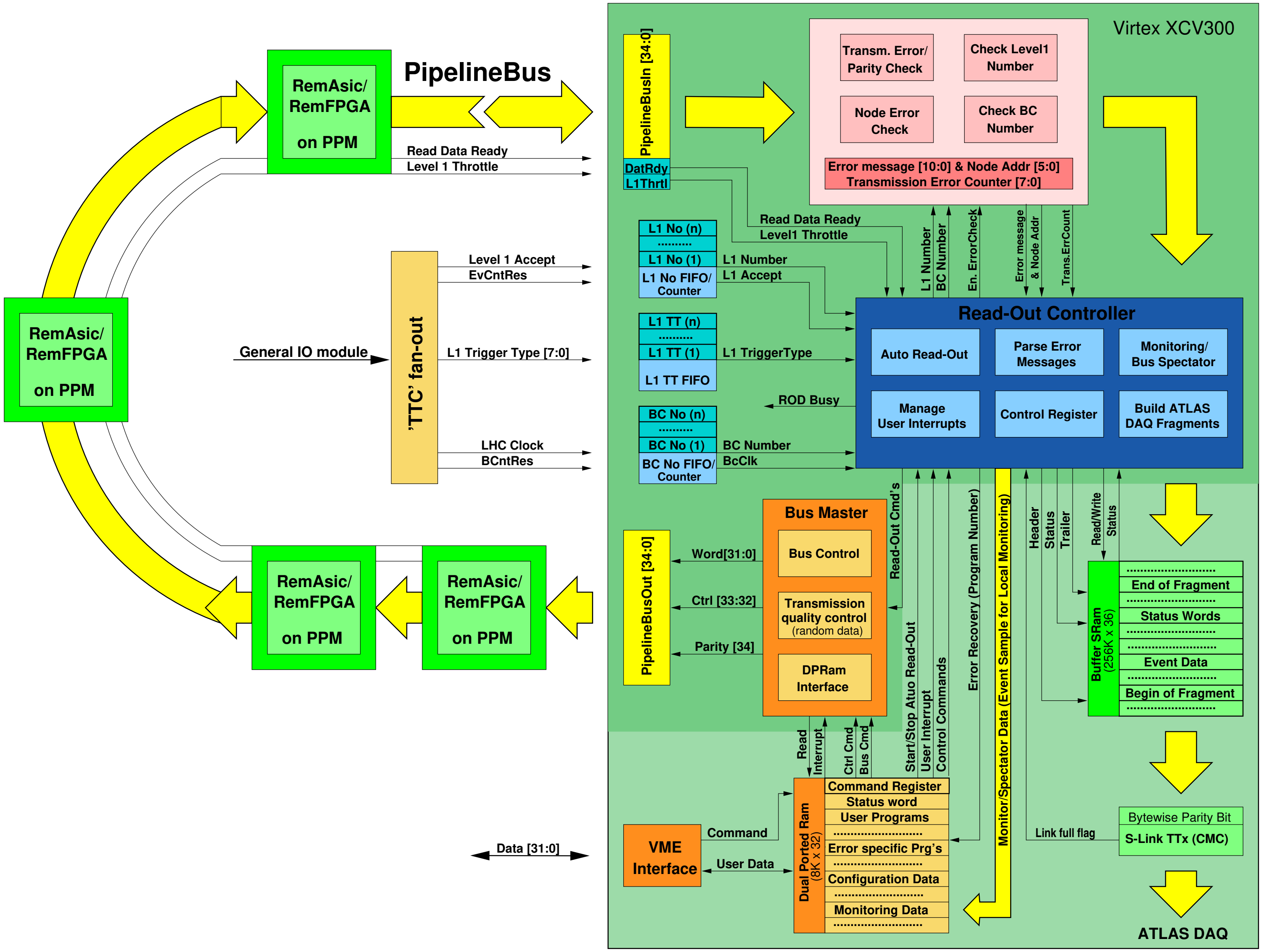


Figure 6.9: The Read-Out Driver prototype-1 Block Diagram

The command `RodCtrlReg` sets several control register bits of the design. The most important ones are the lowest four bits. With them it is possible to switch the error check on incoming data either ON or OFF (Parity check, Level-1 number check, BC number check, and Node error check).

6.6.3 Interface to TTC Signals

The Timing, Trigger and Control system (TTC) provides all signals necessary to synchronize the detectors at the LHC. All signals are distributed on optical fibers. A dedicated ASIC, the TTCrx will be used to decode the distributed signals [46].

Within the Test-System environment, the most important TTC signals are emulated by the General-I/O-CMC daughtercard and distributed on differential cables. The most important signals will now be described as will the action the ROD prototype-1 performs on reception of these signals.

LHC clock: *The LHC bunch-crossing clock of 40 MHz*
The ROD prototype-1 counts the number of successive bunch-crossings

Level-1 Accept: *The CTP has identified an interesting event*
The ROD prototype-1 initiates a read-out process to collect the data on which the trigger decision was based.

Level-1 ID: *The TTC distributes the current number of asserted Level-1 Accepts*
The ROD prototype-1 generates this number by counting the received Level-1 Accept signals. The RemAsic also receives a copy of the lower four bits of the Level-1 Number. In this way the ROD can check if the read-out data is synchronized to the 'TTC' system. The Level-1 ID is held in a FIFO until it is placed on the appropriate position of the ATLAS DAQ fragment.

Event Counter Reset (EvCntRes): *Request of setting the L1ID back to zero*
This signal resets the Level-1 ID.

Bunch Crossing ID: *For every Level-1 Accept, the corresponding bunch number is distributed (this number is reset after every complete revolution of the LHC beam)*
The ROD prototype-1 generates this number by counting the received bunch-crossings until a Level-1 Accept signal is received, then the counted value is transferred into a FIFO where it is held until it can be placed on the appropriate position in the DAQ fragment

Bunch Counter Reset (BCntRes): *Request of setting the BCID back to zero*
This signal resets the Bunch Crossing ID.

Trigger Type: *For every Level-1 Accept, the trigger type is distributed*
Not implemented on the ROD prototype-1, but will be added on the ROD prototype (see Outlook and Summary). For tests, a dummy value of hex 21 has been used.

6.7 Test Setup and First Measurements

To test the functionality of the ROD prototype-1 described in Section 6.6, a test setup has been established which is illustrated in Figure 6.10.

A Read-Out Driver unit interconnects two Read-out Nodes and one Spectator unit. In this test setup, the Spectator unit carries the General-I/O-CMC, which distributes the bus-clock and emulated TTC signals to all nodes. The Read-out Nodes have an on-board data source, which is labeled 'Front-End Data Source' as it emulates the read-out data coming from the Front-End ASIC, the predecessor of the Pre-Processor Asics. The ROD-1 collects the read-out data for every Level-1 Accept and sends the generated ATLAS DAQ fragment via the S-Link to a Data-Sink module where the received data is accessible via VME. This Data-Sink module emulates the Read-Out Buffers of the ATLAS DAQ system, being the receiver of read-out data.

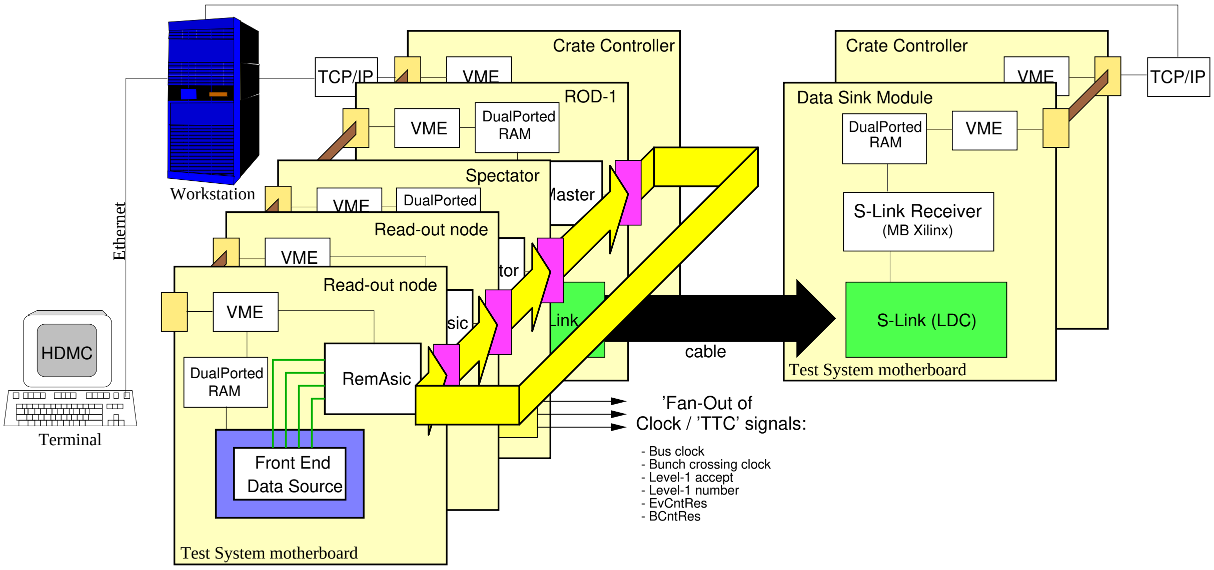


Figure 6.10: Schematic view of the Read-Out Driver prototype-1 test setup

Block SelectRAM

As mentioned above, by the time of writing this thesis the SRAM was not available for use. The ROD prototype-1 FPGA design needed therefore to be changed. Instead of the external SRAM, 5 Block SelectRAMs inside the Virtex have been applied to perform the tasks of the SRAM. These internal RAMs however had comparably little buffer space (256 words instead of 256,000 words of the SRAM). But fortunately, the S-Link interface was working even with this little amount of buffer space on the ROD prototype-1.

Front-End Data Source

The Front-End Data Source is a FPGA design, implemented in the Test System motherboard FPGA and connects to the four serial interfaces of the RemAsic. On a read-out request of the RemAsic, one eight-bit 'trigger-tower' signal is transferred for every channel. The RemAsic can be configured, that it requests on a Level-1 Accept signal, up to eight 8-bit data words for every channel.

Spectator & General I/O

The Spectator unit in this test setup carries the General I/O-CMC. The Spectator design mentioned in Section 5.2.2 has been upgraded so that it can control the distribution of emulated TTC signals. The distributed signals include the 40 MHz bus-clock, the 40 MHz LHC clock, a Level-1 Accept, a four-bit Level-1 number and the reset of the event-counter (EvCntRes) and bunch-crossing counter (BCntRes).

A Level-1 Accept can either be asserted in a 'single-shot mode', where a single Level-1 Accept signal is asserted on request of the user, or in a 'multi-event mode' where a periodic Level-1 Accept is distributed wihtin a range of 500 Hz and 120 kHz.

Data Sink Module

In the ATLAS data acquisition system, the destination for read-out data coming from the Read-Out Drivers via the S-Links, are the Read-Out Buffers. There, read-out data is stored in pipeline memories until the Level-2 Trigger has made its decision as to whether the event is of interest or not.

Within the ROD-1 test set-up, the reception of S-Link data is performed by the Data Sink Module. This module carries a Virtex-CMC in the first slot of the motherboard and a S-Link receiver card (LDC) on the second one. A FPGA design has been written to make the received data accessible to the VME interface of the motherboard. It is therefore possible to check and process the received data which has been transmitted via the S-Link from the ROD-1.

6.7.1 Single-Event Tests

Single-Event tests have been performed to verify the functionality of the Read-Out Driver prototype-1. By asserting a single Level-1 Accept signal, the whole chain of operations, presented in Section 6.6.2, have to be performed correctly, in order to receive a satisfying result at the S-Link Receiver. Figure 6.11 shows a HDMC screen-shot of the S-Link Receiver memory on the Data Sink Module. It shows the formatted read-out data as it was received by the S-Link receiver card (LDC). The meaning of the bit fields, according to the ATLAS Event Format described in Section 6.5 has been added. The memory of the Front-End Data Source was filled with a 'dummy event' consisting of constant 8-bit values of (hex) AB (To easily recognize the data elements). The RemAsic requested 8×4channels= 32 8-bit 'trigger towers' from the Front End Data source and made them available to the PipelineBus. There, the data words are mapped onto 32-bit PipelineBus words. In total, eight data words are read out from one Read-out Node (see Figure 6.11).

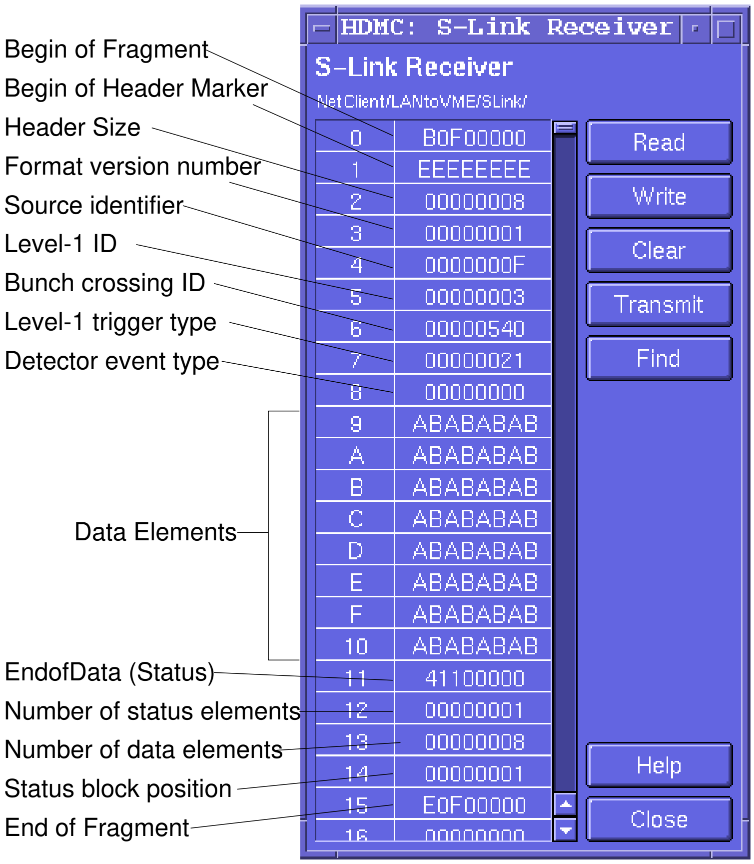


Figure 6.11: HDMC screen-shot of the S-Link Receiver memory

If the RemAsic performs compression on the read-out data (this is configurable, for more information, see [36]), the succeeding data blocks, coming from several Read-out Nodes, are not easily distinguishable. That's the reason why there is a status element (EndofData) after the block of data elements. This has been used in the tests to identify succeeding blocks of read-out data coming from different Read-out Nodes[5]. However, this status element will be adjusted to the RemFPGA implementation in the future.

6.7.2 Multi-Event Tests

To visualize the successive processes of data requests and data submissions, a periodic Level-1 Accept signal has been distributed by the General-I/O-CMC. It turned out, that for the RemAsic, the maximum Level-1 Accept rate is ≈ 78 kHz. The relative timing of the data chain is shown in Figure 6.12.

The Level-1 Accept signal triggers the read-out request of the RemAsic (last row). Data is then collected from the serial interfaces (which are fed by random data coming

[5] In this example one Read-Out Node with 4 channels has been read out, to be able to accommodate the whole ATLAS DAQ format on the screen-shot. Data from the second Read-out Node would just be attached to the first data block, with the status element in between.

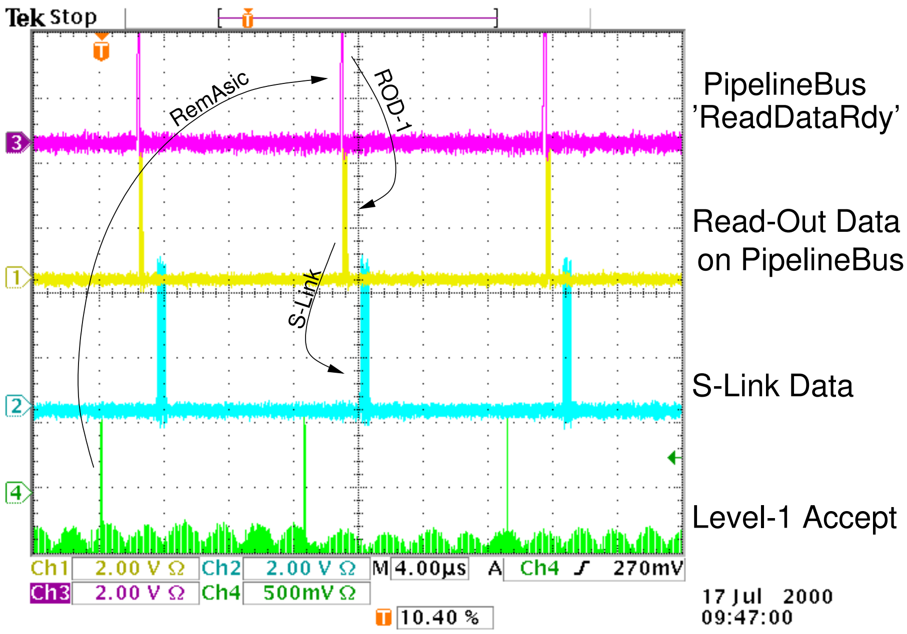


Figure 6.12: Relative timing on the read-out path for a 78 kHz Level-1 Accept rate

from the Front-End Data Source) and processed within the ASIC. Finally, the data is made available to the PipelineBus which is indicated by the 'ReadDataReady' signal (first row).

The Read-Out Driver parses this signal and initiates a read-out operation as soon the initial state changes. Data is then collected via the PipelineBus and eventually received by the Read-Out Driver (second row). The ROD then builds ATLAS DAQ fragments and buffers the data to be transmitted via the S-Link (third row).

In Figure 6.13 one event has been extracted from Figure 6.12. It can be seen, that the RemAsic makes data available to the PipelineBus about 2.2 μs after the Level-1 Accept has occurred. This delay is due to the serial interfaces of the RemAsic. On this interface one bit is transmitted per clock cycle (in this example, eight 8-bit values have been transferred per serial interface and Level-1 Accept, therefore the serial data transmission takes already $64 \times 25ns = 1600ns$) over the link.

However, this delay is not a problem, as the whole read-out is performed in a pipelined fashion. The 'ReadDataReady' is shown in the first row and shows a bit of an awkward shape. This is due to the 'open collector' circuit used to realize the 'wired-AND' logic described in Section 6.6.2. A smaller resistor at the 'open collector' circuit might be used to accelerate the signal rise time.

The delay between the PipelineBus data, shown in the second row, to the S-Link Data in the third row, is due to the buffering performed in the ROD-1. First the full event-data, collected from the PipelineBus, is written to the buffer memory. Then the ATLAS DAQ trailer and header is added. The data is transferred to the S-Link transmitter card (LSC) as soon as a full fragment has been written to the buffer memory.

A photo of the test-setup is shown in Figure 6.14. The PipelineBus and connected

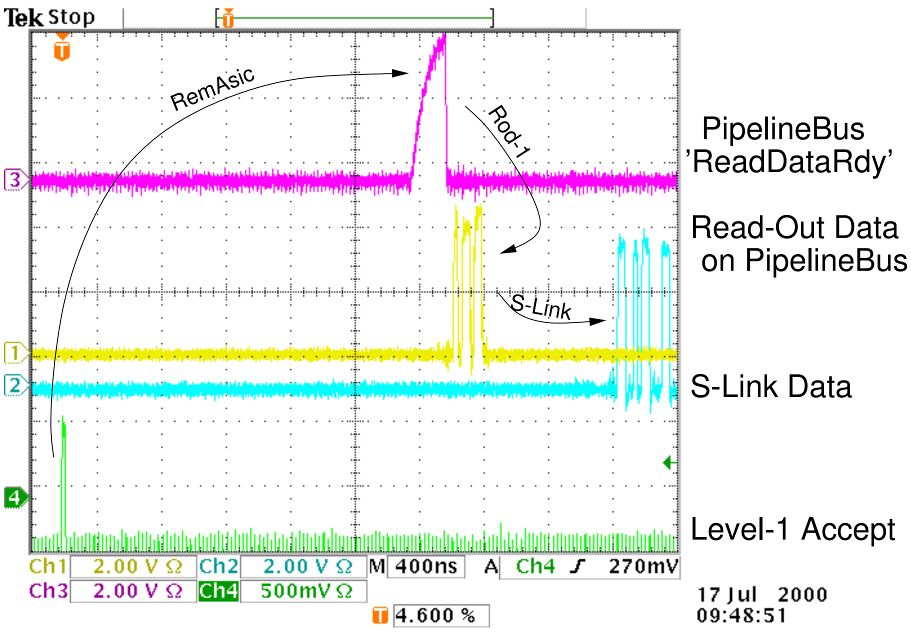


Figure 6.13: Relative timing on the read-out path

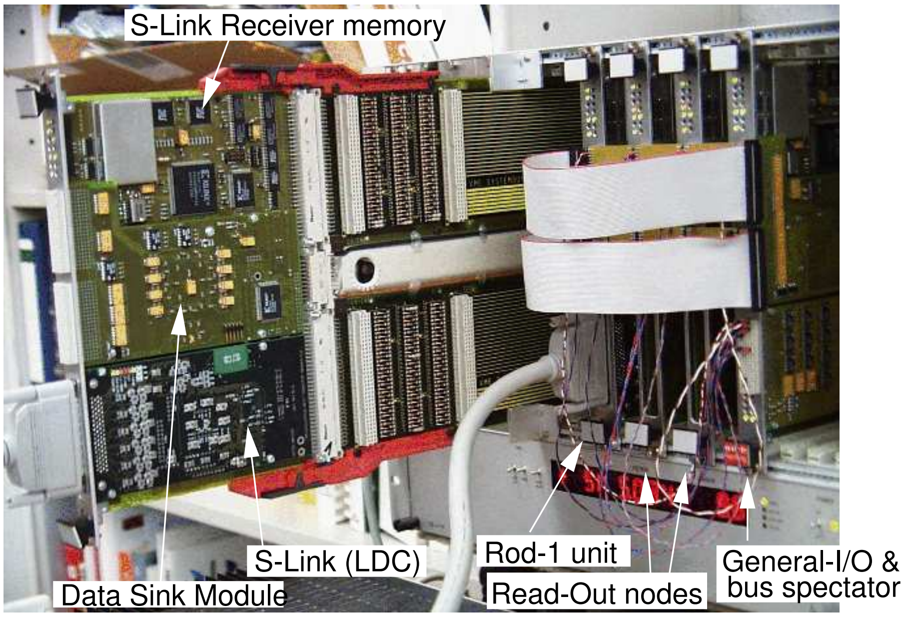


Figure 6.14: Picture of the test set-up for the Read-Out Driver prototype-1

nodes are hosted inside the crate. The PipelineBus ring is closed by a flat ribbon cable.

The Data Sink Module with the S-Link Receiver is carried by a VME extender-card (but might also be located in another crate). The connection between the S-Link source card and the S-Link destination card is performed by a standard SCSI-II cable.

Summary and Outlook

The PipelineBus of the ATLAS Level-1 Calorimeter Trigger Pre-Processor System has been successfully brought into operation within the modular Pre-Processor Test System.

After timing adjustments, the bus showed a very reliable performance. Bit Error Rate Tests (BER) on the basis of pseudo-random-numbers showed 262 hours of error free operation. This corresponds to an upper limit for the bit error rate of $BER < 1.47 \times 10^{-14}$.

The main advantage of the PipelineBus is that it provides high bandwidth with a relatively low complexity. It fits very well into the Pre-Processor and the overall ATLAS data acquisition system. The bus can easily be accommodated on the backplane of the final Pre-Processor crates and the similarities to the S-Link protocol make the PipelineBus look like an extension to the standard ATLAS data acquisition system (DAQ).

For developments towards a final Read-Out Driver (ROD) and a final Read-Out Merger FPGA (RemFPGA) for the Pre-Processor System, a CMC daughtercard has been designed, manufactured and successfully commissioned within the Test System. First hardware tests have been performed on a Read-Out Driver Prototype-1 implementation and the basic functionalities have been verified.

A final Read-Out Driver needs to interface to the ATLAS Trigger Timing and Control system (TTC) and the Central Trigger Processor (CTP). Moreover, it would be desirable to have on-board processing power (CPU, DSP or an additional FPGA) to perform data processing tasks on collected read-out data. This devices were not available within the Test-System.

It is planned to use the Virtex-CMC daughtercard on a motherboard equipped with these additional devices. This module can then be used to serve as a Read-Out Driver Prototype in a planned 'Slice Test' in summer 2001. Within this 'Slice Test', all Calorimeter Trigger Modules will be interconnected for the first time and it will be possible to test their performance in a system similar to the final Level-1 Calorimeter Trigger.

Appendix A

Token of PipelineBus Commands

Name	Value	Description of associated action
Nop	'h00	No operation
LoadNodeAddress	'h01	Configure node address
StartInput	'h02	Start of input operation
StartReadout	'h03	Start of readout operation
StopReadout	'h04	Stop the readout operation
ReadSubDevice	'h05	Start readout of subdevice
ReadoutReset	'h06	Reset the state machines
FlushMainBuffer	'h07	Read out complete buffer memory
ClearMainBuffer	'h08	Clear buffer memory
LoadPBRegister	'h09	Load PipelineBus register
GetStatus	'h0a	Read status register
GetPBRegister	'h0b	Read PipelineBus register
GetEventCounter	'h0c	Read event counter
GetSpyBus	'h0d	Read incoming SpyBus
SendSignal	'h0e	Send signals to device
BeginOfData	'h0f	Mark begin of data block
EndOfData	'h10	Mark end of data block
LoadGroupAddress	'h11	Configure group address
GetAddress	'h12	Read addresses of the node

Table A.1: Token of PipelineBus commands

Appendix B

Randomness Properties of Shift Register Sequences

A rather simple method of generating binary sequences with pseudo-random-properties is to use a shift register with a modulo-2 feedback.

Period of linear shift registers with feedback

The sequences so generated are periodic, with periods $p \leq 2^n - 1$, where n is the number of registers.

Proof: Each state of the shift register is completely determined by the previous state. When a state is therefore the same as the state at some earlier stage, the following states are the same, so that periodicity is established.

With $n-$registers in a shift register, each register holds either a logic value of 1 or 0. There are only 2^n possible states. Thus, a repetition occurs somewhere among the first $2^n + 1$ states, and there is periodicity, with $p \leq 2^n$.

Finally, if the state where all registers hold a logic 0's occurs, the subsequent states will also consist of all 0's, and the periodicity is $p = 1$. A long period can therefore not include this state and $p \leq 2^n - 1$.

A necessary condition for maximum length LFSR

Every shift register sequence a_m satisfies a linear recurrence relation $a_m = \sum_{i=0}^{n} c_i x^{m-i}$ where c_i is either 1 or 0 according whether the i^{th} register is or is not involved in the feedback circuit. By the method of generating functions, it can be shown:

$$\sum_{r=0}^{\infty} a_r x^r = \frac{g(x)}{f(x)}, f(x) = 1 - \sum_{i=1}^{n} c_i x^i \tag{B.1}$$

where the numerator is a polynomial of degree less than n, and the denominator is the *characteristic polynomial.* The necessary condition for the period of a_m to have a maximum length of $p = 2^n - 1$ is, that $f(x)$ be irreducible, which means unfactorable. The aim is therefore to search for characteristic polynomials which are irreducible. For a more detailed explanation, see [29].

n	XNOR from	n	XNOR from	n	XNOR from	n	XNOR from
3	3,2	45	45,44,42,41	87	87,74	129	129,124
4	4,3	46	46,45,26,25	88	88,87,17,16	130	130,127
5	5,3	47	47,42	89	89,51	131	131,130,84,83
6	6,5	48	48,47,21,20	90	90,89,72,71	132	132,103
7	7,6	49	49,40	91	91,90,8,7	133	133,132,82,81
8	8,6,5,4	50	50,49,24,23	92	92,91,80,79	134	134,77
9	9,5	51	51,50,36,35	93	93,91	135	135,124
10	10,7	52	52,49	94	94,73	136	136,135,11,10
11	11,9	53	53,52,38,37	95	95,84	137	137,116
12	12,6,4,1	54	54,53,18,17	96	96,94,49,47	138	138,137,131,130
13	13,4,3,1	55	55,31	97	97,91	139	139,136,134,131
14	14,5,3,1	56	56,55,35,34	98	98,87	140	140,111
15	15,14	57	57,50	99	99,97,54,52	141	141,140,110,109
16	16,15,13,4	58	58,39	100	100,63	142	142,121
17	17,14	59	59,58,38,37	101	101,100,95,94	143	143,142,123,122
18	18,11	60	60,59	102	102,101,36,35	144	144,143,75,74
19	19,6,2,1	61	61,60,46,45	103	103,94	145	145,93
20	20,17	62	62,61,6,5	104	104,103,94,93	146	146,145,87,86
21	21,19	63	63,62	105	105,89	147	147,146,110,109
22	22,21	64	64,63,61,60	106	106,91	148	148,121
23	23,18	65	65,47	107	107,105,44,42	149	149,148,40,39
24	24,23,22,17	66	66,65,57,56	108	108,77	150	150,97
25	25,22	67	67,66,58,57	109	109,108,103,102	151	151,148
26	26,6,2,1	68	68,59	110	110,109,98,97	152	152,151,87,86
27	27,5,2,1	69	69,67,42,40	111	111,101	153	153,152
28	28,25	70	70,69,55,54	112	112,110,69,67	154	154,152,27,25
29	29,27	71	71,65	113	113,104	155	155,154,124,123
30	30,6,4,1	72	72,66,25,19	114	114,113,33,32	156	156,155,41,40
31	31,28	73	73,48	115	115,114,101,100	157	157,156,131,130
32	32,22,2,1	74	74,73,59,58	116	116,115,46,45	158	158,157,132,131
33	33,20	75	75,74,65,64	117	117,115,99,97	159	159,128
34	34,27,2,1	76	76,75,41,40	118	118,85	160	160,159,142,141
35	35,33	77	77,76,47,46	119	119,111	161	161,143
36	36,25	78	78,77,59,58	120	120,113,9,2	162	162,161,75,74
37	37,5,4,3,2,1	79	79,70	121	121,103	163	163,162,104,103
38	38,6,5,1	80	80,79,43,42	122	122,121,63,62	164	164,163,151,150
39	39,35	81	81,77	123	123,121	165	165,164,135,134
40	40,38,21,19	82	82,79,47,44	124	124,87	166	166,165,128,127
41	41,38	83	83,82,38,37	125	125,124,18,17	167	167,161
42	42,41,20,19	84	84,71	126	126,125,90,89	168	168,166,153,151
43	43,42,38,37	85	85,84,58,57	127	127,126		
44	44,43,18,17	86	86,85,74,73	128	128,126,101,99		

Figure B.1: List of feedback functions for LFSRs from n=3 to n=168. The number of the registers which has to be combined to generate the XNOR feedback circuit are listed. The register counting starts with 1 [45]

Bibliography

[1] **G. Altarelli**, *Status of Precision Tests of the Standard Model*,
CERN-TH/96-265, (1996)

[2] **Pavel S. Aleksandrov**, *Einf/"uhrung in die Gruppentheorie*,
Dt.Verl.d.Wiss., Berlin, (1994)

[3] **E. Noether**, *Invariante Variationsprobleme*,
Nachr. d. Koñig. Gesellsch. d. Wiss. zu Göttingen, Math-phys. Klasse 235-257, (1918)

[4] **Christopher T. Hill, Leon M. Lederman**, *Teaching Symmetry in the Introductory Physics Curriculum*,
http://www.EmmyNoether.com

[5] **G. Arnison et al.**,
Phys. Lett. B122 103, (1983)

[6] **M. Banner et al.**,
Phys. Lett. B122 476, (1983)

[7] **P.W. Higgs**, *Broken Symmetries and the masses of Gauge Bosons*,
Phys. Rev. Lett. 13, 16, (1964)

[8] **W. B. Rolnick**, *The Fundamental Particles and Their Interactions*,
Addison-Wesley Pub. Company, (1994)

[9] **F. Abe et al., The CDF Collaboration**,
Phys. Rev. Lett. 74, 2626, (1995)

[10] **S. Abachi et al., The D0 Collaboration**,
Phys. Rev. Lett. 74, 2632, (1995)

[11] **ATLAS Education Committee** *ATLAS experiment, Introduction Tour*,
http://pdg.lbl.gov/atlas/physics13.html

[12] **European Committee for Future Accelerators**, *Large Hadron Collider Workshop, Proceedings, Volume I*,
CERN 90-10, (1990)

[13] **CERN AC**, *LHC dipole : standard cross-section*,
CERN Document Server, CERN-DI-9906025, (1999)
http://press.web.cern.ch/Press/Photos/

[14] **ATLAS General Site**, *ATLAS A Toroidal LHC ApparatuS*,
http://atlas.web.cern.ch/Atlas/

[15] **CERN AC**, *ALICE detector layout*,
CERN Document Server, CERN-DI-9803025, (1998)
http://press.web.cern.ch/Press/Photos/

[16] **CERN AC**, *CMS detector layout*,
CERN Document Server, CERN-DI-9803027, (1998)
http://press.web.cern.ch/Press/Photos/

[17] **CERN AC**, *LHC-B detector layout*,
CERN Document Server, CERN-DI-9803030, (1998)
http://press.web.cern.ch/Press/Photos/

[18] **Z. Kunszt et. al.**, *Higgs production at the LHC*
Z. Phys., C74:479, (1996)

[19] **R.M. Barnett et al.**, Physical Review D54 1, (1996)

[20] **The LEP Collaborations ALEPH, DELPHI, L3, OPAL, the LEP Electroweak Working Group and the SLD Heavy Flavour and Electroweak Groups**,
A Combination of Preliminary Electroweak Measurements and Constraints on the Standard Model,
European Organization for Nuclear Research, CERN-EP-2000-016, (2000)
http://delphiwww.cern.ch/ pubxx/www/delsec/papers/public/papers.html

[21] **ATLAS Collaboration**, *THE ATLAS DETECTOR AND PHYSICS PERFORMANCE Technical Design Report*, ATLASTDR-14, CERN/LHCC 99-14 May 1999
http://atlasinfo.cern.ch/Atlas/GROUPS/PHYSICS/TDR/access.html

[22] **ATLAS Homepage**, *Atlas Figures - Figures for LoI*,
http://atlasinfo.cern.ch/Atlas/ATLASFIGS/atlasfigures.html

[23] **ATLAS Inner Detector Community**, *Inner Detector, Technical Design Report*,
ATLAS TDR 4, CERN/LHCC/97-16, ISBN 92-9083-102-2, (1997)
http://atlasinfo.cern.ch/Atlas/GROUPS/INNER_DETECTOR/TDR/tdr.html

[24] **B. Aubert et al.**, *Performance of a Liquid Argon Electromagnetic Calorimeter with an "Accordion" Geometry*,
Nucl. Instr. and Meth. A309, 438-449, (1991)

[25] **ATLAS Muon Collaboration**, *ATLAS Muon Spectrometer Technical Design Report*,
CERN/LHCC 97-22 (1997)
http://atlasinfo.cern.ch/Atlas/GROUPS/MUON/TDR/Web/TDR.html

[26] **ATLAS Collaboration**, *Technical Proposal*,
CERN/LHCC 94-43 1994
http://atlas.web.cern.ch/Atlas/GROUPS/TP/tp.html

[27] **Ullrich Pfeiffer**, *A Compact Pre-Processor System for the ATLAS Level-1 Calorimeter Trigger*,
PHD thesis, University of Heidelberg, IHEP-99-11], (1999)
http://wwwasic.kip.uni-heidelberg.de/atlas/docs/publications.html

[28] **Cornelius Schumacher**, *The Readout Bus of the ATLAS Level-1 Calorimeter Trigger Pre-Processor*,
Fifth Workshop on Electronics for LHC Experiments, Snowmass, (1999)
http://wwwasic.kip.uni-heidelberg.de/atlas/docs/publications.html

[29] **Volker Schatz**, *Test of a Readout and Compression ASIC for the ATLAS Level-1 Calorimeter Trigger*,
Diploma thesis, University of Heidelberg, [HD-KIP 00-13], (2000)
http://wwwasic.kip.uni-heidelberg.de/atlas/docs/publications.html

[27] **Cornelius Schumacher**, *Der ATLAS Level-1 Trigger, Auslese des Frontends*,
Diploma thesis, University of Heidelberg, [IHEP-97-05], (1997)
http://wwwasic.kip.uni-heidelberg.de/atlas/docs/publications.html

[28] **XILINX Inc.**, *Virtex, Field Programmable Gate Arrays*,
Preliminary Product Specification (2000)
http://www.xilinx.com/products/virtex.htm

[29] **Solomon W. Golomb**, *Shift Register Sequences*,
2nd edition, Aegean Park Press, (1982)

[30] **Cadence Design Systems, Inc.**, *Verilog-XL User Guide*,
Openbook documentation,
http://www.cadence.com/

[31] **Design Acceleration Inc.**, *Advanced Design Analysis and verification tools for Verilog, VHDL and mixed Analog / Digital designs*,
http://www.designacc.com/

[32] **SYNOPSIS Inc.**, *FPGA Compiler II / FPGA Express Verilog HDL Reference Manual*,
Online documentation, Version 1999.05, (1999)
http://www.synopsis.com

[33] **XILINX Inc.**, *Alliance 2.1 Software Manuals Online*,
http://toolbox.xilinx.com/docsan/2_1i/

[34] **IEEE**, *Draft Standard for a Common Mezzanine Card Family: CMC*,
IEEE P1386/Draft 2.0, April 1995

[35] **Oliver Stelzer**, *Compact Digitizing and Serializing for the ATLAS Level-1 Calorimeter Trigger*,
Diploma thesis, University of Heidelberg, (2000)
http://wwwasic.kip.uni-heidelberg.de/atlas/docs/publications.html

[36] **A. Mass, B. Niemann, C. Schumacher** *DRAFT RemAsic User and Reference Manual*,
http://wwwasic.kip.uni-heidelberg.de/atlas/docs/chips.html

[37] **C. Schumacher et. al.** *Hardware Diagnostics, Monitoring and Control Software*,
http://wwwasic.kip.uni-heidelberg.de/atlas/projects/hdmc.html

[38] **ATLAS Trigger-DAQ Steering Group**, *Trigger & DAQ Interfaces with Front-End Systems: Requirement Document Version 2.0*,
ATLAS Internal Note, DAQ-NO-103 (1998)
http://atlas.web.cern.ch/Atlas/GROUPS/FRONTEND/essai.htm

[39] **Cadence Design Systems Inc.**, *Concept Schematic - User Guide*,
Openbook documentation
http://www.cadence.com/

[40] **O. Boyle, R. McLaren & E. van der Bij**, *The S-Link Interface Specification*
ECP Division, CERN (1997)
http://hsi.web.cern.ch/HSI/s-link/

[41] **R.A. McLaren et. al.** *An application of S-Link, a data link interface specification in the ATLAS readout System*,
Second International Data Acquisition Workshop (DAQ96) on Networked Data Acquisition systems, RCNP Osaka, JAPAN, (1996)
http://hsi.web.cern.ch/HSI/s-link/introduc/introduc.htm

[42] **C.Bee et. al.** *The event format in the ATLAS DAQ/EF prototype-1*,
http://atddoc.cern.ch/Atlas/Notes/050/Note050-1.html

[43] **SAMSUNG Electronics** *K7M803625M, 256K×36 Flow-Through NtRAM*,
http://www.intl.samsungsemi.com/Memory/SRAM/datasheets.htm

[44] **J. Olszowska, W. Iwanski** *E - SLink, the Parallel Electrical S-link Implementation*,
Institute of Nuclear Physics, Krakow, Poland
http://www.ifj.edu.pl/ iwanski/e-slink/slink.html

[45] **Xilinx Inc.**, *Linear Feedback Shift Register Tabs in Virtex Devices*,
XAPP210 v1.1 (2000)
http://www.xilinx.com/apps/virtexapp.htm#appnotes

[46] **B.G. Taylor**, *Timing, Trigger and Control (TTC) Systems for LHC Detectors*, http://ttc.web.cern.ch/TTC/intro.html

Acknwoledgement

It is a pleasure to be able to thank the many people who have supported me during the last year and who contributed to such a rewarding time at the Kirchhoff-Institut für Physik.

The spirit of the ATLAS group in Heidelberg was always guided by perfect team work, good humor and great enthusiasm.

All members of the Heidelberg ATLAS group, the ASIC Lab and the Electronic Workshop deserve recognition for their help and advice that were given on the numerous occasions when I needed assistance.

Figure 2.2: The Heidelberg ATLAS group. From left to right:
Oliver Stelzer, Prof. Eike-Erik Kluge, Volker Schatz, Ralf Achenbach,
myself, Dan Husmann, Klaus Schmitt, Cornelius Schumacher, Dr. Paul Hanke,
Dr. Ullrich Pfeiffer, Michael Keller, Prof. Karlheinz Meier.