Next Generation Data Processing for Future X-ray Observatories

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von

Henning Wende

aus Hann. Münden

Tübingen

2015

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	23.07.2015
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dott. Andrea Santangelo
2. Berichterstatter:	Prof. Dr. Klaus Werner

"Design is an iterative process.
The necessary number of iterations is
one more than the number you have done.
This is true at any point in time."

Akin's Law of Spacecraft Design #3

Abstract

During the last two decades space-based X-ray observatories have been used to study the most energetic sources in the Universe and to investigate the physics and characteristics of matter under extreme conditions. Detectors sensitive to X-ray photons have been significantly improved, their area has increased, their energy resolution has reached the Fano limit (120 eV @ 6 keV), and the detection time during which the detector is inactive (dead-time) has been reduced to nanoseconds.

These technological advancements have enabled X-ray observations with very high time resolution. Time resolved spectra in the range from 1 keV to 80 keV with good energy resolution open up a new field of X-ray astronomy since several types of X-ray sources, e.g. rotating neutron stars and black holes, show flux and spectral variations in time in the order of milliseconds and below. These variations can only be observed if an instrument is sensitive enough to detect single photons and if the read-out is fast enough to determine the energy of the photon and thus produce an event before the next photon arrives.

Concepts of instruments capable of producing spectra or single events with high time resolution are the High Time Resolution Spectrometer (HTRS) aboard the International X-Ray Observatory (IXO) and the Large Area Detector (LAD) aboard the Large Observatory for X-ray Timing (LOFT). While the detectors of these instruments can detect every single photon coming from an X-ray source, the subsequent data processing electronics have to deal with unprecedented bandwidths. Depending on the brightness of the source and the available telemetry bandwidth from the satellite to the ground station the need for data compression and even reduction arises.

In this thesis I present the work that was done in the context of the development of two instruments for X-ray observatories at the Institute for Astronomy and Astrophysics in Tübingen (IAAT). From 2010 to 2014 our institute participated in the development of the Data Processing Unit (DPU) for the HTRS aboard the International X-Ray Observatory and in the development of several components for the data handling chain of the LAD instrument aboard the Large Observatory for X-ray Timing, in particular the Panel-Back-End-Electronics (PBEE).

HTRS DPU
The first project was the definition and development of the HTRS DPU prototype. The configurable spectrum generation and detector handling that is part of the DPU was designed in VHDL and implemented into a Spartan 3 FPGA. To enable the use of the bzip2 compression algorithm in the DPU the VHDL model of the LEON3 microcontroller was integrated into the same FPGA and a custom designed system software was written in C using the RTEMS framework. The functionality of the DPU has been demonstrated and the required performance and efficiency of the bzip2 compression has been validated.

LAD PBEE
The second project was the design and development of the LAD PBEE prototype and the interface components used for communication between the Module-Back-End-Electronics (MBEE) and the PBEE. The custom interface and the data handling procedures of the PBEE are implemented in VHDL using a special VHDL design technique, the Two Process Method. For the PBEE prototype a PCB was designed, built, and equipped with a Virtex 4 and a Spartan 3 FPGA. The interface was implemented in the MBEE and the PBEE, and a communication test-bench with both prototypes additional MBEE simulators demonstrated the full functionality of the design as well as the required interface performance.

Structure of the Thesis
The first part of this thesis introduces X-ray astronomy (Chapter 1), explains the impact of the programmatic mission context on hardware design (Chapter 2), and gives an overview of the processes by which a proposed mission is selected for implementation.

The second part starting with Chapter 3 gives an overview of the basic hardware components that are used for data handling and processing, and illustrates the choices made for the HTRS and LAD equipment. Details on the software and programming techniques used during the development of the prototypes are given in Chapter 4.

The third part finally presents the two missions, IXO in Chapter 5, and LOFT in Chapter 6. The instruments are introduced with the main focus on the Data Processing Unit (IXO) and the Panel-Back-End-Electronics (LOFT), their functionality, and the results of the prototype validation. A summary of the work is given in Chapter 7 including an outlook on possible studies that might be conducted with the current or future prototypes.

Additionally the appendix of this work contains detailed step-by-step instructions on how to implement the LEON3 microcontroller design in an FPGA development board. This process was not very well documented before and it might be useful for other studies with the LEON3 to refer to this comprehensive guide on how to setup and configure a working system.

Deutsche Zusammenfassung

In den letzten zwei Jahrzehnten wurden eine Reihe von Röntgensatelliten entwickelt, gebaut und schließlich genutzt, um hochenergetische Röntgenquellen zu untersuchen, sowie die Eigenschaften und das Verhalten von Materie unter extremen Bedingungen zu studieren. Besondere Fortschritte wurden dabei auf dem Gebiet der Detektortechnik gemacht. Die für Röntgenstrahlen sensitive Fläche wurde vergrößert, die Energieauflösung der Detektoren konnte bis an das Fano-Limit herangebracht werden und die Zeit, die für eine Detektion notwendig ist und während der ein Teil des Detektors vorübergehen inaktiv bleibt (Totzeit), wurde auf wenige Nanosekunden reduziert.

Die technologische Entwicklung der Detektoren und der Ausleseelektronik ermöglicht heute Beobachtungen mit hoher Zeitauflösung und guter Energieauflösung von Röntgenstrahlen im Bereich von 1 keV bis 80 keV. So wurde ein neues Feld der Röntgenastronomie erschlossen, denn viele in diesem Licht sichtbare Quellen zeigen eine Variabilität im zeitlichen Verhalten ihrer Leuchtkraft und der spektralen Zusammensetzung, die im Bereich von Millisekunden und darunter liegt. Zu diesen Quellen gehören zum Beispiel rotierende Neutronensterne sowie rotierende Schwarze Löcher. Allerdings können die hier vorhandenen zeitlichen Variationen nur dann aufgelöst werden, wenn der Detektor über eine entsprechend kurze Totzeit verfügt und die auslesende und verarbeitende Elektronik die Detektordaten auch mit ausreichend hohem Durchsatz verarbeiten kann.

Im Besonderen sind zwei geplante Instrumente in der Lage, die benötigte Zeitauflösung bei guter Energieauflösung zu liefern. Das High Time Resolution Spectrometer (HTRS) des International X-ray Observatory (IXO) und der Large Area Detector (LAD) auf dem Röntgensatelliten Large Observatory for X-ray Timing (LOFT).

In der vorliegenden Arbeit stelle ich zwei funktionsfähige Prototypen vor, an deren Entwicklung das Institut für Astronomie und Astrophysik Tübingen (IAAT) in den Jahren von 2010 bis 2014 beteiligt war. Im Rahmen meiner Tätigkeit wurde die Data Procecssing Unit (DPU) des HTRS Instruments als Teil des IXO Observatoriums entwickelt und die Panel-Back-End-Electronic (PBEE) des Large Area Detectors der LOFT Mission gebaut. Beide Prototypen wurden sowohl in Simulationen wie auch in einem Laboraufbau auf ihre korrekte Funktionalität und vorgesehene Leistungsfähigkeit hin untersucht. In beiden Fällen kann diese Arbeit über die positiven Ergebnisse berichten.

HTRS DPU
Zu den Aufgaben der DPU gehören die Steuerung des Detektors und das Erzeugen von konfigurierbaren Spektren aus Einzelereignissen zur Datenreduktion. Diese Funktionalität wurde in VHDL entwickelt und in einem Spartan 3 FPGA implementiert. Eine besonders wichtige Aufgabe der DPU ist die Datenkompression mit dem bzip2 Algorithmus. Die Kompression wurde in der Programmiersprache C implementiert und unter Verwendung des VHDL-Modells des LEON3 Mikroprozessors ebenfalls in den FPGA integriert. Die Funktionalität der DPU und die erwartete Leistung der bzip2 Kompression konnten in einem Laboraufbau demonstriert werden.

LAD PBEE
Die Arbeit am zweiten Projekt beinhaltete die Definition und die Entwicklung eines PBEE Prototypen und einer neuartigen Schnittstelle für die Kommunikation zwischen Module-Back-End-Electronic (MBEE) und PBEE. Sowohl die Schnittstelle wie auch die gesamte Logik der PBEE wurden in VHDL entwickelt. Dabei wurde eine spezielle Methode der VHDL Entwicklung, die Two Process Method, verwendet. Ein Platinenlayout des Prototyps wurde erstellt, gefertigt und bestückt. Dabei kamen Trägermodule mit einem Virtex 4 und einem Spartan 3 FPGA zum Einsatz. In einem Laboraufbau aus einem MBEE Prototypen, dem PBEE Prototypen und zwei weiteren MBEE Simulatoren, die auf der PBEE Platine untergebracht sind, konnte die geforderte Datenübertragungsrate der Schnittstelle und Funktionalität der PBEE gezeigt werden.

Struktur der vorliegenden Arbeit
Nach einer Einführung in die Röntgenastronomie in Kapitel 1 wird der Einfluss des programmatischen Kontextes einer Mission auf die Entwicklung der Hardware (Kapitel 2) erläutert.

Kapitel 3 präsentiert einen Überblick der zur Verfügung stehenden Hardware für die Datenverarbeitung und begründet auch die getroffene Auswahl für die entwickelten Prototypen. Im Anschluss daran gibt Kapitel 4 einen Einblick in die verwendete Software und erklärt insbesondere die Vorteile der verwendeten VHDL Entwicklungstechnik in Abschnitt 4.2.

Im dritten Teil der Arbeit werden die Missionen IXO (Kapitel 5) und LOFT (Kapitel 6) vorgestellt und die genannten Instrumente detailliert beschrieben. Diese Kapitel gehen auch detailliert auf die entwickelten Komponenten (HTRS DPU und LAD PBEE) ein und stellen die Entwicklung der Prototypen bzw. die Ergebnisse der Labortests vor. Abschließend fasst Kapitel 7 die Arbeiten und Ergebnisse noch einmal zusammen und gibt auch einen Ausblick auf Weiterentwicklungen der Prototypen und mögliche weiterführende Studien.

Im Anhang der Arbeit finden sich eine Reihe detaillierter Anleitungen wie der LEON3 Mikrocontroller auf einem FPGA-Entwicklungsboard in Betrieb genommen wird.

Contents

Part I

X-ray Astronomy

Chapter 1

X-ray Astronomy

1.1 A Brief History of X-ray Astronomy

The reason why X-ray astronomy was not available to astronomers during most of the history of mankind is, that the Earth's atmosphere is nontransparent to X-rays. This is because X-ray photons carry more energy than visible photons and are able to ionize oxygen and nitrogen atoms in the atmosphere. The process is called photo-electric absorption since the X-ray photon is absorbed by the atom, while a free electron is produced.

When the first X-ray detectors were invented, it immediately became clear that the Sun was an intense source of X-rays. The question arose if there were more X-ray sources in our Galaxy. But even in a starry night with clear skies, no X-rays reach the ground. X-ray detectors therefore had to be lifted into the upper atmosphere. The first rocket flight that carried an X-ray detector and successfully detected a cosmic source of X-ray emission was launched in 1962 by a group at American Science and Engineering (AS&E). The very bright source found by Riccardo Giacconi, Herbert Gursky, Frank R. Paolini, and Bruno B. Rossi was named Scorpius X-1 (Giacconi et al. 1962).

The launch of the first dedicated X-ray satellite Uhuru (Chapter 1.2.2) in 1970 marked the beginning of an era of space-based X-ray astronomy. With the detection of more extrasolar sources, e.g. Cen X-3 and Her X-1 by Uhuru (Krishnaswamy 1996), scientists had to think of mechanisms in which stars, or their remainders, could produce X-rays. When pulsations in the X-ray flux in the order of seconds were discovered, it became clear that rotating neutron stars had to be the natural sources of the emission. Variations in the pulse period due to Doppler shifting indicated that the neutron star was moving in an orbit. The hypothesis that the neutron star was part of a binary system with a normal star companion proved correct. Since then X-ray binaries are studied by scientists all over the world because they feature conditions (density, temperature) that are unique amongst a large spectrum of astronomical objects (Ramadevi 2007).

The binary nature of the star system allows astronomers to determine the masses of the neutron star and its companion. For some of these systems, the mass of the X-ray emitting object was found to be hundreds or thousands of solar masses, but the mass of a neutron star is limited to less than three solar masses, otherwise the star would collapse. The findings thus supported the idea of the existence of black holes.

The pulsations found in some X-ray emissions led to the understanding of the accretion mechanism. In a binary system the neutron star accumulates matter from the companion star. Since the neutron star has a magnetic field and is spinning rapidly, the falling matter forms a disk around the star, the accretion disk. When the matter reaches the surface of the neutron star it becomes very hot and the emitted radiation can only escape along the magnetic axis of the neutron star. Since the axis of rotation and the magnetic field are not aligned the emission beam shows precession. This can be seen as the pulsation of the X-ray emission.

Some sources were found to suddenly appear in the sky, remain bright for several days to a few weeks, and then slowly fade again until no longer observable. Interestingly the progress of the decay of the emission is related to the absolute brightness of the source. Most of these sources, called X-ray transient sources, were understood to be supernova explosions (Mazzali et al. 2007).

The inner regions of some galaxies were also found to emit X-rays. The X-ray emission from these active galactic nuclei is believed to originate from ultra-relativistic gas near very massive black holes at the Galaxy's center (Kraft et al. 2007). And finally a diffuse X-ray mission was found originating from all over the sky. As of today the source of this emission is not clearly identified (Snowden 2011).

1.2 Advancements in Instrumentation

1.2.1 First Rocket Experiments

The first cosmic X-ray source, Scorpius X-1 and the cosmic X-ray background were discovered simultaneously in 1962 with a rocket experiment of the National Aeronautics and Space Administration (NASA), which was equipped with a Geiger-Müller counter with the aim of detecting X-rays reflected from the Moon (Giacconi et al. 1962).

1.2.2 Uhuru

On 12 December 1970 NASA launched its first dedicated X-ray satellite Uhuru from the San Marco platform near Malindi, Kenya. The satellite is also known as Small

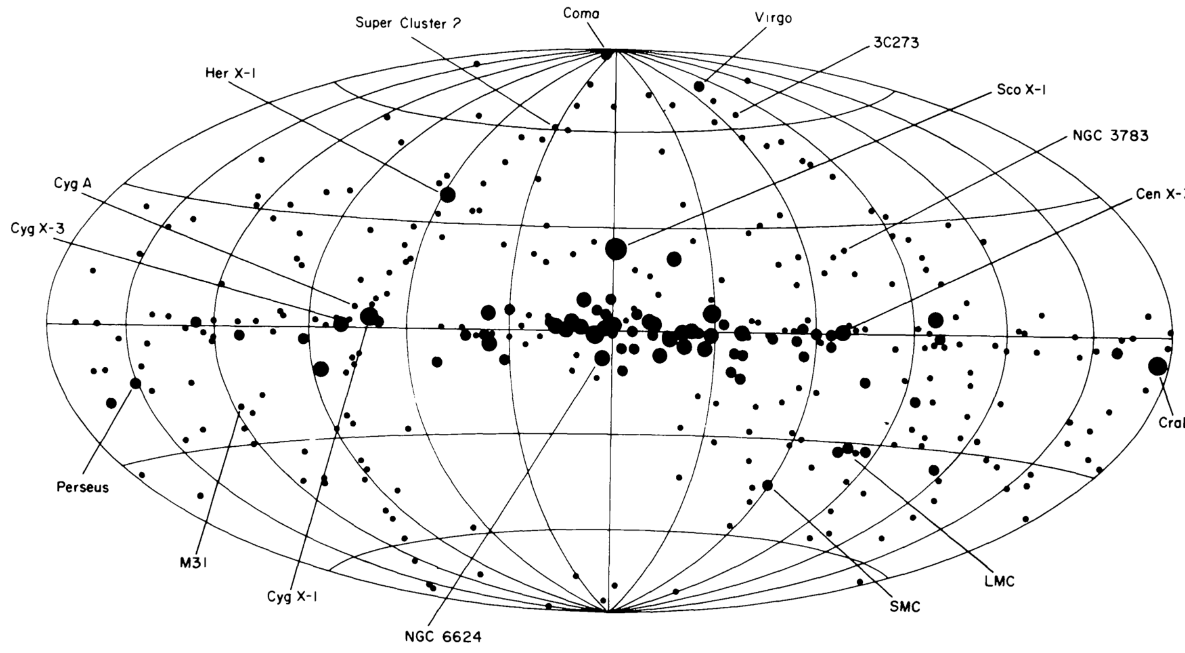


Figure 1.1
The sources from the Fourth Uhuru Catalogue displayed in galactic coordinates. The size of each symbol representing a source is proportional to the logarithm of the peak intensity. Source: Forman et al. (1978).

Astronomical Satellite 1 (SAS-1) but was renamed after launch to "Uhuru", which is Swahili for "freedom", to honour its launch date, the seventh anniversary of Kenya's independence. Uhuru carried two sets of proportional counters with a total effective area of $0.084\,\mathrm{m}^2$ covering an energy range from 2 keV to 20 keV (Santangelo and Madonia 2014).

The science payload, at 64 kg, weighed no more than a typical rocket experiment at the time. The two sets of conventional proportional counters with simple honeycomb collimators were used to undertake the first X-ray survey of the sky. The spacecraft was spin stabilized at 12 minutes per revolution. One detector had a field of view of 1 degree x 10 degrees, so it viewed each source for 2 s during each scan. A second detector had a field of view of 10 x 10 degrees and spent 20 seconds on sources during each scan. Uhuru scanned sky regions several times (typically about 60 passes per 24 hour period) thereby greatly increasing its sensitivity to weak sources. The net result of this was that Uhuru was able to detect X-ray sources 10 times fainter than the faintest detectable sources during earlier rocket flights down to a limiting sensitivity of about 0.001 of the intensity of the Crab Nebula. Although not uniform in sensitivity, 95% of the sky was scanned during the 2.5 year lifetime of the mission. Finally the Uhuru catalogue was issued, containing 339 objects as shown in Figure 1.1.

1.2.3 Einstein

NASA's second of three High Energy Astrophysical Observatories, HEAO-2, renamed Einstein after it became operational, was launched 13 November 1978 and was the first satellite mission that used X-ray telescopes with mirrors. This was made possible when in 1951 the physicist Hans Wolter at the University of Kiel developed a mirror configuration consisting of paraboloid and hyperbolic mirrors mounted confocally and coaxially that was able to focus X-ray light. These Wolter telescopes had already been used on Skylab in the early 1970's to investigate the corona of the Sun (Underwood et al. 1977).

The Einstein observatory however was the first imaging X-ray telescope put into space that was pointed at the sky. The few arcsecond angular resolution, the field-of-view of tens of arcminutes, and a sensitivity several 100 times greater than any mission before it provided, for the first time, the capability to image extended objects, diffuse emission, and to detect faint sources. It made the first X-ray images of shock waves from exploded stars, and images of hot gas in galaxies and clusters of galaxies. Einstein also located accurately over 7000 X-ray sources and detected X-ray jets from Cen A and M87 aligned with radio jets.

1.2.4 EXOSAT

The European Space Agency's EXOSAT (European X-Ray Observatory Satellite), was operational from May 1983 to April 1986. Similar to the Einstein Observatory it carried two Wolter telescopes and two proportional counters of which one had a total effective area 3 times larger than anything flown before. During its lifetime, EXOSAT made 1780 observations of a wide variety of objects, including active galactic nuclei, stellar coronae, cataclysmic variables, white dwarfs, X-ray binaries, clusters of galaxies, and supernova remnants. It also discovered Quasi Period Oscillations in LMXRB and X-ray Pulsars and measured the iron line in galactic and extra galactic sources (Taylor et al. 1981; White and Peacock 1988). Natural decay of the orbit caused the satellite to enter the atmosphere on 6 May 1986.

1.2.5 ROSAT

The Roentgensatellite or ROSAT, a joint venture between Germany, the United Kingdom and the United States, carried even larger X-ray telescopes into orbit on 1 June 1990. The two imaging telescopes operating in the soft X-ray (0.1 keV to 2.4 keV) and EUV (0.06 keV to 0.2 keV) ranges consisted of four / three nested Wolter type I-mirrors. For the manufacturing of the X-ray mirrors - the largest and most accurate ones at that time - Zerodur, a ceramics with zero thermal expansion and an extremely low surface roughness, was used for the first time. The telescopes focused X-rays on two position sensitive proportional counters (PSPC) with a spatial resolution of 20 arcsec and one high resolution imager (HRI) with a spatial resolution of 5 arcsec. The satellite and the X-ray telescopes had been designed, built and operated by Germany while NASA provided the Delta launch and the high resolution imager and the UK built and operated the EUV telescope.

ROSAT has expanded the number of known X-ray sources to more than 60 000 and has proved to be especially valuable for investigating the multi-million degree hot gas present in the upper atmospheres of many stars. ROSAT also performed the first all sky survey with imaging telescopes leading to the discovery of 125 000 X-ray and 479 EUV sources (Trümper 1984; Voges et al. 1999). In addition the diffuse galactic X-ray emission was mapped with unprecedented angular resolution (< 1 arcmin).

1.2.6 ASCA

After the great successes of satellites with imaging X-ray telescopes ASCA (Advanced Satellite for Cosmology and Astrophysics) was designed to study the detailed distribution of X-rays with energy. Therefore ASCA carried four large-area X-ray telescopes, each composed of 120 nested gold-coated aluminum foil surfaces. At the focus of two of the telescopes located was a Gas Imaging Spectrometer (GIS), while a Solid-state Imaging Spectrometer (SIS) was operated at the focus of the other two. The sensitivity of ASCA's instruments allowed for the first detailed, broad-band spectra of distant quasars to be derived. In addition, ASCA's suite of instruments provided the best opportunity at the time for identifying the sources whose combined emission makes up the cosmic X-ray background. ASCA was launched on 20 February 1993 and operated successfully till 15 July 2000 when it was transferred into a safe-hold mode. The satellite re-entered on 2 March 2001 after 7 and half years of scientific observations.

1.2.7 RXTE

Although RXTE did not have focusing X-ray mirrors, it had the unique capability to study rapid time variability in the emission of X-ray sources over a wide range of X-ray energies. The Rossi X-ray Timing Explorer was launched on 30 December 1995 from NASA's Kennedy Space Center. Originally designed for a required lifetime of two years with a goal of five, RXTE surpassed that goal and completed 16 years of observations before being decommissioned on 5 January 2012. The mission carried two pointed instruments, the Proportional Counter Array (PCA) to cover the lower part of the energy range (2 keV - 60 keV), and the High Energy X-ray Timing Experiment

(HEXTE) covering the upper energy range (15 keV to 250 keV). These instruments used collimators to limit their field of view to 1° which did not provide any imaging capabilities but allowed for a timing resolution of 1 µs (Gruber et al. 1996; Rothschild 1996). In addition, RXTE carried an All-Sky Monitor (ASM) that scans about 80 % of the sky every orbit, allowing monitoring at time scales of 90 minutes or longer.

1.2.8 BeppoSAX

BeppoSAX was a program of the Italian Space Agency with participation of the Netherlands Agency for Aerospace Programs. It was launched on April 30, 1996 from Cape Canaveral, and was the first X-ray mission with a scientific payload covering more than three decades of energy - from 0.1 keV to 300 keV, with moderate imaging capability. BeppoSAX proved to be useful for X-ray imaging sources associated with Gamma-ray bursts, determining their positions with unprecedented precision, and monitoring the X-ray afterglow (Boella et al. 1997; Piro 1997). All in-orbit operations of the BeppoSAX mission ended in April, 2002. In April 2003, the spacecraft re-entered Earth's atmosphere and splashed down in the Pacific Ocean.

1.2.9 Chandra

One of the most important X-ray astronomy mission of the present decade is NASA's Chandra X-ray Observatory, which was launched on July 23, 1999. The unique feature of the Chandra observatory is the High Resolution Mirror Assembly, a Wolter 1 type X-ray telescope with a focal length of 10 m and an effective area of 400 cm^2 @ 5 keV. Chandra uses three instruments of which one is the High Resolution Camera (HRC) that can achieve a spatial resolution of 0.4 arcsec with a pointing uncertainty of less than 0.1 arcsec (Garmire et al. 2003; Weisskopf et al. 2000). The result of this very high angular resolution is impressively shown in Figure 1.2.

The optical elements used in Chandra's mirror have four paraboloid-hyperboloid pairs resulting in the ten meter focal length. The mirror shells are made from Zerodur because of its low coefficient of thermal expansion and because Zerodur permits extremely smooth polished surfaces to the degree of a roughness of only a few atoms. The mirror surface is also coated with iridium, a material more reflective than gold (Weisskopf 2011).

Figure 1.3 shows the design of the observatory.

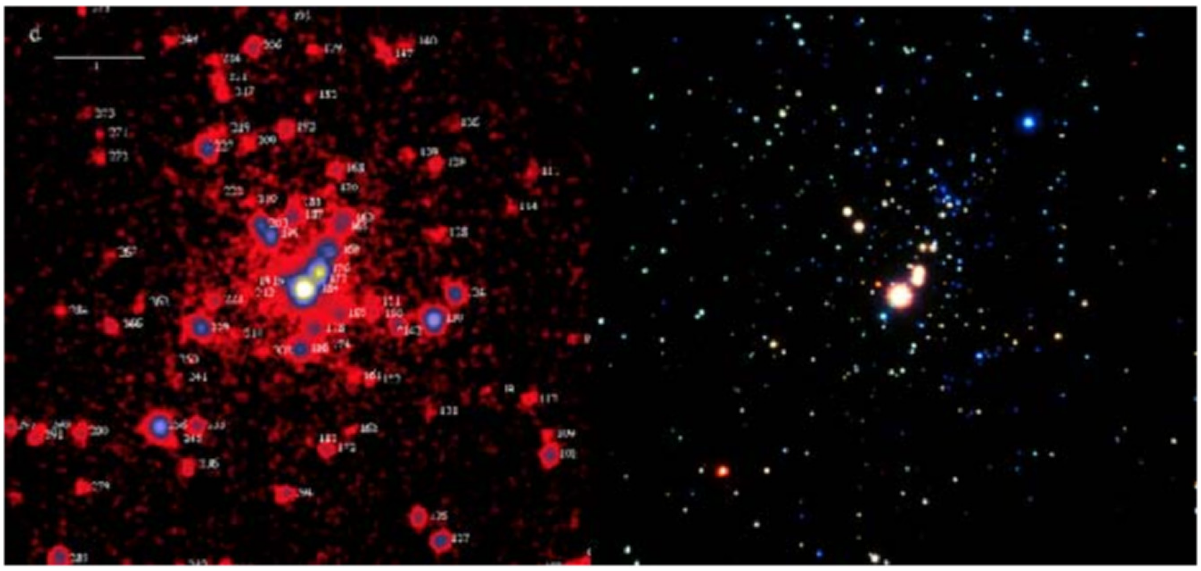


Figure 1.2
Left: X-ray image obtained by ROSAT shows 75 X-ray sources in the Orion star cluster. Right: The same region observed by Chandra resolves 1500 sources. Source: Weisskopf (2011).

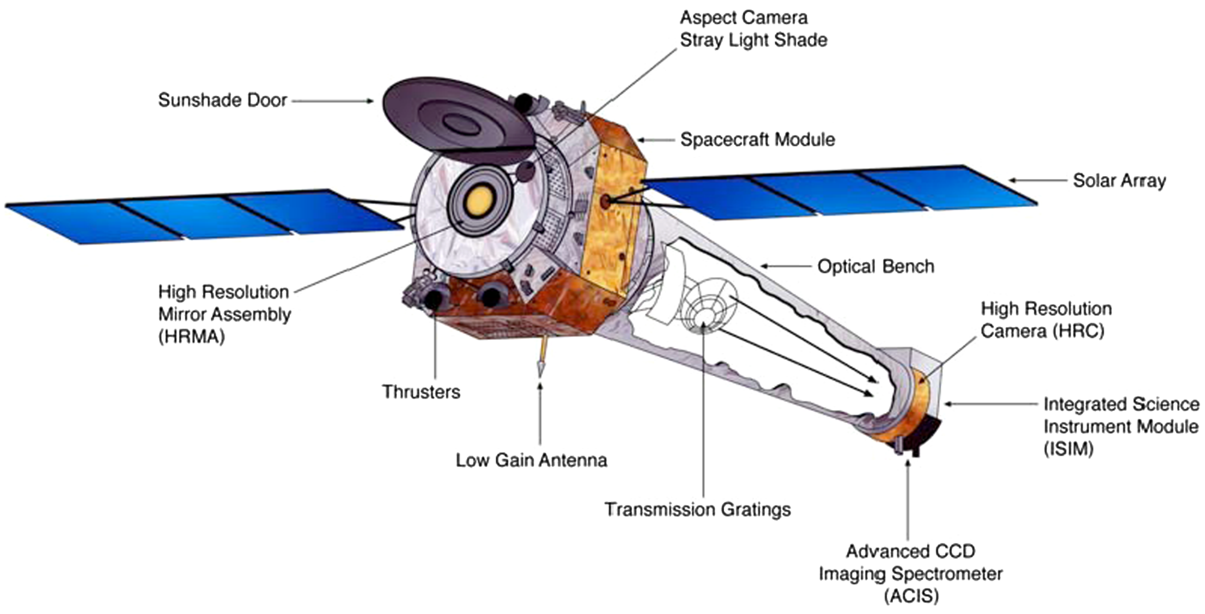

Aspect Camera Stray Light Shade
Sunshade Door
Spacecraft Module
Solar Array
Optical Bench
High Resolution Mirror Assembly (HRMA)
High Resolution Camera (HRC)
Integrated Science Instrument Module (ISIM)
Thrusters
Low Gain Antenna
Transmission Gratings
Advanced CCD Imaging Spectrometer (ACIS)

Figure 1.3
Chandra has two focal plane instruments. One is a High Resolution Camera (HRC).
This camera while similar to the HRI carried by ROSAT and Einstein has significantly smaller pore size of the MCP and a larger microchannel plate (MCP), lower background, charged particle anticoincidence mechanisms, and energy resolution. It is used for high resolution imaging, fast timing measurements, and for observations requiring a combination of both.
The second instrument, the Advanced CCD Imaging Spectrometer (ACIS), is an array of charged coupled devices (CCDs) that can be used simultaneously for imaging and spectroscopy. Images of extended objects can be obtained along with spectral information from the observed sources.
Source: Weisskopf (2011) and Schwartz (2004).

1.2.10 XMM

ESA's XMM (X-ray Multi-Mirror Mission) is one of the most successful missions launched by the European Space Agency. The telescope was launched on December 10, 1999. It utilizes three mirror modules, each is a Wolter 1 type grazing incidence telescope with a focal length of 7.5 m and a spatial resolution of 15 arcsec. The three telescopes are equipped with imaging cameras and spectrometers that operate simultaneously, together with a coaligned optical telescope. An overview of the telescopes design including mirrors and instruments is show in Figure 1.4 (Bagnasco et al. 1999; Lumb, Schartel, and Jansen 2012).

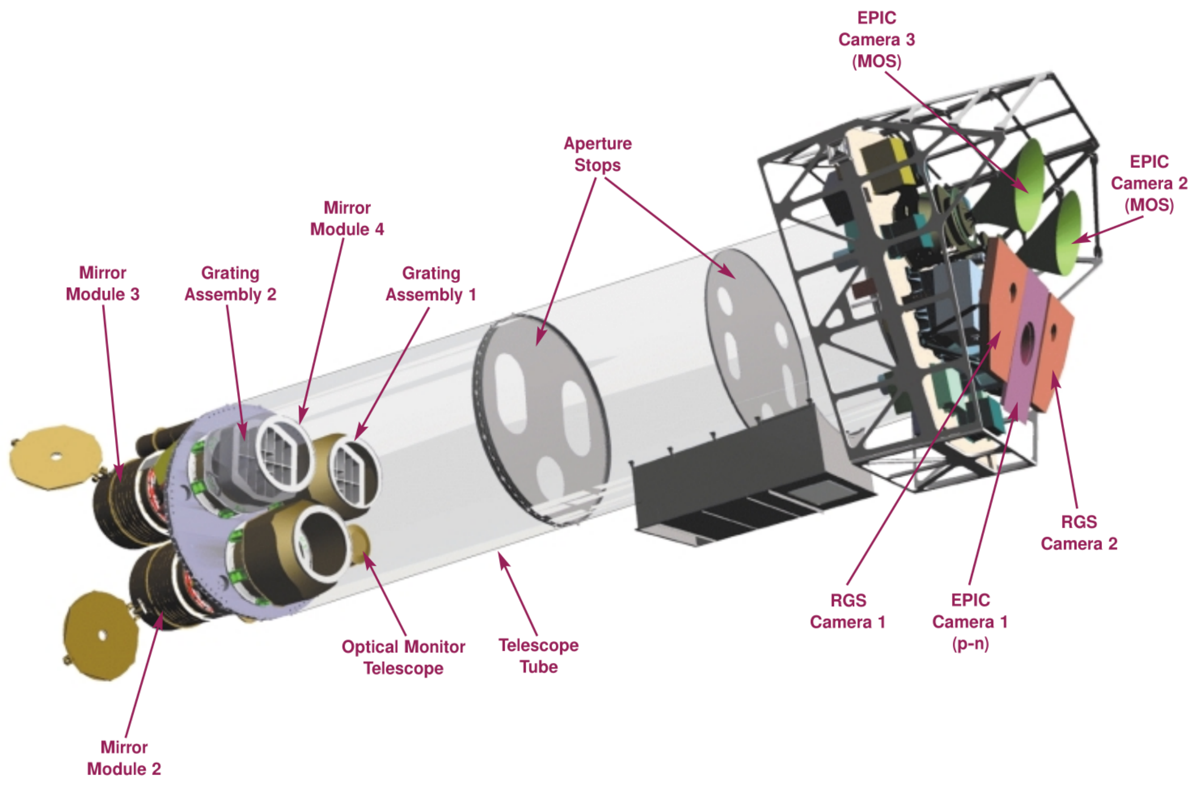


Figure 1.4
The X-ray Multi-Mirror Mission (XMM). The main components are the three Wolter 1 type grazing incident telescopes, two of them equipped with Reflection Grating Spectrometer assemblies (RGS). The three instruments are the two RGS cameras used for spectroscopy, the two EPIC MOS (metal-oxide) CCD cameras and another EPIC p-n CCD camera used for imaging. Source: Bagnasco et al. (1999).

Chapter 2

X-ray Mission Context

The programmatic context in which design and development of any space mission are carried out is an important part of the overall project. This context might be a national space agency with limited budget or a large consortium pooling the resources of several countries. While the latter obviously allows for more elaborate (i.e. expensive) mission concepts, the decision processes in such a context are much more complex and time consuming. Usually some sort of competition between several mission types, concepts, and designs is implemented.

Nevertheless, even the very first ESA mission entirely devoted to the study of X-rays, the EXOSAT mission (mission details in Chapter 1.2.4) launched in 1983, was developed and built by a multi-national European consortium including the United Kingdom, the Netherlands, Germany, and Italy. The ESA share of the satellites costs (development, integration, and testing) amounted to 52 million euros, while the construction of the instruments and satellite bus were contributions by the consortium. The ESA context in which the selection and management of missions takes place is explained in this chapter. A similar mechanism used by NASA is also discussed briefly in Chapter 2.2.

2.1 The ESA Cosmic Vision Program

Currently ESA has 20 member states (see Table 2.1) and one associated member (Canada). In a context of different national interests in space-based astrophysics it is difficult to decide which mission to develop, build, and launch next. Therefore, ESA has developed a mechanism that allows for a reasonable selection of important scientific themes of common interest in the European scientific community. The most important cornerstone for the long-term planning, selection, and execution of missions is currently the Cosmic Vision 2015-2025 program (Bignami et al. 2005).

This program is the successor to the previous definition of important questions to be addressed by space-based telescopes and observatories “Horizon 2000” (Longdon

1984). The Horizon 2000 program proved to be a great success when in 2005 the probe Huygens descended through Titan's atmosphere onto the moons surface. Development and execution of this mission took 23 years, from the submission of the proposal in 1982 through the production, integration, and validation phase (1988-1997) and a launch in 1997 to the arrival in 2005. This enormous effort was enabled by the Horizon 2000 program originally presented in 1984.

To provide a solid and scientifically justified base for decisions regarding the selection of suitable space missions ESA defines four main Cosmic Vision 2015-2025 Science Objectives. These are considered the most important themes of astronomical research across Europe:

1. What are the conditions for planet formation and the emergence of life?
2. How does the Solar System work?
3. What are the fundamental physical laws of the Universe?
4. How did the Universe originate and what is it made of?

These questions establish the scientific context in which ESA then issues several "Call for Missions" for ideas of next-generation observatories and experiments. Scientists from ESA member states (Table 2.1) can submit their ideas and selected missions will receive funding from national space agencies to be studied and refined in more detail. After a second selection only a few missions remain and receive further funding to conduct a Phase A study. While further refined at scientific facilities the mission is also evaluated by industry partners for a possible realization. One - rarely two - missions are selected after the assessment phase for a possible launch and the development and assembly of prototypes begins, eventually resulting in the production and testing of flight hardware and finally the launch.

2.1.1 ESA Cosmic Vision Timeline

This section summarizes important events related to the Cosmic Vision 2015-2025 program. It is compiled from several ESA press releases.

The first Call for Mission for the current Cosmic Vision 2015-2025 program was issued in March 2007, targeting two missions: one medium (M-class) and one large (L-class), for launch in 2017 and 2018. From 50 proposals received, four M-class candidates (Euclid, PLATO, Marco Polo, and Cross Scale), a mission of opportunity (SPICA, led by JAXA) and three L-class candidates (IXO, Laplace and TandEM) were selected for assessment. The LISA mission, carried over from the predecessor of the Cosmic Vision program "Horizon 2000 Plus", was included as an L-class candidate.

Table 2.1
Left: The 10 ESA founding members.
Right: 10 more countries joined the European Space Agency since then.

1. Belgium	11. Austria
2. Denmark	12. Czech Republic
3. France	13. Finland
4. Germany	14. Greece
5. Italy	15. Ireland
6. Netherlands	16. Luxembourg
7. Spain	17. Norway
8. Sweden	18. Poland
9. Switzerland	19. Portugal
10. United Kingdom	20. Romania

In 2009, the Cosmic Vision program was updated to reflect the overall funding availability, the technical progress of the studies, the situation of the mission technology readiness, and the availability of international partners. The two launch slots in 2017 and 2018 were confirmed, but were both designated as M (M1 and M2) missions. Solar Orbiter was reclassified as a sixth M-class mission candidate for M1/M2. The Laplace mission to the Jupiter system was selected as the outer planet candidate L-class mission. IXO, Laplace and LISA were maintained in the plan as candidates for the L1 launch opportunity, with a large involvement of international partners (NASA and JAXA), and targeting a launch year in 2020, subject to partnership consolidation.

The second call for Cosmic Vision missions was released in July 2010 with the goal of selecting a third M-class mission (M3) with a targeted launch date of 2024. From the 47 proposals submitted, four were recommended by the ESA Advisory Structure and selected for assessment. These missions (EChO, LOFT, MarcoPolo-R, and STE-QUEST) were joined by PLATO and became candidates for the M3 launch opportunity. In February 2014 the planet hunter PLATO was selected as medium-class science mission.

In 2011, ESA imposed major requirement changes on all three L-class mission candidates to take account of developments with ESA's international partners, namely NASA withdrawing from the L-class mission. During the reformulation exercise IXO became

ATHENA and Laplace became JUICE. After the selection of JUICE (Jupiter Icy Moons Explorer) as current L1 mission by ESA in 2012 ATHENA was again revised to become a candidate for the 2013 selection of a science theme for an L2 or L3 mission.

A call for a small (S-class) missions in the science program was issued in March 2012, advertising a single launch opportunity in 2017. The CHEOPS mission was later selected for implementation from a total of 26 proposals.

In March 2013, a Call for White Papers was issued asking the science community to propose science themes and associated questions that could be addressed by the next two large (L-class) missions, L2 and L3, which are currently planned for launch 2028 and 2034.

In September 2013 "The Hot and Energetic Universe" was in fact selected as science theme for an L2 mission with a possible launch in 2028. ATHENA was the only candidate for this mission slot. A mission proposal was submitted to ESA in April 2014 (Nandra 2014).

2.2 The NASA Astro2010 Decadal Survey

To define the most important astronomical and astrophysical themes the National Research Council of the National Academy of Sciences releases the Astronomy and Astrophysics Decadal Survey every ten years. The current survey "New Worlds, New Horizons in Astronomy and Astrophysics" was released in August 2010 and recommends priorities for the most important scientific and technical activities of the decade 2010-2020. Recommendations for scientific questions to be answered include:

- How did the Universe begin?
- What were the first objects to light up the Universe and when did they do it?
- How do cosmic structures form and evolve?
- What are the connections between dark and luminous matter?
- How do stars and black holes form?
- How do circumstellar disks evolve and form planetary systems?
- Why is the Universe accelerating?
- What is dark matter?
- What are the properties of the neutrinos?
- What controls the masses, spins and radii of compact stellar remnants?

While this definition of important questions to be answered by the science community is very similar to ESA's Cosmic Vision program, NASA uses a different process to select and finance suitable space missions that is shown in Figure 2.1.

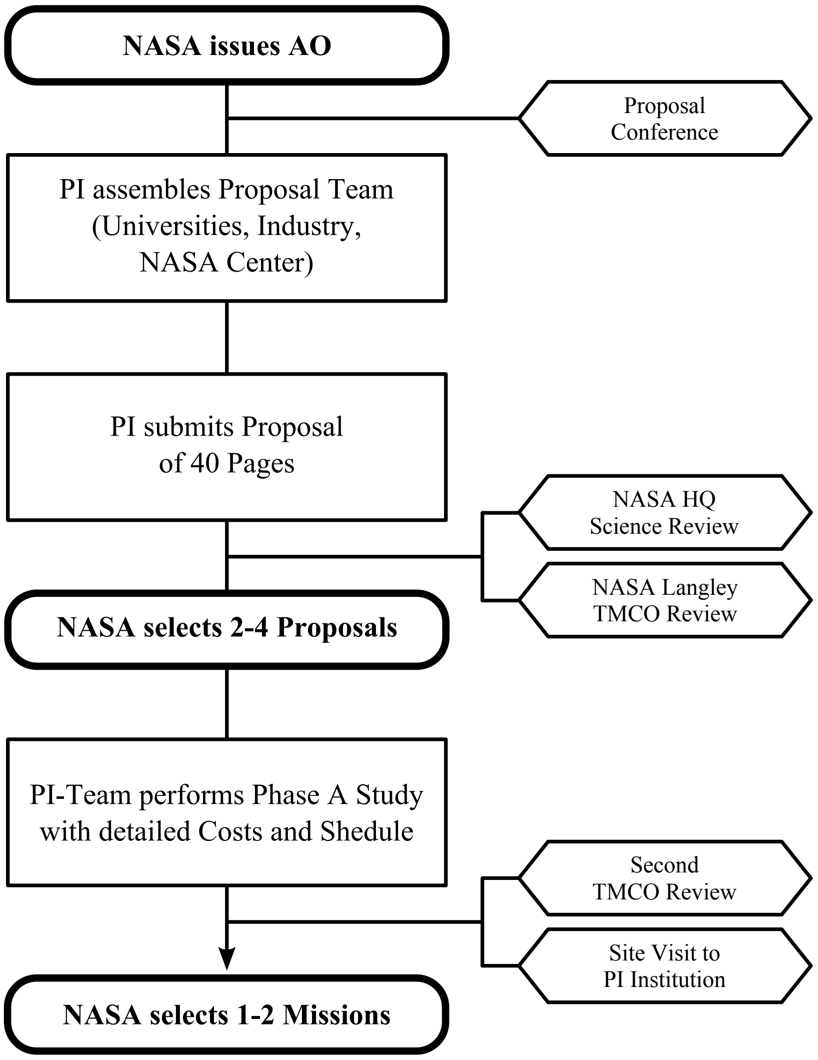

NASA issues AO
Proposal Conference
PI assembles Proposal Team (Universities, Industry, NASA Center)
PI submits Proposal of 40 Pages
NASA HQ Science Review
NASA Langley TMCO Review
NASA selects 2-4 Proposals
PI-Team performs Phase A Study with detailed Costs and Shedule
Second TMCO Review
Site Visit to PI Institution
NASA selects 1-2 Missions

Figure 2.1
Overview of the NASA Mission Selection Process.
After the Announcement of Opportunity (AO) the PI-Team submits a proposal for a mission concept. Scientifically relevant concepts are selected on which a detailed Phase A study is performed. Finally a technical, management, cost and other program factors (TMCO) panel assesses the feasibility of the mission implementation approach. Once selected by the TMCO and after at least one more formal confirmation review the mission is ready for realization. Derivative work; original figure: National Research Council (2006).

Part II

Data Processing Elements

Chapter 3

Hardware Layer

Usually the first choice to be made during the development of data processing equipment is the selection of the actual hardware that will be used to perform data handling tasks. Several types of processing hardware are available, each with its own performance, complexity, and costs. The selection is usually based on a trade between costs and performance on one side and complexity, weight, and power consumption on the other. Another aspect is the implementation of the desired functionality in software, e.g. by using a programming language like C for a microcontroller or using a hardware description language for an FPGA. While this chapter deals with the different hardware options, Chapter 4 provides insights into the software layer.

3.1 Microprocessors

The basic component of most data processing units is the Central Processing Unit (CPU) or microprocessor. Early uses of microprocessors in space include e.g. the General Electric 18-bit word TTL. It was used on the Voyager satellites launched in 1976, included 4096 bytes of RAM and could execute 25 000 instructions per second (Tomayko 1988). A modern example is the Dual BAE RAD750, a radiation-hardened version of the IBM PowerPC 750, running at 200 MHz with access to 256 megabytes of DRAM that is used e.g. by the Kepler Observatory (Yra et al. 2010).

Many different CPUs have been used aboard space missions. Such CPUs must be highly reliable and very durable since the temperatures in space, even with heaters, can vary widely and the radiation that a system is exposed to can be immense (up to 20 Gy/year). Therefore microprocessors used in space environment have to be designed and built including techniques like error checking and correcting memory (ECC RAM), radiation-hardened components, and hardware redundancies. They also have to be especially qualified for the use in the given environment. This qualification process represents the verification that a particular component's design, fabrication, workmanship, and application are suitable and adequate to assure the operation and survivability under the required conditions (Kayali 2007).

Unfortunately such a qualification process is rather expensive, in terms of costs as in terms of time and can easily take five to ten years. This significantly reduces the number of available components in comparison to the availability of processors for ground-based applications. Table 3.1 shows a list of space missions and the CPU(s) used.

When a microprocessor for space applications has been designed from suitable, space qualified components, the whole system is tested and validated extensively. If all components are already qualified, a simpler but still time consuming process is conducted on all flight hardware equipment. The time and work invested into the qualification of the whole system can be further reduced by selecting one of two possible approaches explained in detail in the next two chapters.

One approach is to use a System-on-a-Chip where many different components are already integrated into one complex system that is then thoroughly tested and qualified for the use in space (Chapter 3.2). Another approach is to implement the whole processing unit as a VHDL (Chapter 4.1) design into an FPGA (Chapter 3.3). FPGAs are available in space qualified versions and VHDL designs can be tested and verified for the most part in a software environment.

Table 3.1
CPUs used in space missions throughout the years. Source: John Culver (2014; via e-mail).

Mission	CPU
Cassini	1750A
Cluster (ESA)	1750A
MSTI-1,2	1750A
Rosetta (ESA)	1750A
EOS Terra	1750A
EOS Aqua	1750A & 8051
EOS Aura	1750A & 8051
Clementine	1750A, 32 bit RISC
MSTI-3	1750A, R-3000
Pluto Express	32 bit RISC
Sampex	80386, 80387
SMEX	80386, 80387
SWAS	80386, 80387
TRACE	80386, 80387
WIRE	80386, 80387
FUSE	80386, 80387, 68000
Surrey MicroSat	80386EX
FAST	8085
Galileo AACS	ATAC (bit slice) and 1802
SPOT-4	F9450
EO-1/WARP	Mongoose V
IceSat Glas	Mongoose V
MAP	Mongoose V, UTMC 69R000
CGRO	NSSC-1
Topex/Poseidon	NSSC-1
UARS	NSSC-1
EUVE	NSSC-1, 1750A
HST	NSSC-1/386, DF-224-486
Coriolis	RAD6000
Deep Space-1	RAD6000
Gravity Probe B	RAD6000
HESS	RAD6000
MARS 98	RAD6000
SIRTF	RAD6000
SMEX-Lite	RAD6000
Swift	RAD6000
Triana	RAD6000

3.2 System-on-a-Chip

The high-level operations of satellite systems are usually handled by a System-on-a-Chip (SoC). Its tasks include command and control of the instrument, relaying commands to distributed sub-units, handling the telemetry (not necessarily directly to the ground station but to the satellite bus), collecting and processing housekeeping data, and applying lossy or lossless compression algorithms to the collected data to reduce its size

A SoC is an integrated circuit (IC) that – very similar to a microcontroller – includes a central processing unit (CPU). It also includes additional advanced hardware components such as memory (RAM), timing sources and counters, peripheral components like USB, Ethernet, and SpaceWire interfaces, analog-digital-converters (ADCs), and different kinds of specialized components (audio, video, compression).

For reasons explained in Section 3.1 future missions will tend to replace SoCs by microcontrollers designed in a synthesizable hardware description language (e.g. VHDL) and implemented in an FPGA. While additional components can also be implemented in the FPGA creating a SoC like structure, some FPGA parts may still contain parallel logic providing the benefits described in Section 3.3.3.

3.2.1 Traditional SoC Design

In the traditional design, different data sources (i.e. scientific space instruments) are equipped with stand-alone data processing units (DPUs) using radiation-hardened parts. Present available radiation-hardened processors (e.g. ERC32, TSC21020 or AT697) offer low to medium processing performance and therefore are typically used for control and sequencing tasks only (Fiethe et al. 2007). To handle the high data rates of modern instruments, DPUs need to include some dedicated hardware for offline processing via mass storage or online processing, e.g. data compression. Such classical systems show clear disadvantages in resource allocation (low integration density and moderate performance). The state-of-the-art solution could be a complete SoC system implemented within a radiation-hardened ASIC. But this approach has the drawback of a long development time and low adaptability to changes in mission requirements due to the fact that the implementations have to be fixed to a simplified scheme, which needs to be frozen in an early project phase.

Commercial off-the-shelf (COTS) DPU hardware uses standard components and plastic encapsulation. This approach combines good performance with small outline lightweight packages and the intrinsic high reliability of high volume production (Fiethe et al. 2007). Usually plastic encapsulation and failure rates of COTS parts are acceptable or can be made acceptable for a space environment by additional measures. This allows very compact DPU designs (Fiethe et al. 2003; Gliem and Gerlach 2001). The big disadvantage of using COTS parts is that every part has to be tested for radiation resistance and its correct operation has to be verified under space-like environmental conditions. This needs a lot of preparatory work and due to the short availability time of commercial parts, such components need to be selected and fixed in a very early phase of the project or they might otherwise not be available later on.

3.2.2 Reconfigurable SoC Design

The availability of radiation tolerant FPGAs and processor technology enable new approaches to the instrument system architecture. An advanced SoC design integrates special functions (e.g. data compression or formatting/coding (Michalik et al. 2006)) together with the processor system in a single FPGA. Figure 3.1 shows the general architecture of such an advanced DPU/SoC design.

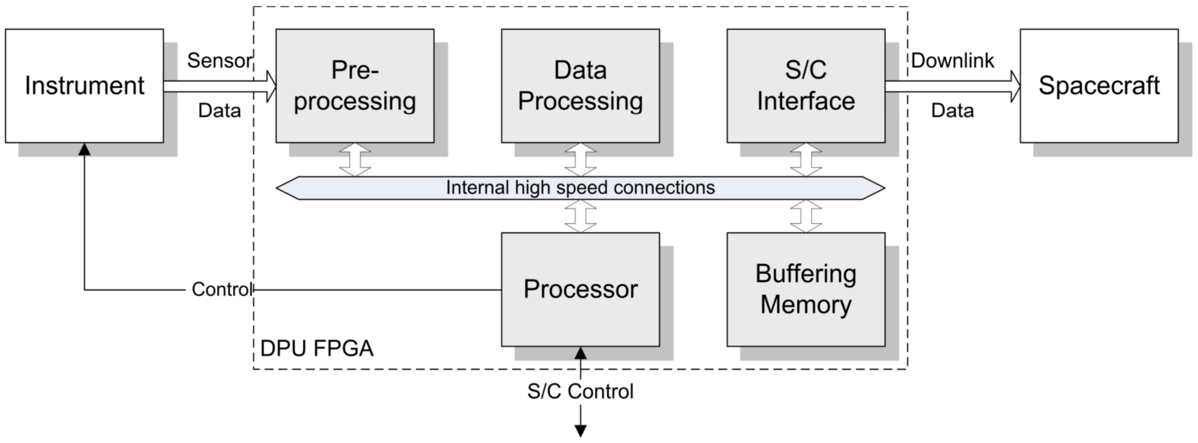


Figure 3.1
SoC architecture using an FPGA. The SoC design integrates specialized functions (like data processing) together with the main processor (CPU) in a single FPGA. Interfaces to the spacecraft (S/C) are included. Source: Fiethe et al. (2007).

3.3 Field-Programmable-Gate-Array

A Field-Programmable-Gate-Array (FPGA) is a silicon chip that can be re-programmed even when already in use (field-programmable). The chip contains a number of components that are grouped into functional units – so called "slices" – inside a semiconductor device. Slices on the chip are connected via programmable interconnects and form "logic blocks". Modern FPGA devices contain large numbers ($\gg 10\,000$) of blocks enabling them to provide the capability to implement huge and complex circuits.

Using a hardware description language (HDL) engineers can realize arbitrary circuit designs in an FPGA. The work done in the context of this thesis, implementing a LEON3 microprocessor in a Spartan FPGA (see Chapter 5, IXO), and developing the LOFT LAD PBEE and implementing it in a Virtex FPGA (Chapter 6, LOFT), was done using VHDL.[1] The description of an FPGA design has some similarities to the use of a programming language, such as a certain syntax, modularity, conditional constructs and loops, data types. But it is also different by not using sequentially processed commands but parallel executed statements instead.

FPGAs can be considered "becoming" any circuit design by a chain of processes that is described in detail in Chapter 4.1.2 - Design Implementation on page 38 and visualized in Figure 4.2 on page 39. Due to their high flexibility, and the ability to be re-programmed repeatedly, FPGAs are used especially in the development of new hardware designs. They can also be used very cost efficiently when only a small number of rather complex processing units is needed and the initial development of an ASIC is too expensive and longsome. Both cases are certainly true for developing and operating a space-based telescope. But also in the commercial satellite business in general the use of FPGA based hardware solutions is increasing (Habinc 2002).

3.3.1 Advantages over Microprocessors

One simple argument for the superiority of FPGAs over microprocessors is the fact that the LEON3 microprocessor can be implemented in an FPGA, thus the FPGA becomes the microprocessor. A more serious advantage is the fact, that FPGAs are inherently capable of real hardware-parallelism. All functional units (logic blocks) inside an FPGA are operating in parallel. All operations on input signals are done on the e.g. rising edge of a clock signal, sequential operations on internal data have to be implemented manually (which can be done, see Chapter 4.2). The parallel execution can make the implementation of complex algorithms very difficult since sequentially

[1] The "V" in VHDL is short for VHSIC, which itself is the acronym for Very-High-Speed-Integrated-Circuit.

written code will not produce sequential operations. This is a very basic and important characteristic in the description of an FPGA design and is therefore explained in more detail in Section 3.3.3.

There are however applications for microprocessors that can not simply be implemented in an FPGA. A microcontroller loads its instructions entirely from external memory, even though this memory might be integrated in a SoC design. It can, when given enough memory, which nowadays is rarely a limiting factor, process an arbitrary long sequence of operations and therefore perform almost arbitrary complex calculations. Since the equivalent to the microcontroller's instructions is hardwired in the FPGA's logic blocks there is a natural limit to the design size. A fundamental advantage of microcontrollers are their significantly lower prices. FPGAs are easily a factor of 10 more expensive than modern microprocessors.

3.3.2 Advantages over ASICs

When a digital circuit has been designed, assembled from single parts, and tested, it is eventually produced as an integrated circuit (IC). Whenever large numbers of ICs are used in a complex design, the final assembly is implemented in one single complex part, the Application-Specific Integrated Circuit (ASIC). The major difference in comparison to an FPGA is that the ASIC can not be modified after production. The development process of an ASIC however is very similar to the development of an FPGA design, the hardware description language VHDL can be used in both cases.

The final production of an ASIC is much cheaper than the production of the extremely complex structure of an FPGA, since only the final circuit is realized. On the other hand, the costs of the initial development of the required photomasks, and the assembly of a production line are very high, reaching up to several hundred thousand euros. One reason for this is the need for an extremely high quality of the masks, since every error is passed onto the final product.

3.3.3 FPGA Hardware Parallelism

This section explains in detail the consequences of the real hardware parallelism of statements in an FPGA design written in e.g. VHDL. Note that a CPU is not capable of real hardware parallelism.[2] A CPU always executes instructions sequentially. A user may have the impression that a CPU can work on several tasks at the same time because it switches between the execution of different processes very quickly (more than hundred times per second) when running at up to 3 GHz.

The VHDL source code shown in Code Example 3.1 describes one functional entity (hence the keyword entity). Many of such entities of different size and complexity are used to build the final design in a modular way. The entity shown is supposed to take an 8-bit value on a parallel bus as input and to process this value to produce an 8-bit result on a parallel bus as output.

The section starting with "main: process (clk)" contains the logic that is to be applied to the input signals on the rising edge of a clock signal. This is achieved by using the if-statement with "rising_edge(clk)". Two different things are specified to happen:

1. The 8-bit value in the register "input" is transferred into a local register "A".
2. The value in the local register "A" is mathematically added as a binary number with the value in register "input" and the result is placed in the register "output".

The code is written in sequential statements. A CPU would process these statements sequentially and the result would be the addition of the input value with itself which is placed on the output bus. This result would be equal to the multiplication by 2 of the input value. That is not what's going to happen!

The reason for the very different behavior lies within the parallelism of the statements. Those statements are not interpreted as commands being processed by a CPU but instead they create and connect flip-flops, adders and other simple logic blocks. Those logic blocks are wired to the clock signal and are activated at the same time, the rising edge of the clock. Therefore while the input value is transferred into the local register "A", the current value (the old one) of register "A" is added with the (new) input signal!

Result: If the input value changes, on the next rising edge of the clock the old value of the input (which is still located in "A") is added with the new value of input and the result is placed in the output register. Only after another clock cycle and with a constant input the result would in fact be equal to the multiplication by 2.

[2] While the introduction of multi-core CPUs enabled small-scale hardware parallelism (a few processes running at the same time), the technical details like inter-process-communication, and uniform / non-uniform memory access still remain very challenging.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

-------------------------------------------------------------------------------
-- E N T I T Y
-------------------------------------------------------------------------------
entity Surprise is

    port(
    clk    : in  std_logic;
    input  : in  std_logic_vector(7 downto 0);
    output : out std_logic_vector(7 downto 0)
  );

end entity;

-------------------------------------------------------------------------------
-- A R C H I T E C T U R E
-------------------------------------------------------------------------------
architecture Behavioral of Surprise is

  signal A : std_logic_vector(7 downto 0);

begin

  main: process (clk)
  begin
    if rising_edge(clk) then
      A <= input;
      output <= input + A;
    end if;
  end process;

end architecture;
```

Code Example 3.1
This entity demonstrates the important concept of real hardware prallelism in FPGAs. The VHDL design is split into the declarative entity header and the functional architecture body. A process is defined that contains the logic of the module.
The clock synchroneous process (activated on “rising edge”) reads an 8-bit input value into the local register “A”. *At the same time* the (old) value “A” is added with the (new) input value and written to the output regsiter. The result does not equal the multiplication by 2 of the input value, but rather the addition of the current input value with the last one. The written order of the statements has no effect.
While hardware parallelism allows for very fast data processing, it can make understanding the functionality of an unknown entity very difficult.

3.3.4 FPGAs @ IAAT

Because of the advantages that FPGAs can have over microprocessors and ASICs (presented in the previous sections 3.3.1 and 3.3.2) different FPGAs are used for the prototype boards developed in the context of this thesis. Both applications are described in-depth in Chapters 5 and 6.

HTRS DPU

For the Data Processing Unit (DPU) of the High Time Resolution Spectrometer (HTRS), an instrument aboard the International X-ray Observatory (presented in Chapter 5.5), a Xilinx Spartan 3 XC3S on a Pender GR-X3CS development board was used to build the DPU prototype. The Spartan 3 was chosen since a similar FPGA, the Virtex 4, is available as radiation-hardened version qualified for the use in space. The board manufactured by Pender Electronic Design is shown in Figure 3.2.

Both FPGAs – when used to implement a LEON3 microcontroller – offer better performance (processing speed) as already available, space qualified, off-the-shelf microcontrollers. The LEON3 microcontroller (which is discussed in more detail in Chapter 4.3 is chosen for implementation because the Data Processing Unit of the HTRS has to execute complex compression algorithms which require the use of the programming language C to implement the desired functionality.

Additionally the DPU is required to process the data from multiple independent sources in parallel and therefore this part of the functionality is implemented directly in the FPGA to make use of the parallel processing capabilities discussed in 3.3.3. It is correct to say that in a way the DPU prototype uses the best of both worlds.

LAD PBEE

The second prototype, the Panel-Back-End-Electronics (PBEE) of the Large Area Detector (LAD), the main instrument of the Large Observatory for X-ray Timing (presented in Chapter 6.6), uses a Xilinx Virtex 4 on a custom designed printed circuit board (PCB) shown in Figure 3.3.

The PBEE implements 21 custom interfaces operated in parallel to communicate with and receive events from the Module-Back-End-Electronics (MBEE). The event data received from the 21 interfaces is at some point serialized and sent through a single space-wire link to the LAD instrument computer. This is a prime example of the hardware parallelism that only FPGAs are capable of providing.

The prototype board also contains an additional Xilinx Spartan 3 FPGA that is used to simulate two MBEE interfaces. This FPGA can be connected via USB to a PC to receive simulation data that is then sent to the PBEE. It can also be used to simulate the interface from the PBEE to the Data Handling Unit (DHU), albeit this connection does not use a SpaceWire interface.

3.4 Mass Memory

The design of mass memory as well as the design for all other systems used for space applications must take into account the difficult conditions in a space environment. A large number of effects like ionizing radiation (especially important for the storage of data), thermal conditions (e.g. between development/operation on ground and in space), and mechanical stresses (particularly during launch) have to be considered. The usual approach to this problem is the development of dedicated, space qualified, radiation-hardened hardware components, such as ASICs and FPGAs. The obvious drawbacks are very high costs due to the small number of units needed for a mission and the time consuming development process itself.

A better solution is the adaptation of already available (off-the-shelf) hardware to the given environment. Mass memory is well-suited for this approach since the components used are usually rather simple, compared to the complex structure of processing or interface and communication hardware. The development over the past years in this area, and particularly the advancements in miniaturization, have lead to a rapid growth in the storage capacity of mass memory components. The introduction of solid-state mass memories has created storage solutions which are easily competitive with tape recorders due to higher reliability, comparable density, and better performances. Solid-state mass memories have no moving parts and their operational flexibility has made them suitable for many applications, space missions being one of them.

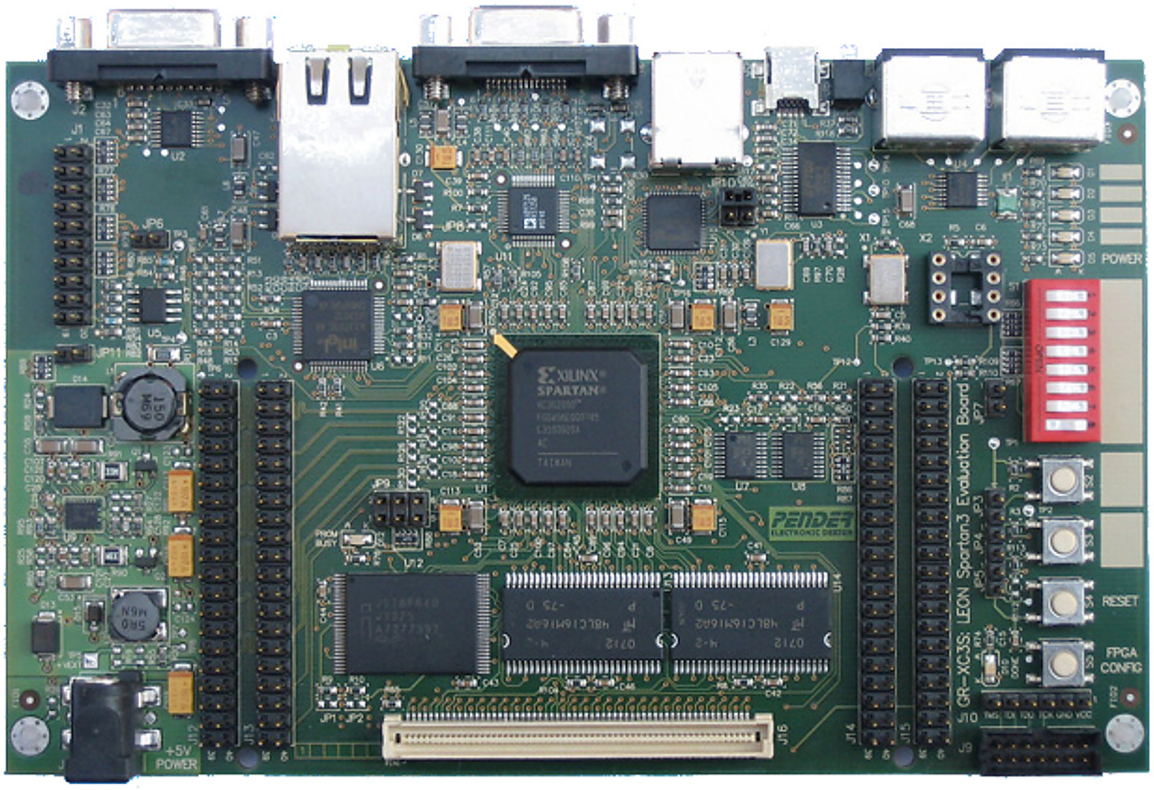


Figure 3.2
A Xilinx Spartan 3 FPGA on a Pender GR-XC3S development board. This board was used for the development of the Data Processing Unit of the High Time Resolution Spectrometer aboard the International X-ray Observatory.
A microcontroller design (LEON3) written in VHDL can be implemented in the Spartan 3 FPGA. The board features on-board RAM, an Ethernet controller, two USB controller, VGA video output, and two RS232 interfaces.
The FPGA is connected to an on-board 50 MHz oscillator and can be programmed via the Ethernet, serial, or an additional JTAG interface. Source: Pender Electronic Design GmbH, Switzerland.

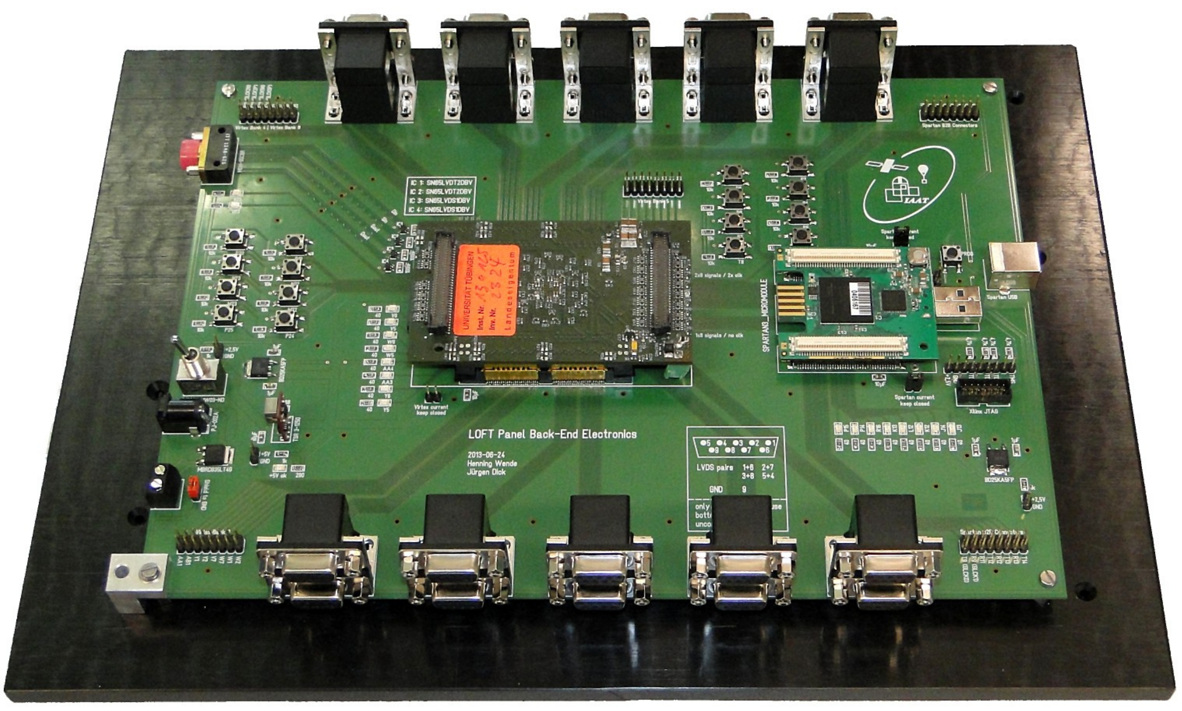


Figure 3.3
The custom designed prototype board for the Panel-Back-End-Electronics (PBEE) of the Large Area Detector (LAD) aboard the Large Observatory for X-ray Timing (LOFT). The board features 20 connectors for the custom designed interface to the Module-Back-End-Electronics.
While the primary FPGA, a Xilinx Virtex 4, is located at the center of the board, an additional FPGA, a Xilinx Spartan 3, was also integrated into the design (right side). This second FPGA has two on-board MBEE interface connections to the PBEE and thus can be used to simulate two MBEEs.
A full test run using two simulated MBEEs via the second FPGA and one prototype MBEE board was used to validate the parallel interface communication by sending commands and receiving simulation data.

Chapter 4

Software Layer

During the early design phase of any astrophysical mission appropriate hardware components have to be selected based on the required performance and the different characteristics as discussed in Chapter 3. After the hardware has been selected a suitable software layer has to be chosen to implement the desired functionality. While the operation of microprocessors usually require the use of a programming language, the development of an FPGA design is done using a hardware description language (HDL).

In the context of this thesis two different FPGA models were used to develop and operate prototype boards. For both models the design was written in the hardware description language VHDL.[1] Therefore Chapter 4.1 presents this hardware description language in more detail and also explains how a HDL design is implemented in an FPGA (4.1.2).

The custom designed interface used for the communication between several components of the LOFT observatory was implemented in VHDL using the *Two Process Method*, which is a special technique of writing VHDL code. Since the entire design of one of the two prototypes developed (LAD PBEE) was also written using the *Two Process Method*, Chapter 4.2 is devoted to a detailed explanation of this technique.

The other prototype design (HTRS DPU) includes a LEON3, a microcontroller design written in VHDL to be implemented in an FPGA. The functionality of this prototype was implemented in the LEON3 core using the programming language C. Additional data processing capabilities were also implemented in VHDL in the same FPGA. Chapter 4.3 is devoted to the LEON3.

Finally Chapter 4.4 deals with the different interfaces used between the components of modern data processing systems aboard space missions.

[1] VHDL is short for **V**HSIC **H**ardware **D**escription **L**anguage.
VHSIC stands for **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit.

4.1 VHDL

As explained in Chapter 3.3 an FPGA differs in several ways from a microcontroller. There is no CPU that processes program instructions stored in binary machine code. Instead an FPGA is configured to reproduce a given circuit design by mapping it into the FPGA hardware structure. The mapping can be changed very late in the design cycle, even after the end product has been deployed in the field.

The configuration/design of the internal FPGA structure is specified using a hardware description language (HDL). The two most common languages are Verilog and VHDL.

4.1.1 General Structure

A digital circuit designed for an FPGA by the means of VHDL is usually composed of a number of modules called “entities” with in- and outputs. The outputs of an entity are functions of the inputs and sometimes of time given by any number of clock signal transitions. The left unit in Figure 4.1, the entity denoted F, is a function $Y(A,B)$.

One way of describing the inner structure of an entity is by assembling it from a number of subsequently less complex sub-entities, whose ports are connected by signals specified in the containing entity. This technique is visualized in the right part of Figure 4.1.

This method is called the “structural description” of an entity. Another way of describing the functionality is by using the “behavioral description”. This can easily be illustrated by a simple example:

> Assume the entity shown in Figure 4.1 should simply realize an OR connection of the two input signals A and B. The one important line of code would then be:
>
> $$Y = \bar{A}.B + A.\bar{B}$$
>
> The final source code would have a few more lines, e.g. to specify the ports and types of signals used, but this single line would define the behavior of the entity.

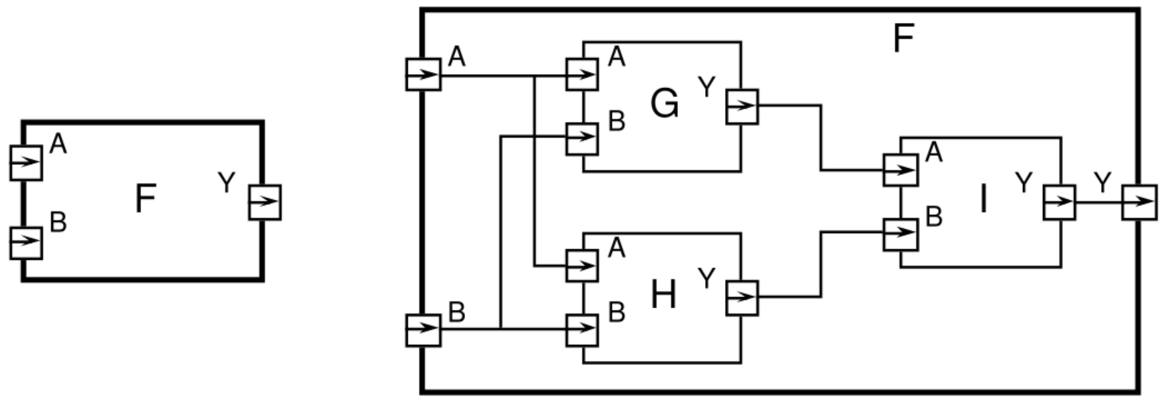


Figure 4.1
Structure of a VHDL entity. Source: Ashden (1990).

To describe the functionality of an entity using the behavioral description the designer can use VHDL language elements very similar to constructs used in programming languages. These include:

- temporary variables
- numbers and constants defined in central, project-wide locations
- strings and other non-trivial data types
- assignments
- if cases
- do, while, and for-loops
- switch-case procedures
- functions

However, there are significant differences when writing VHDL code compared to the source code of any programming language. The reason for this is that FPGAs do not execute commands (like microprocessors), instead VHDL describes the internal, parallel processing structure of the device. This true hardware parallelism is the subject of Chapter 3.3.3 - FPGA Hardware Parallelism (p. 28).

4.1.2 Design Implementation

Several steps lead from the first idea of an VHDL design to the final bitstream that can be used to implement the design into hardware[2]. The VHDL source code can not simply be compiled but rather has to be synthesized into a netlist. During this process every entity is translated into a structure that uses the FPGA's elementary units, the logic cells or slices, to realize the desired function $Y(A,B)$ (see Figure 4.1), e.g. by using look-up-tables. During this process dedicated components inside the FPGA might be allocated, e.g. block-RAM, digital clock manager, and FIFOs.

The next important step is routing the design through the FPGA. At this stage interconnections between entities are generated and the physical location is fixed. The position of the used input and output ports (to the device) is specified at this point in a user constraints file and has a significant effect on the process. The routing of the clock distribution network through the design usually determines the available clock frequency at this point, since clock skew for a given frequency has to be avoided.

Finally a bitstream is generated from the design that contains information about the process of implementing the design into the target FPGA. The bitstream can also be converted to a PROM file that is written into the devices flash memory and allows the FPGA to reconfigure itself upon the power-on sequence. Without a PROM file the FPGA will be “blank” after turning the power off.

The whole process is visualized in Figure 4.2.

4.2 The Two Process Method

The most commonly used design style for synthesizable VHDL models is what can be called the “dataflow” style. A larger number of concurrent VHDL statements and small processes connected through signals are used to implement the desired functionality. Reading and understanding dataflow VHDL code can be very difficult since the concurrent statements and processes do not execute in the order they are written, but when any of their input signals change value. It is not uncommon that when extracting the functionality of dataflow code, a block diagram has to be drawn to identify the dataflow and dependencies between the statements. The readability of dataflow VHDL code can be compared to an ordinary schematic where the wires connecting the various blocks have been removed, and the block inputs and outputs are labeled with signal names!

[2] Note that the implementation of a VHDL design into an FPGA is not done by merely transferring data into the device but by a complex configuration process performed by dedicated hardware.

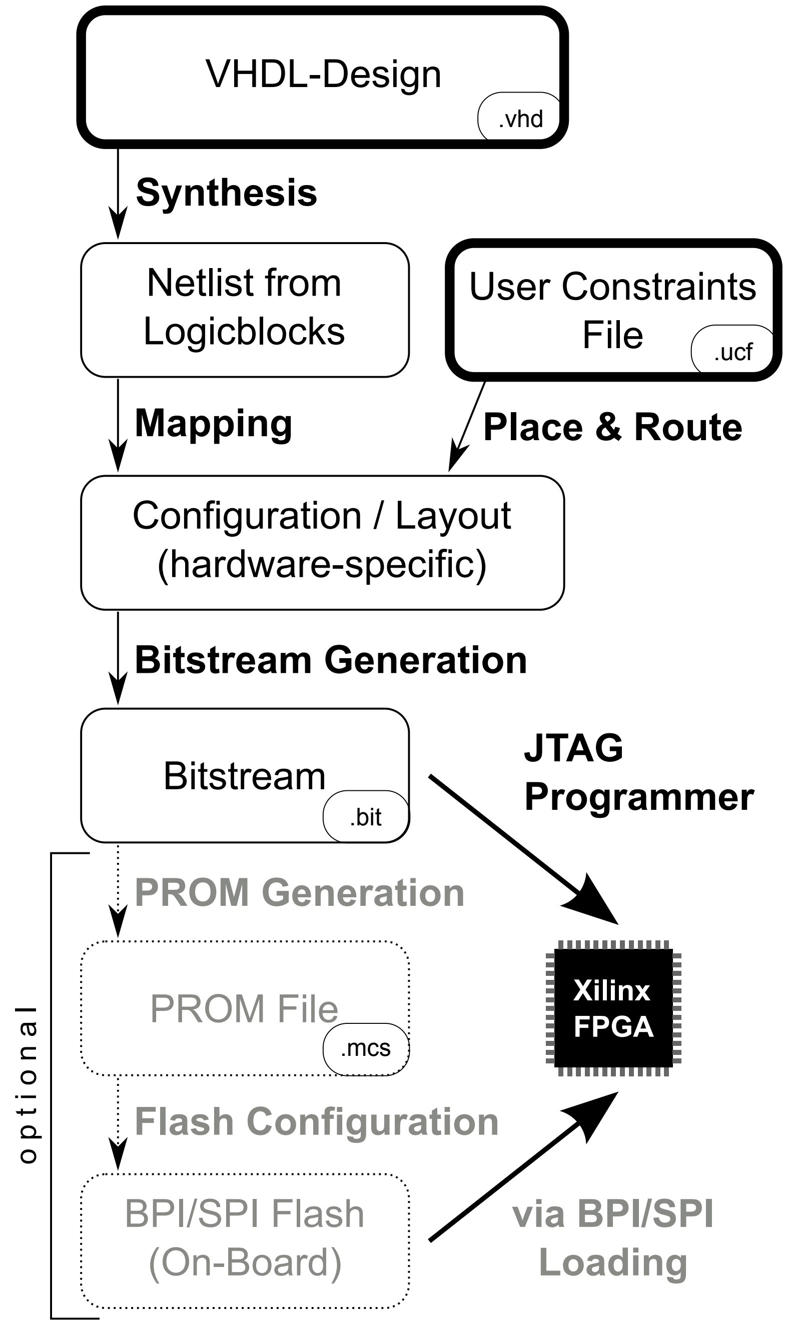


Figure 4.2
Schematic view of the implementation of a VHDL design. The user has to specify at least two files, shown with bold margin. The .vhd file contains the VHDL source code and the .ucf file contains the physical locations of input and output ports of the device.

A problem with the dataflow method is also the low abstraction level. The functionality is coded with simple constructs typically consisting of multiplexers, bit-wise operators and conditional assignments (if-then-else). The overall algorithm might be very difficult to recognize and debug. Yet another issue is simulation time: the assignment of a signal takes approximately 100 times longer than assigning a variable in a VHDL process. This is because the various signal attributes must be updated, and the driving event added to the event queue. With many concurrent statements and processes, a larger proportion of the simulator time will be spent managing signals and scheduling of processes and concurrent statements.

A solution to this situation is to use variables instead of signals in a sequential process and a second, combinatorial process to write an "input" record used in the sequential process with the system clock into a "registered" record. All the entity logic then lies within the sequentially written and executed first process. This technique called the "Two Process Method" was developed by Jiri Gaisler in 1997. The method was adopted by the author of this work for the development of the PBEE of the LAD instrument (see Chapter 6.6) and also used for the interface described in Chapter 6.7.1. This chapter is based on the work of Jiri Gaisler presented in (Gaisler 1996).

4.2.1 Two Processes per Entity

The biggest difference between a program in VHDL and standard programming language such as C is that VHDL allows concurrent statements and processes that are scheduled for execution by events rather than by the order they are written in. While this reflects indeed the dataflow behavior of real hardware, it becomes difficult to understand and analyze when the number of concurrent statements passes some threshold (e.g. 50). On the other hand, analyzing the behavior of programs written in sequential programming languages does not become a problem even if the programs tend to grow, since there is only one thread of control and execution is done sequentially from top to bottom.

In order to improve readability and provide a uniform way to encode the algorithm of a VHDL entity, the Two Process Method uses two processes per entity: one process that contains all combinational (asynchronous) logic, and one process that contains all sequential logic, i.e. registers. Using this structure, the complete algorithm can be coded in sequential (non-concurrent) statements in the combinational process while the sequential process only contains registers, i.e. the state of the entity.

Figure 4.3 shows a block diagram of a two-process entity. Inputs to the entity are denoted **in** and connected to the combinational process. The inputs to the sequential process are denoted **rin** and are driven by the combinational process. In the sequential

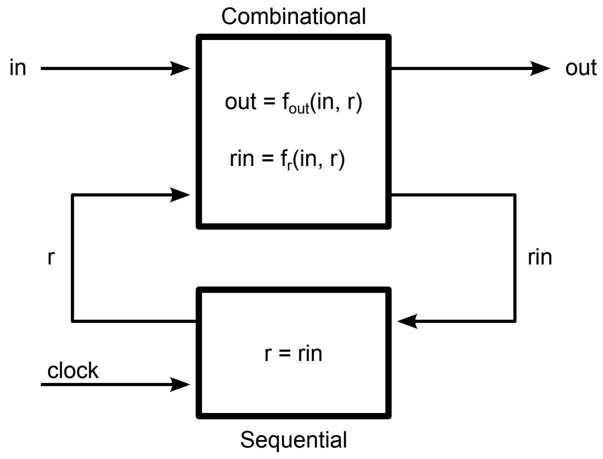


Figure 4.3
This diagram is depicting an entity using the Two Process Method. The inputs (**in**) together with the registered internal record (**r**) are by combinational logic (using variables) producing the intermediate record **rin**. This record (**rin**) is assigned to **r** on the rising/falling edge of the clock. Finally the outputs are generated by combinational logic, but usually only represent a subset of signals from the registered internal record **r**.

process, the inputs **rin** are copied to the outputs **r** on the clock edge. The functionality of the combinational process can be described in two equations:

$$\text{out} = f_{\text{out}}(\text{in}, \text{r}) \qquad \text{rin} = f_r(\text{in}, \text{r})$$

Given that the sequential process only performs a latching of the state vector, the two functions are enough to express the overall functionality of the entity.

4.2.2 Using Records

The port interface list for complex IP blocks can consist of several dozens of signals. Using the standard dataflow method, the signals are not grouped into more complex data types but just listed in the code. The most common data types are scalar types and one-dimensional arrays (buses). Having a port list of several hundreds of signals makes it difficult not only to understand which signals functionally belong together, but also to add and remove signals. Each modification to the interface list has to be made at three separate locations: the entity declaration, the entity's component declaration, and the component instantiation (adding a port map).

By using record types to group associated signals, the port list becomes both shorter and more readable. The signals are grouped according to functionality and direction (in or out, records can't contain both). The record types can be declared in a common global interface package which is imported in each module. Alternatively, the record types can be declared together with the entities component declaration in a component package. This package is then imported into those modules where the component is used. A modification to the interface list using record types corresponds to adding or removing an element in one of the record types. This is done only in one single place, the package, where the record type is declared. Any changes to this package will automatically propagate to the component declaration and the component instantiation (in another entity), avoiding time-consuming and error-prone manual editing.

4.2.3 Clock and Reset Signals

The clock signal is not been included in the record types used for ports. The clock is typically routed from an input pad and through the complete hierarchy of modules. In a synchronous single-clock design, the clock may not be skewed or the function cannot be guaranteed. If the clock was included in a record type, the assignment to the record field would create a delta delay, skewing that part of the clock tree.

Also the reset signal is left out from the record types, much for the same reasons as the clock signal. This reasoning is valid if the reset is asynchronous, it must then be treated as a clock both during routing and timing analysis. A synchronous reset signal can be added to the record types since it behaves like any other non-clock input signal. The two-process methodology can handle both synchronous and asynchronous reset using different coding style. A synchronous reset is treated as any other input signal and used in the combinational process. By placing the reset assignment last in the process, it will have precedence before any other statements.

4.2.4 Summary

The two-process method is a very convenient way to write structured and readable VHDL code that is still suitable for simulation and synthesis. By applying a sequential coding style, the algorithm implemented can be easily identified and the code can be analyzed and maintained more easily. Using sequential statements to code the algorithm also allows for a higher abstraction level.

The custom designed interface used for communication between the LAD's MBEEs and PBEEs was written using the Two Process Method to code a complex state machine. The result is a very efficient interface with error detection and correction mechanisms, that is still easy to understand and comfortable to use. The event handling, sorting, and general functionality of the PBEE was also coded using this method, as was some of the code written by Florian Jetter for the LOFT WFM and by Pascal Uter for the LOFT MBEE.

4.3 LEON3

The LEON3 is a VHDL-model of a 32-bit microprocessor based on the SPARC V8 architecture.[3] It is being actively developed by Aeroflex Gaisler AB and licensed under the free GNU GPL license. The LEON3 is a SoC design incorporating typical components of a microcontroller, such as an AMBA AHB bus, memory controller (PROM, SRAM, SDRAM), USB interfaces, a JTAG interface, an Ethernet controller, and several peripheral components (VGA, PS/2, RS232, general purpose I/O ports). See Figure 4.4 for an extended overview.

The LEON3 is also available in a fault-tolerant version as LEON3FT. This version was especially designed for use in a radiation environment and is used in space-based observatories such as the ESA Planck mission. In fact the first version of the LEON3, the LEON1 VHDL Design was developed 1997 as part of an ESA research project.

Aeroflex Gaisler is providing the LEON3 in several configurations, ready to be implemented on one of several commonly available prototyping boards (e.g. XILINX, Altera). The final design of the LEON3 can be adjusted by using a couple of scripts provided with the source code to modify and enable individual components (e.g. cache size, RAM, interfaces).

4.3.1 LEON3FT

The LEON3FT is a fault-tolerant version of the standard LEON3 processor. In a real space environment certain functionality has to be included into any kind of processor to handle the difficult environmental conditions by preventing otherwise fatal errors or at least by implementing mechanisms suited to detect and handle such errors. The LEON3FT therefore includes functionality to detect and correct errors in all on-chip RAM memories. It supports most of the functionality of the standard LEON3 processor but adds the following important features:

- Register file error-correction of up to 4 errors per 32-bit word
- Cache memory error-correction of up to 4 errors per tag or 32-bit word
- Autonomous and software transparent error handling
- No timing or performance impact due to error detection and correction

[3] **S**calable **P**rocessor **ARC**hitecture (SPARC) is a processor architecture developed in 1986 by Sun Microsystems. Version V8 was released in 1990. The SPARC architecture is based on the Reduced Instruction Set Computing (RISC) design, a very popular processor design that is e.g. used as a reference frame for the well known SPEC CPU95 and SPEC CPU2000 benchmarks.

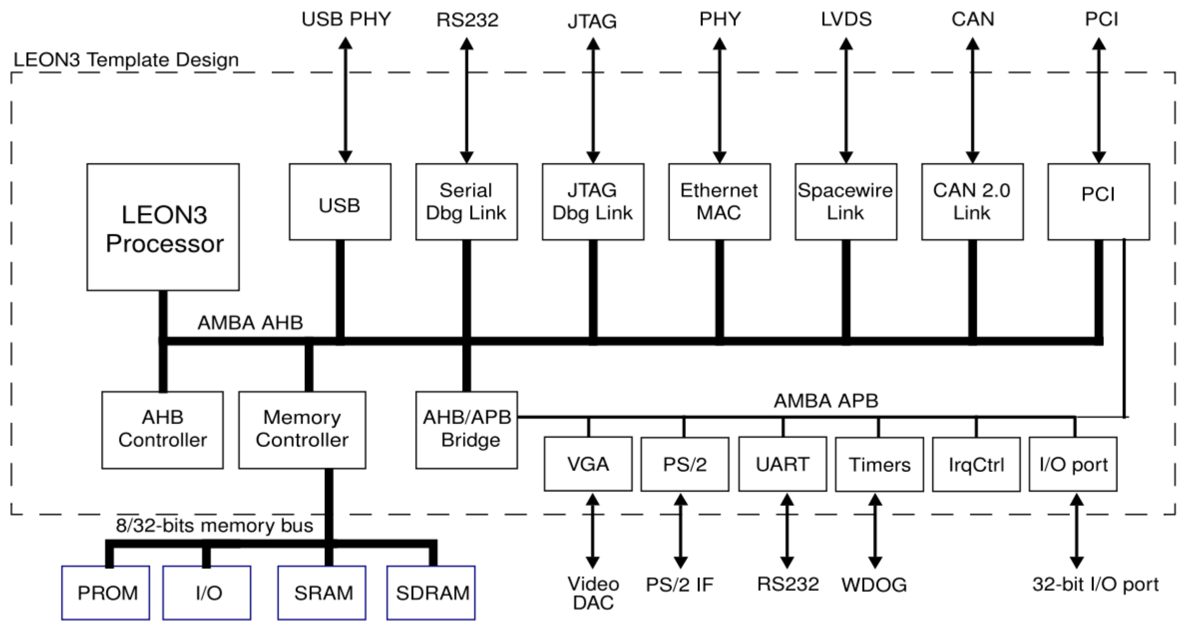


Figure 4.4
Typical structure of a LEON3 design with central AMBA AHB bus. The individual IP Cores are contained in the GRLIB IP library made available by Aeroflex Gaisler AB. Source: Aeroflex Gaisler (2012).

The fault-tolerance in LEON3FT is implemented using ECC coding of all on-chip RAM blocks (Aeroflex Gaisler 2013). The ECC codes are adapted to the type of RAM blocks that are available for a given target technology, and to the type of data that is stored in the RAM blocks. The general scheme is to be able to detect and correct up to four errors per 32-bit RAM word. In RAM blocks where the data is mirrored in a secondary memory area (e.g. cache memories), the ECC codes are tuned for error-detection only. A correction cycle consists then of reloading the faulty data from the mirror location. In the cache memories, this equals to an invalidation of the faulty cache line and a cache line reload from main memory.

In RAM blocks where no secondary copy of the data is available (e.g. register files), the ECC codes are tuned for both error-detection and correction. The focus is placed on fast encoding/decoding times rather than minimizing the number of ECC bits. This approach ensures that the FT logic does not affect the timing and performance of the processor, and that LEON3FT can reach the same maximum frequency as the standard non-FT LEON3. The ECC encoding/decoding is done in the LEON3FT pipeline in parallel with normal operation, and a correction cycle is fully transparent to the software without affecting the instruction timing (Aeroflex Gaisler 2013).

4.3.2 LEON4

The LEON4 processor is the fourth generation LEON processor that was released in January 2010. While the LEON3 IP Core is released under the term of the GNU General Public License, the LEON4 was released under a non-free proprietary license.

The LEON4 processor has all the features of a LEON3 processor, including a SPARC V8 compliant architecture with a seven-stage pipeline for fast and energy-efficient execution, configurable Level 1 (L1) instruction and data caches, configurable number of register windows, and IEEE-754/IEEE-1754 compatible floating-point units (Själander, Habinc, and Gaisler 2009). This makes LEON4 software compatible with code written for the LEON3 processor. For improved performance the LEON4 processor includes a branch prediction unit, a wide (64 or 128-bit) processor bus, support for configurable MMU page size, and a Level 2 (L2) cache (Själander, Habinc, and Gaisler 2009).

The branch prediction unit included in the LEON4 allows the pipeline to continue executing instructions when a branch is encountered in the instruction stream. In previous generations of the LEON processor the pipeline is stalled for one or two clock cycles if the branch condition depends on the result of any of the two instructions executed before the branch. With the branch prediction unit it is not necessary to wait for the branch condition to be evaluated before fetching new instructions. There is no penalty for miss-speculations. The speculatively fetched instructions are simply flushed from the pipeline and new instructions are fetched from the correct branch target. This micro-architectural improvement makes the LEON4 processor on average 35% faster than a LEON3 processor operating at the same clock frequency (Själander, Habinc, and Gaisler 2009).

4.4 Interfaces

4.4.1 SpaceWire

One commonly used type of network for on-board communication is SpaceWire. This comprehensive summary over the SpaceWire standard is based on the work by Parkes and Rosello (Parkes and Rosello 2003).

SpaceWire is designed to connect processing hardware, i.e. FPGAs, large solid-state memories, and the downlink telemetry subsystem providing an integrated on-board, data-handling network. SpaceWire links are serial, high-speed (2 Mbit/sec to 400 Mbit/sec), bi-directional, full-duplex, point-to-point data links which connect SpaceWire equipment. While often dedicated SpaceWire hardware is used, it is possible to implement SpaceWire core in an FPGA. Application information is sent along a SpaceWire link in discrete packets.

SpaceWire is defined in the European Cooperation for Space Standardization ECSS-E50-12A standard. SpaceWire is based on the "DS-DE" part of the IEEE-1355 standard (IEEE 1996) combined with the TIA/EIA-644 Low Voltage Differential Signaling (LVDS) standard (TIA 1996). Several problems with IEEE-1355 have been solved in the SpaceWire standard and connectors and cables suitable for space application are defined. The SpaceWire standard is divided into several protocol levels:

- **Physical Level** which provides connectors, cables and EMC specifications.
- **Signal Level** which defines signal encoding, voltage levels, noise margins and data rates.
- **Character Level** which specifies the data and control characters used to manage the flow of data across a link.
- **Exchange Level** which covers the protocol for link initialization, flow control, fault detection and link restart.
- **Packet Level** which details how a message is delivered from a source node to a destination node.

The physical level of the SpaceWire standard covers cables, connectors, cable assemblies and printed circuit board tracks. SpaceWire was developed to meet the EMC specifications of a typical spacecraft.

Cables

The SpaceWire cable comprises four twisted pair wires with a separate shield around each twisted pair and an overall shield. To achieve a high data signaling rate with SpaceWire over distances up to 10 m a cable with the following characteristics is used:

- Characteristic impedance of 100 Ω differential impedance, which is matched to the line termination impedance.
- Low signal-signal skew between each signal in a differential pair and between Data and Strobe pairs.
- Low signal attenuation.
- Low cross-talk.
- Good EMC performance.

Connectors

The SpaceWire connector has eight signal contacts plus a screen termination contact. A nine pin micro-miniature D-type is specified as the SpaceWire connector. This type of connector is available in a space qualified version.

Cable Assemblies

SpaceWire cable assemblies are made from SpaceWire cable of up to 10 m length terminated at each end by nine-pin micro-miniature D-type plugs. A typical assembly is shown in Figure 4.5.

PCB Tracks

SpaceWire can also be run over printed circuit boards (PCBs) including backplanes. The PCB tracks must have 100 Ω differential impedance.

Figure 4.5
A typical SpaceWire cable assembly. Source: Axon' Cable, France.

Data Encoding

SpaceWire uses Data-Strobe (DS) encoding. This is a coding scheme which encodes the transmission clock with the data into Data and Strobe signals so that the clock can be recovered by simply XORing the Data and Strobe lines together. The data values are transmitted directly and the strobe signal changes state whenever the data remains constant from one data bit interval to the next.

4.4.2 CAN

While SpaceWire is used for communication between larger processing units, smaller units, e.g. environmental controls, housekeeping boards, have to be included into the communication network as well. Since SpaceWire, while well-suited for the transfer of larger amounts of data, needs dedicated hardware, and therefore adds to the weight and power budget it is often oversized for the relaxed requirements on the communication between sub-units. Another downside of using SpaceWire equipment is the physical concentration of I/O interfaces on the processing units which results in a large amount of wiring and connectors with impact on mission reliability, testing, and validation.

Instead a serial bus is used for all non-SpaceWire communications. Standards used for space missions are mainly based on MIL-STD-1553B and RS-485 which provide a communication paradigm based on Master/Slave or Client/Server configuration (ILC 2003). These standards are originally aimed at the specific requirements of military aircraft and still require power and mass that might be further reduced by simplifying the complexity of the bus system. A very promising approach to adopt a simple bus for the use on space mission is the work done by ESA departments on the vehicle bus CAN.

CAN (Controller Area Network) is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer. Its a message based protocol, designed specifically for automotive applications but also used in other areas such as industrial automation and medical equipment. On a space mission CAN would allow scalable bus architectures with reduced power consumption, wiring harness, and engineering effort and would also increase functionality and reliability. CAN studies, development and validation activities performed at ESTEC have recently demonstrated that the CAN bus is an effective solution for the implementation of a low power, reliable platform and payload serial data line for scientific missions. The results are taken into account by ESA and drive the standardization process which will result in ECSS-E-ST-50-15C and will contain all services and recommendations to cover the peculiarities of space applications. Several other initiatives have been planned by ESA to complete in 2014 the standardization of the CAN bus and in particular of a space qualified ISO11898 transceiver (Caramia, Bolognino, and Furano 2013).

Part III

Implementation

Chapter 5

IXO

The International X-ray Observatory (IXO) (White, Parmar, and Kunieda 2009; ESA 2011b) was developed as an an L-class ESA mission devised to address important astrophysical questions regarding the evolution and content of the Universe. It was designed with superb capabilities in X-ray spectroscopy, imaging, timing, and polarimetry, enabling IXO to reveal the entire history of the Hot Universe, from the Cosmic "Dark Ages" to the present day. Figure 5.1 shows an artist's view of the proposed observatory design. In the selection process for the next L-class mission that ended in 2013, the JUICE[1] mission was chosen for implementation. Chapter 5.7 describes how IXO evolved since and became ATHENA. The next chapters however describe IXO and its instruments in their development state during the HTRS studies in 2010/11.

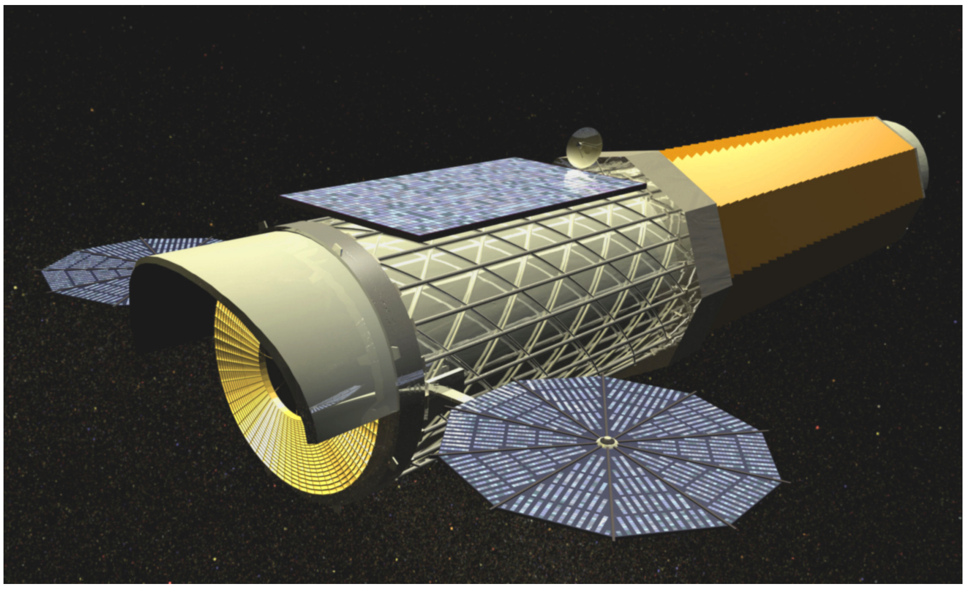


Figure 5.1
Artist's view of the International X-ray Observatory. Source: NASA.

[1] **JU**piter **IC**y Moons **E**xplorer

5.1 Mission Overview

The 25 m long[2] and 3.5 m wide International X-ray Observatory, weighing about 6400 kg, is a focusing telescope with a total effective area of 2.5 m^2 at 1.25 keV, a focal length of 20 m and 6 scientific instruments, of which five are residing in the focal plane (see Figure 5.2). Detailed performance requirements and their scientific drivers are presented in Table 5.1, the scientific themes are presented in Chapter 5.4.

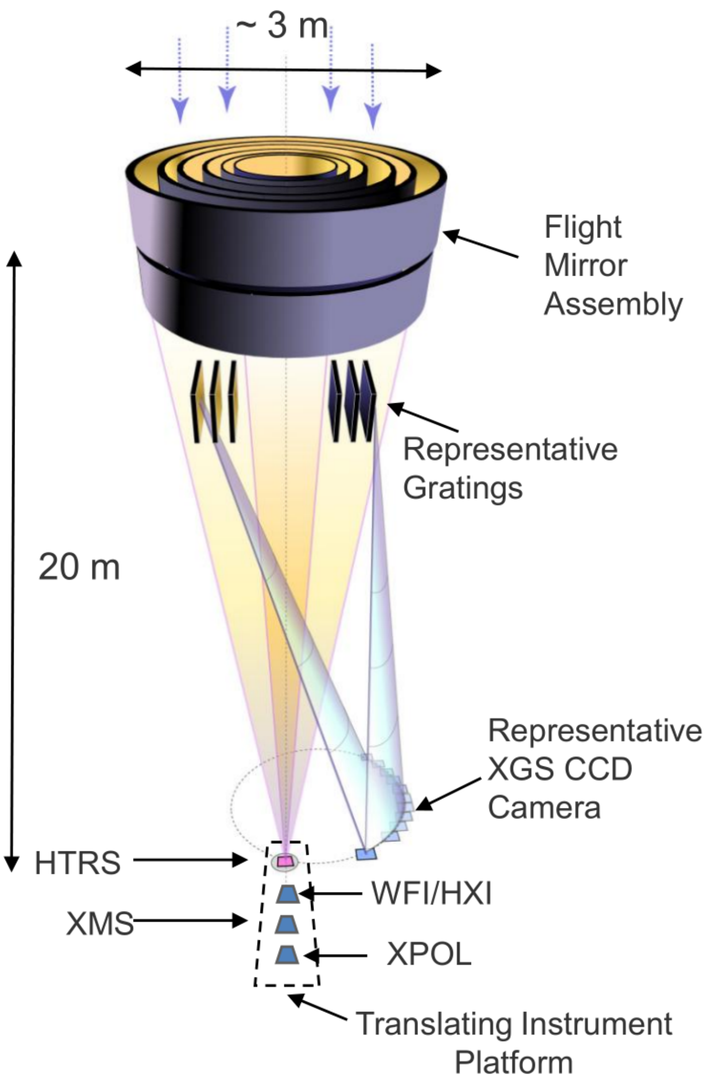


Figure 5.2
Overview of IXO's telescope setup and the focal plane assembly. Note that the XGS instrument is not located within the focal plane. Source: White, Parmar, and Kunieda (2009).

[2] The stowed spacecraft is 11 m long and designed to extend to 25 m once deployed in its L2 orbit.

Table 5.1
IXO engineering requirements and driving science themes.
Specs reflect the design state from the IXO Assessment Study Report (ESA 2011b).

Item	IXO Requirement	IXO Science Theme
Effective Area	$2.50\,\mathrm{m}^2$ @ 1.25 keV $0.65\,\mathrm{m}^2$ @ 6 keV $150\,\mathrm{cm}^2$ @ 30 keV	Black Hole Evolution, Large Scale Structure, Strong Gravity, EOS, Cosmic Feedback / Acceleration
Energy Resolution	2.5 eV @ 6 keV ($2' \times 2'$) 10 eV @ 6 keV ($5' \times 5'$) 150 eV @ 6 keV ($18' \times 18'$) $E/\Delta E = 3000$ (0.3 - 1 keV) with an area of $1000\,\mathrm{cm}^2$	Black Hole Evolution Cosmic Feedback Large Scale Structure Missing Baryons
Angular Resolution	5 arcsec HPD (0.1 - 7 keV) 30 arcsec HPD (7 - 40 keV)	Large Scale Structure, Cosmic Feedback, Black Hole Evolution, Missing Baryons
Count Rate	1 Crab with > 90 % throughput $\Delta E < 200$ eV @ 6 keV (0.3 - 15 keV)	Strong Gravity Equation Of State
Absolute Time Accuracy	100 µs	Neutron Star Studies
Dead-Time	< 1 % @ 1 Crab	Neutron Star Studies
Polarimetry	1 % MDP @ 1 mCrab $3\,\sigma$ @ 2 - 6 keV	AGN Geometry Strong Gravity
Astrometry	1.5 arcsec @ $3\,\sigma$	Black Hole Evolution

IXO was to be launched in 2022 into an L2 halo orbit with a diameter of 1.5 million km as shown in Figure 5.3. This orbit would provide the spacecraft with a stable thermal environment and with very good sky visibility. The launch vehicle can be either the Ariane 5 or the Atlas V/551, both are capable of lifting a 6.5 ton observatory into a direct transfer trajectory to L2. The nominal lifetime of the spacecraft was designed to be 5 years with a possible 5 year extension.

The spacecraft design allows for a large pointing excursion around the spacecraft-Earth-Sun direction (S/C pitch, +/- 180 deg) and modest rotations in yaw (around the S/C's vertical Z-axis, +/- 20 deg) and roll (+/- 10 deg).

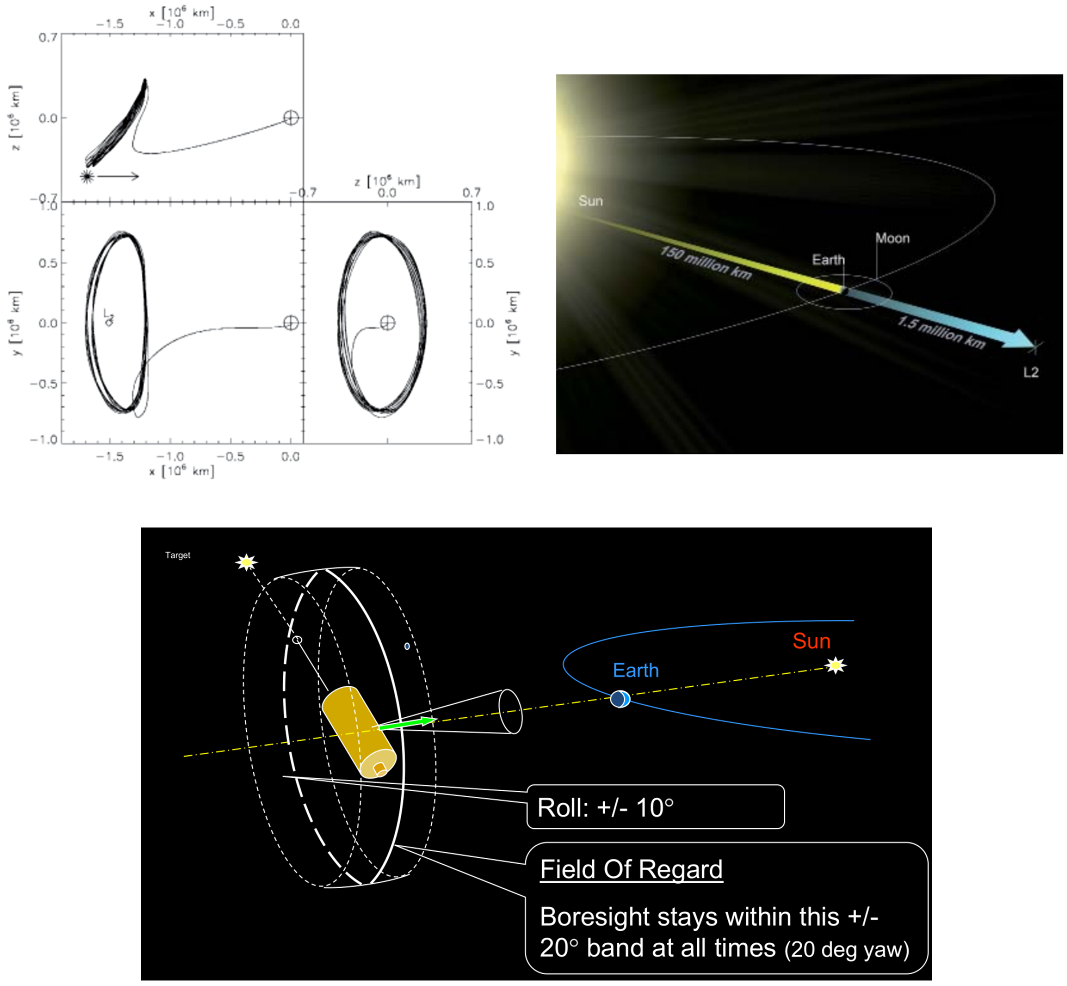


Figure 5.3
Top: IXO in a large halo orbit around the second Lagrangian point of the Earth-Sun system. Source: ESA (2011b). Bottom: Orientation and operation of the spacecraft with regard to the direction to the Sun. Source: White, Parmar, and Kunieda (2009).

5.2 Optics and Payload

The IXO mission required a large effective area of 2.5 m^2 at 1.25 keV and a high angular resolution of 5 arcsec below 7 keV (see Chapter 5.4 for more details). Two technologies have been developed by ESA and NASA to enable the construction of the appropriate optics.

The baseline technology silicon pore optics (SPO) was developed by ESA and uses the stacking of ribbed and slightly bent silicon plates to form monolithic mirror modules. These modules are robotically assembled to create very robust and lightweight Wolter I-type optics. Figure 5.4 shows a structural overview of the silicon pore optics assembly.

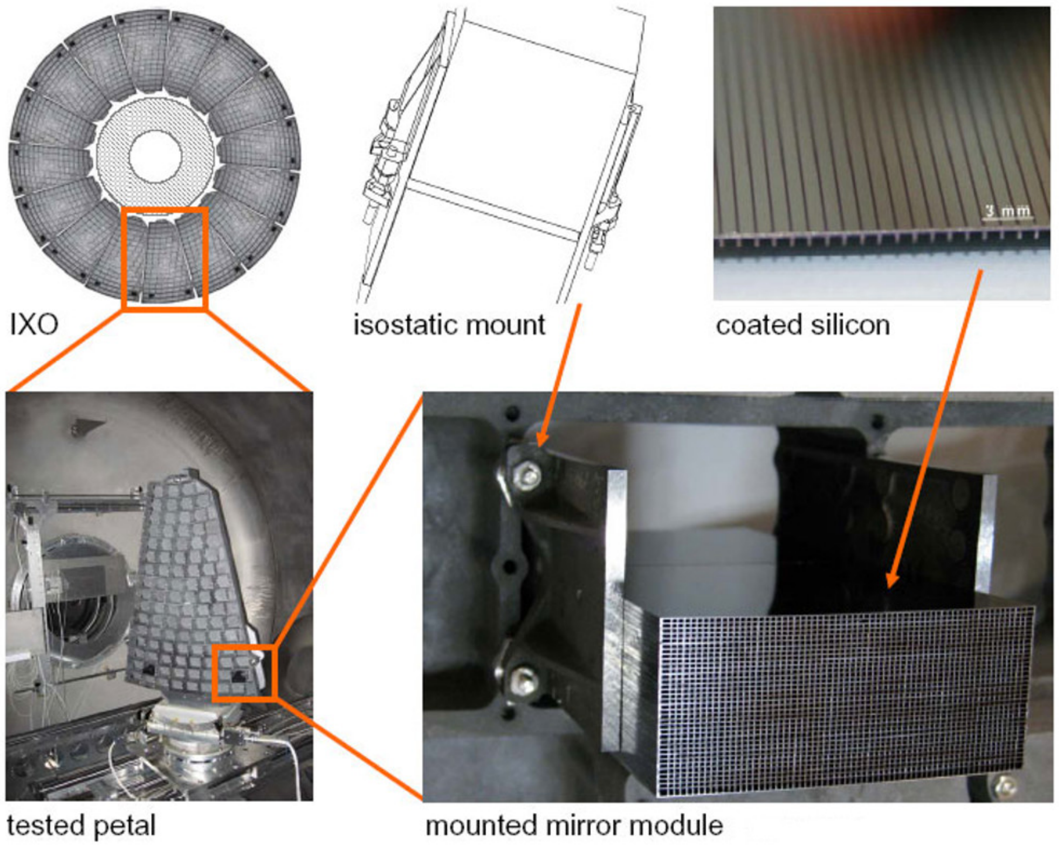


Figure 5.4
Mirror plates made from ribbed coated silicon are stacked together by a stacking robot (not shown here) to form a monolithic mirror stack. Two mirror stacks are assembled into a mirror module and mounted in a mirror petal. 16 mirror petals are finally assembled to form the IXO optics assembly. Source: Bavdaz et al. (2012).

The backup technology segmented glass optics (SGO) was developed by NASA. The SGO assembly uses nested grazing incidence glass mirror segments, assembled into mirror modules that are aligned and mounted onto a wheel-like primary structure. The majority of the optical system is dedicated to the low energy (up to 15 keV), high-resolution response. The high-energy response (10 to 40 keV) is provided by a single mirror module located at the center of the mirror assembly.

To achieve the performance specified in Table 5.1 on page 55 IXO was designed to utilize a total of six scientific instruments. Five of these instruments were located on the movable instrument platform in the focal plane of the telescope. A sixth instrument was placed in proximity and a small portion of light is reflected here via dedicated X-ray gratings placed in the light path. The six instruments selected for IXO were:

- An X-ray imaging Microcalorimeter Spectrometer (**XMS**) that covers the 0.3 - 12 keV energy range with unprecedented energy resolution, a 5x5 arcmin FoV, and relatively low count rate capability.
- A Wide Field Imager (**WFI**) covering the 0.1 - 15 keV energy range with a large, 18 arcmin diameter FoV, excellent spatial resolution and efficiency, good energy resolution, and adequate count rate capability.
- A confocal Hard X-ray Imager (**HXI**) that covers an 8 arcmin FoV with excellent spatial resolution and efficiency in the 10 - 40 keV energy range, in combination with good energy resolution and count rate performance.
- A non-imaging High Time Resolution Spectrometer (**HTRS**) that covers the 0.3 - 15 keV energy range with good energy resolution, and ultra-high count rate capability.
- An imaging X-ray Polarimeter (**XPOL**) with a modest FoV, modest energy resolution, and excellent sensitivity to polarization in the 2 - 10 keV energy range.
- An X-ray Grating Spectrometer (**XGS**) for the 0.3 - 1 keV range with 1000 cm^2 effective area and a resolving power of > 3000.

These instruments account for 1100 kg of the spacecraft's mass (6500 kg total) and for 1800 W of the power budget (5700 W total). The technological readiness level of the instruments is very good (TRL > 5) with the exception of the XMS cryogenic chain and the movable instrument platform itself. The IXO development plan took this into account and an alternative design was developed and tested for the pre-cooling and final stage of the microcalorimeter.

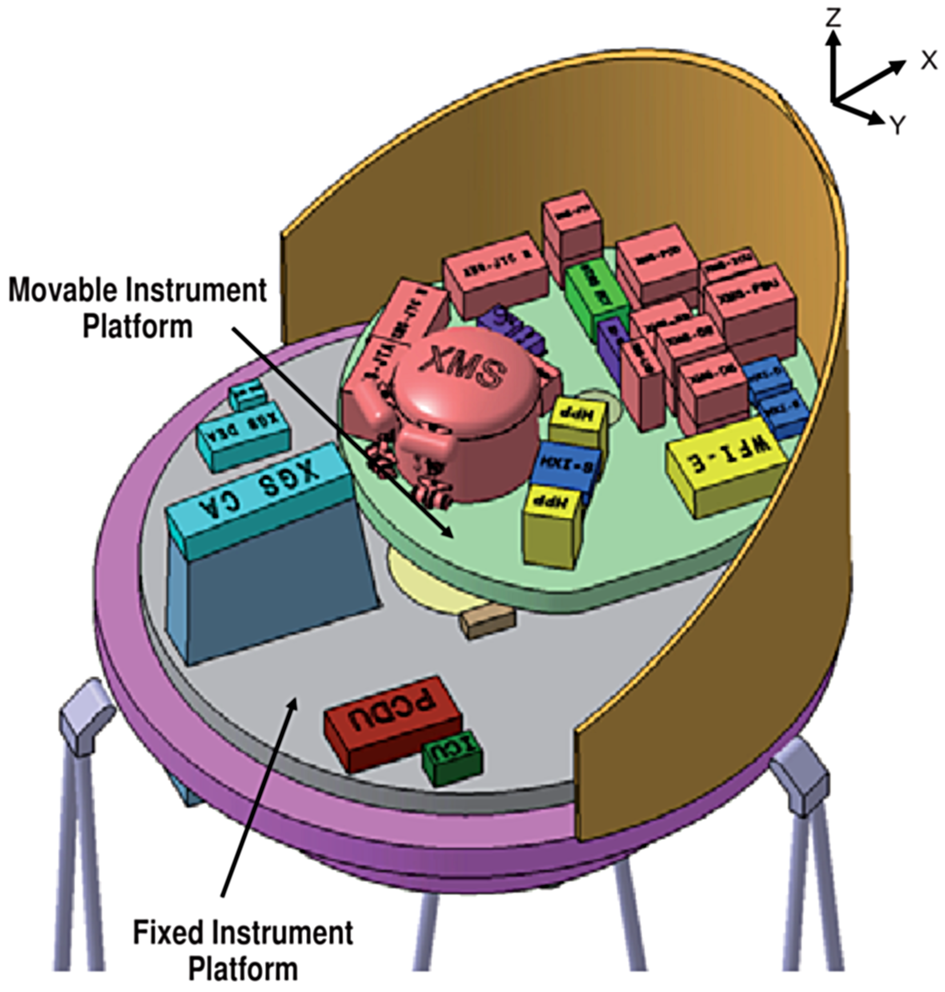
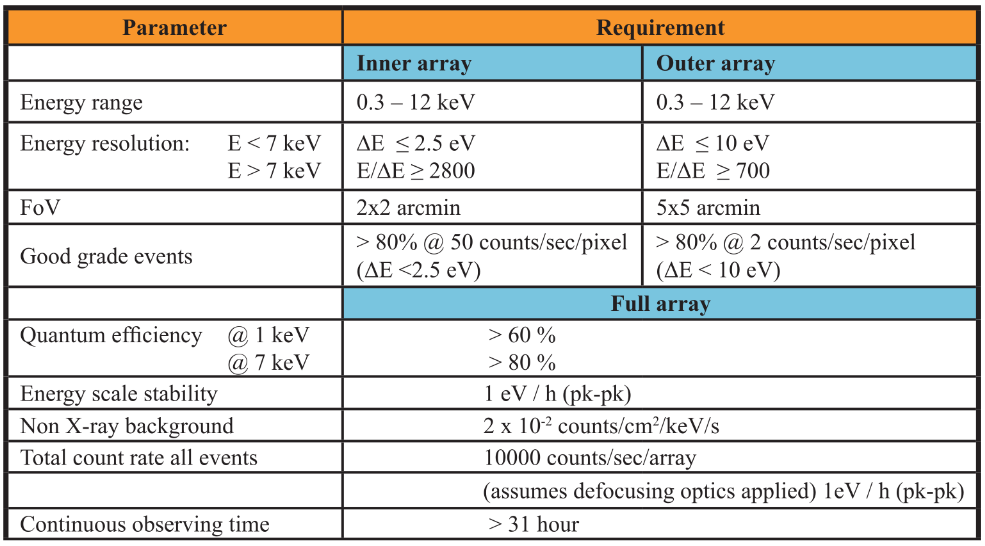
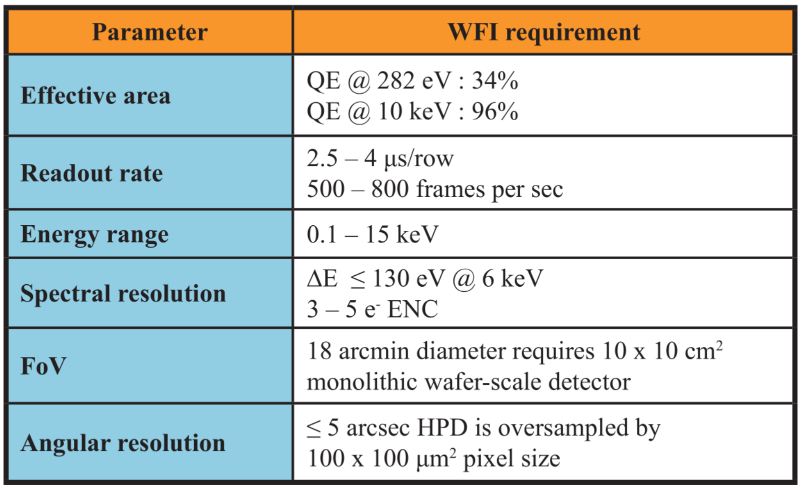


Figure 5.5
This drawing shows one possible layout of the focal plane instrument module. The position of the XGS is fixed and the instrument platform and not located in the focal plane. Light is reflected to the XGS via dedicated gratings (see Figure 5.2). The other five instruments are located on the movable instrument platform (green). Note that in this drawing the platform is shown as rotating, earlier designs used a linear moving platform. Source: ESA (2011b).



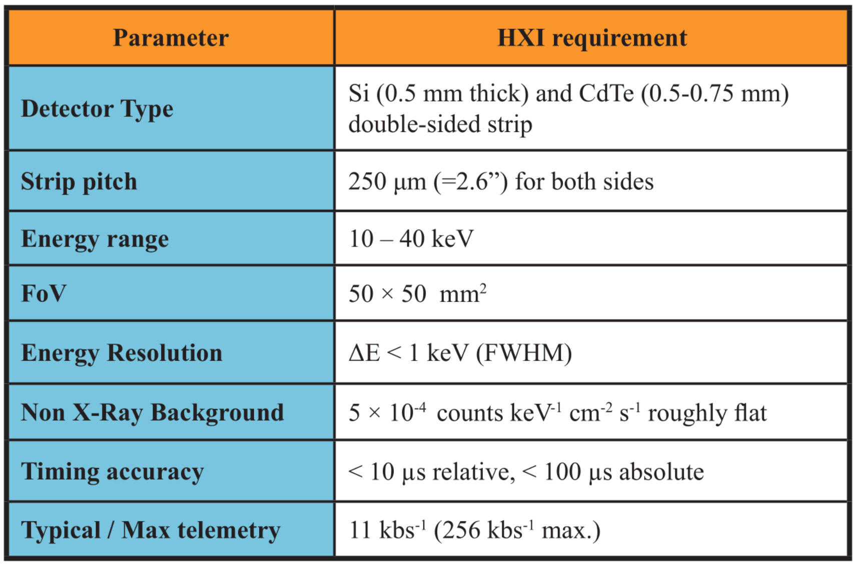
Parameter	Requirement	
	Inner array	**Outer array**
Energy range	0.3 – 12 keV	0.3 – 12 keV
Energy resolution: $E < 7$ keV $E > 7$ keV	$\Delta E \leq 2.5$ eV $E/\Delta E \geq 2800$	$\Delta E \leq 10$ eV $E/\Delta E \geq 700$
FoV	2x2 arcmin	5x5 arcmin
Good grade events	> 80% @ 50 counts/sec/pixel (ΔE <2.5 eV)	> 80% @ 2 counts/sec/pixel (ΔE < 10 eV)
	Full array	
Quantum efficiency @ 1 keV @ 7 keV	> 60 % > 80 %	
Energy scale stability	1 eV / h (pk-pk)	
Non X-ray background	2 x 10^{-2} counts/cm^2/keV/s	
Total count rate all events	10000 counts/sec/array	
	(assumes defocusing optics applied) 1eV / h (pk-pk)	
Continuous observing time	> 31 hour	

Figure 5.6
XMS instrument specifications. Source: ESA (2011b).



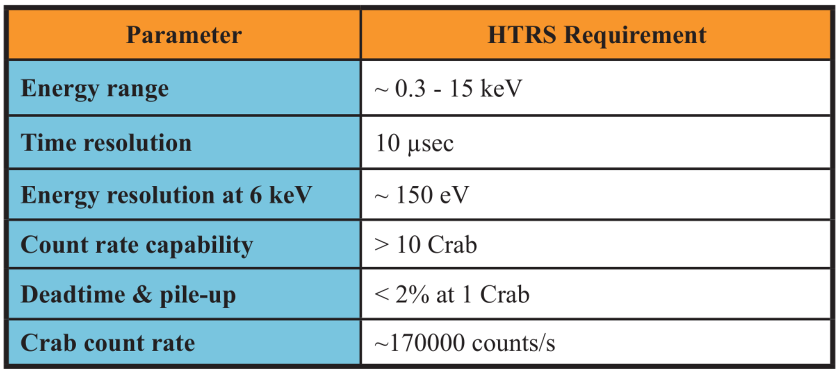
Parameter	WFI requirement
Effective area	QE @ 282 eV : 34% QE @ 10 keV : 96%
Readout rate	2.5 – 4 µs/row 500 – 800 frames per sec
Energy range	0.1 – 15 keV
Spectral resolution	$\Delta E \leq 130$ eV @ 6 keV 3 – 5 e^- ENC
FoV	18 arcmin diameter requires 10 x 10 cm^2 monolithic wafer-scale detector
Angular resolution	$\leq$ 5 arcsec HPD is oversampled by 100 x 100 µm^2 pixel size

Figure 5.7
WFI instrument specifications. Source: ESA (2011b).



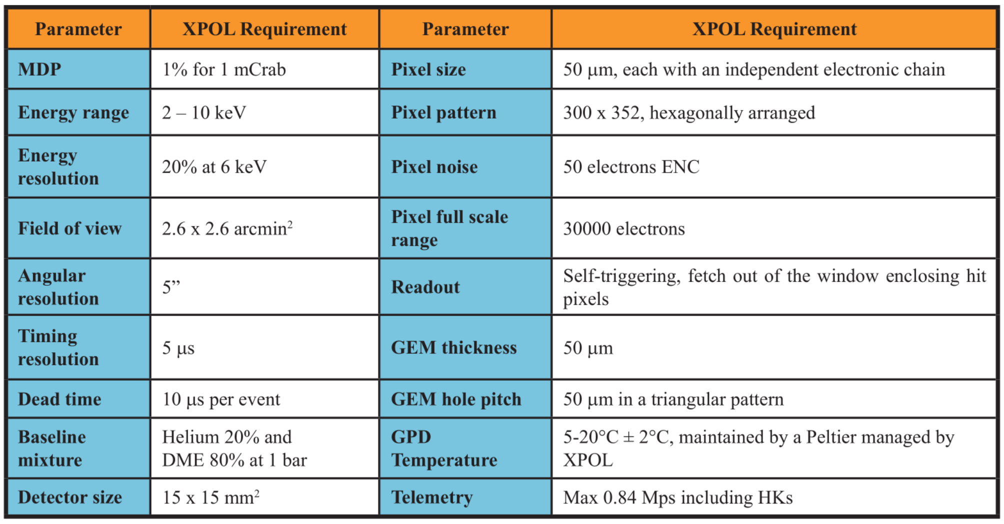
Parameter	HXI requirement
Detector Type	Si (0.5 mm thick) and CdTe (0.5-0.75 mm) double-sided strip
Strip pitch	250 μm (=2.6”) for both sides
Energy range	10 – 40 keV
FoV	50×50 mm^2
Energy Resolution	$\Delta E < 1$ keV (FWHM)
Non X-Ray Background	5×10^{-4} counts keV^{-1} cm^{-2} s^{-1} roughly flat
Timing accuracy	< 10 μs relative, < 100 μs absolute
Typical / Max telemetry	11 kbs^{-1} (256 kbs^{-1} max.)

Figure 5.8
HXI instrument specifications. Source: ESA (2011b).



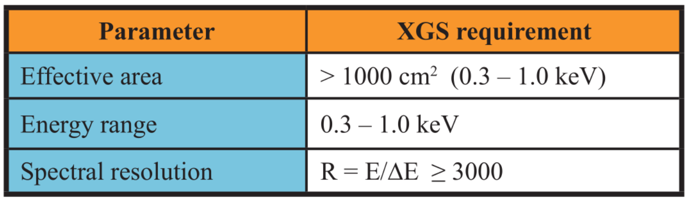
Parameter	HTRS Requirement
Energy range	~ 0.3 - 15 keV
Time resolution	10 μsec
Energy resolution at 6 keV	~ 150 eV
Count rate capability	> 10 Crab
Deadtime & pile-up	< 2% at 1 Crab
Crab count rate	~170000 counts/s

Figure 5.9
HTRS instrument specifications. Source: ESA (2011b).



Parameter	XPOL Requirement	Parameter	XPOL Requirement
MDP	1% for 1 mCrab	**Pixel size**	50 μm, each with an independent electronic chain
Energy range	2 – 10 keV	**Pixel pattern**	300 x 352, hexagonally arranged
Energy resolution	20% at 6 keV	**Pixel noise**	50 electrons ENC
Field of view	2.6 x 2.6 arcmin^2	**Pixel full scale range**	30000 electrons
Angular resolution	5”	**Readout**	Self-triggering, fetch out of the window enclosing hit pixels
Timing resolution	5 μs	**GEM thickness**	50 μm
Dead time	10 μs per event	**GEM hole pitch**	50 μm in a triangular pattern
Baseline mixture	Helium 20% and DME 80% at 1 bar	**GPD Temperature**	5-20°C ± 2°C, maintained by a Peltier managed by XPOL
Detector size	15 x 15 mm^2	**Telemetry**	Max 0.84 Mps including HKs

Figure 5.10
XPOL instrument specifications. Source: ESA (2011b).



Parameter	XGS requirement
Effective area	> 1000 cm^2 (0.3 – 1.0 keV)
Energy range	0.3 – 1.0 keV
Spectral resolution	R = E/ΔE ≥ 3000

Figure 5.11
XGS instrument specifications. Source: ESA (2011b).

5.3 Programmatic Background

In 2005 ESA presented its current long-term program "Cosmic Vision 2015-2025" (Bignami et al. 2005). It is the successor to the previous definition of important questions to be addressed by space- and ground-based telescopes and observatories "Horizon 2000" (Longdon 1984). In the Cosmic Vision 2015-2025 Science Objectives ESA defines four main questions that are considered the most important themes of astronomical research across Europe:

1. What are the conditions for planet formation and the emergence of life?
2. How does the Solar System work?
3. What are the fundamental physical laws of the Universe?
4. How did the Universe originate and what is it made of?

In 2007, three large L-class missions, EJSM-Laplace (ESA 2011a), IXO (ESA 2011b) and LISA (ESA 2011c), were selected for an Assessment Phase study following a Call for Missions for the first L-class mission opportunity in the Cosmic Vision 2015-2025 plan. IXO in particular was selected because it attempted to provide insights into the following themes which are based on the questions in chapters three and four of the Cosmic Vision document (Bignami et al. 2005):

- **The Evolving Violent Universe**, by finding massive black holes growing in the centers of galaxies, from early times to the present, and understanding how they influence the formation and growth of the host galaxy through feedback processes.
- **The Universe taking shape**, by studying how the baryonic component of the Universe formed large-scale structures and by finding the large fraction of baryons that are still missing; and understanding how and when the Universe was chemically enriched by supernovae.
- **Matter Under Extreme Conditions**, by revealing how matter behaves in very strong gravity, such as occurs around black holes and compact objects, where General Relativity predicts many effects; and how matter behaves at densities higher than in atomic nuclei in the interiors of neutron stars.

To make the required measurements possible, IXO employed optics with 10 times more collecting area at 1 keV than any previous X-ray observatory. Furthermore, the focal plane instruments would have delivered a 100-fold increase in effective area for high-resolution spectroscopy, deep spectral imaging over a wide field of view, high polarimetric sensitivity, microsecond spectroscopic timing, and high count rate capability (ESA 2011b).

5.4 IXO Scientific Requirements

5.4.1 Co-Evolution of Galaxies and their Supermassive Black Holes

The first SMBH

Find young growing massive black holes at the dawn of the Universe by conducting multi-tiered surveys, in conjunction with observations at longer wavelengths. Test the growth mode of SMBH by measuring their spin. To make significant progress, deep X-ray survey capabilities are needed to chart the formation of typical SMBHs at $z > 6$ (when the Universe was $< 5\%$ of its current age), and their subsequent growth over cosmic time.
Instrument: **WFI**

Obscured growth of SMBH

Find and characterize SMBH growing in an obscured phase at $z \sim 1 - 3$, including Compton-thick AGN, and measure their energetics.
Instruments: **WFI, HXI, XMS**

Cosmic feedback from SMBH

Measure feedback from growing SMBH in galaxies, groups and clusters, by relating AGN activity to star formation at high-z, and measure gas motions in galaxies and clusters induced by AGN.
Instruments: **XMS, WFI**

5.4.2 Large-Scale Structure and the Creation of Chemical Elements

Missing baryons and the Intergalactic Medium

Find the remaining missing baryons in the current Universe, likely in a Warm-Hot Intergalactic Medium phase, by detecting hundreds of intervening absorption systems towards bright background targets.
Instruments: **XGS, XMS**

Cluster Physics and Evolution

Find out how gas dynamically evolves in the cluster dark matter potentials, by measuring gas motions and turbulence. Determine the physical processes behind the production of cosmic rays in clusters. Also find when and how the excess entropy was generated in clusters by studying their precursors at early epochs.
Instruments: **XMS, HXI**

Galaxy Cluster Cosmology

Provide an independent measurement of Dark Energy density and its equation of state by counting clusters at various epochs and measuring their gas fraction.
Instruments: **WFI**

Chemical Evolution along Cosmic Time

Measure chemical abundances of the various families of elements in intracluster gas, find at which epoch they were dispersed and derive what is the main production mechanism.
Instruments: **XMS**

5.4.3 Matter under Extreme Conditions

Strong Gravity and Accretion Physics

Measure black hole spin of stellar and supermassive black holes via time-averaged spectroscopy, reverberation mapping, timing and polarimetry. Probe General Relativity in the strong field regime by mapping the emission of the innermost regions of accretion disks. Measure the kinetic energy of outflows produced by matter falling onto black holes.
Instruments: **WFI, HXI, HTRS, XPOL, XMS**

Neutron Star Equation of State

Measure mass and radius of tens of neutron stars via a number of timing and spectroscopic techniques, and constrain the equation of state at supra-nuclear density. Detect vacuum polarization effects in highly magnetized neutron stars.
Instruments: **HTRS, XMS, XPOL**

5.4.4 Life Cycles of Matter and Energy in the Universe

Supernovae Explosion Mechanisms

Probe the supernova explosion material and its environment by measuring temperatures, velocities, turbulences in Supernova Remnants.
Instruments: **XMS**

Supernovae Nucleosynthesis

Measure nucleosynthesis products for the various types/subtypes of Supernovae by detecting weak emission lines from radioactive elements in Supernova Remnants.
Instruments: **XMS HXI**

Cosmic Ray Acceleration in Supernova Remnants

Measure hard X-ray synchrotron emission due to fluctuating magnetic fields.
Instruments: **WFI, HXI, XPOL**

Characterizing the Interstellar Medium in the Galaxy

Determine the aggregation state of elements in the Inter-stellar Medium in the Galaxy (solid state, molecular, atomic, etc) by precise measurements of the edge energy of bound-free transitions.
Instruments: **XGS**

Galactic Center

Study the energetics and geometry of the various components around Sgr $A^\star$.
Instruments: **WFI, XPOL**

Accretion in Young Stellar Objects

Characterize the geometry of accretion disks in Young Stellar Objects, via time resolved high-resolution spectroscopy.
Instruments: **XGS**

The Atmospheres of Solar System Planets

Study particle populations and their acceleration mechanisms occurring in the outer layers of the atmospheres of these planets and show their link to solar activity.
Instruments: **XMS**

5.5 HTRS Instrument and Detector

The IXO science case called for the capability to observe the strongest X-ray sources with count rates up to one million counts per second. The High Time Resolution Spectrometer was developed to perform these observations. The HTRS provides a good spectral resolution of 150 eV at 6 keV simultaneously with sub-millisecond timing, low dead-time and low pile-up (< 1 % at 1 Crab).

In order to meet these performance requirements, the HTRS is based on an array of 31 Silicon Drift Detectors (SDDs) in a circular shape as shown in Figure 5.12 with a sensitive volume totaling 4.5 cm^2 x 450 µm.

Overview of the Key Capabilities of the HTRS

- high precision timing measurements up to 1 million counts per second
- operating bandwidth in the range of 0.3 keV - 15 keV
- spectral resolution of 150 eV FWHM @ 6 keV
- event losses due to pile-up and dead-time < 1 % @ 1 Crab

5.5.1 HTRS Detector Chip

Silicon Drift Detectors (SDDs) are based on the principle of sidewards depletion. A large volume of a high resistivity semiconductor, in this case n-type silicon, is depleted by a small sized n$^+$ bulk contact that is reverse biased with respect to rectifying p$^+$ junctions covering both surfaces of the structure. The p$^+$ junctions are strip-like segmented and biased in such a way that an electric field parallel to the surface exists. Figure 5.13 shows the described structure of an SDD. Electrons released within the depleted volume by the absorption of ionizing radiation or by thermal generation drift in the field towards the n$^+$ substrate contact, which acts as collecting anode and is connected to an amplifier. The generated holes are drained away by the p$^+$ junctions (Lechner et al. 2010).

5.5.2 HTRS Detector Electronics

The baseline for the signal amplifying electronics is a fully analog readout chain integrated in an 8-channel readout chip. The readout circuit provides pre-amplification and analog filtering of the SDD signals. The pre-amplifier operates with the first FET

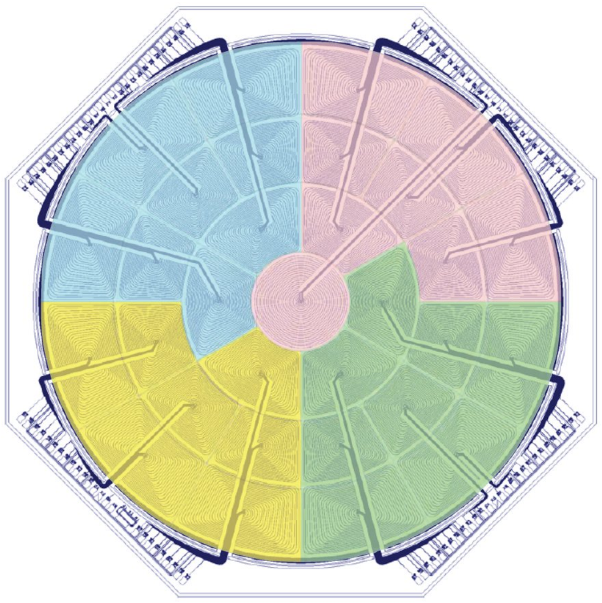


Figure 5.12
The HTRS detector chip has four quadrants, each with its on read-out electronic and power support. If one part of the detector fails, this merely reduces the sensitivity since the whole instrument is placed slightly out of focus. Source: Peter Lechner, Munich (2010).

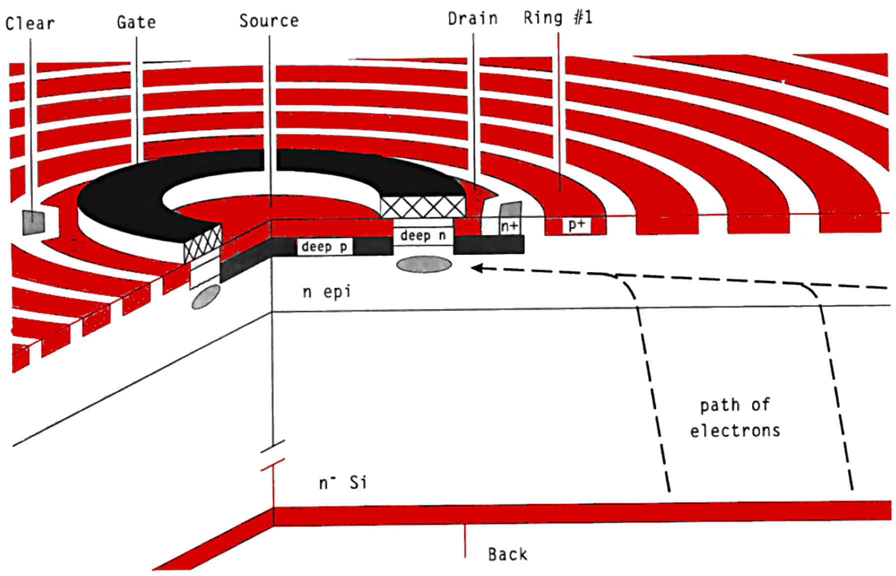


Figure 5.13
Schematic view of a cylindrical Silicon Drift Detector. Electrons are guided by a radial electric field towards the small sized collecting anode and the readout transistor in the center of the device. Source: Lechner et al. (2010).

integrated on the SDD chip configured as a charge sensitive amplifier (Niculae et al. 2006).

Since the HTRS is a non-imaging device it will be operated out of focus, in such a way that the focal beam from the mirrors is spread almost uniformly over the 31 SDDs to reduce dead-time and pile-up and therefore increase the overall count rate capability of the instrument.

5.5.3 HTRS Detector Readout

The 31 Silicon Drift Detectors are capable of performing an event-triggered readout completely independent for each cell in parallel, thus allowing for very high count rates up to 1 million counts per second. Event separation in one cell is possible up to 200 ns.

A temperature dependent leakage current and the signal charges are collected on a feedback capacitance. On this capacitance the charges generate an increasing voltage with short, steep steps whenever a photon is absorbed. Whenever the output voltage exceeds a given threshold value the readout chip produces a signal which is used to trigger an external reset circuit discharging the SDD. This process is visualized in Figure 5.14.

For this purpose a p^+-doped reset diode is integrated directly in the anode (also shown in Figure 5.13. The reset diode is reverse biased (-15 V) during signal integration and pulsed to forward bias (+7 V) for the reset. The duration of the reset pulse is 500 nsec. The reset signal is common for all eight channels of the readout chip, and so the average reset period is determined by the channel with the highest sum of leakage and signal currents (Niculae et al. 2006).

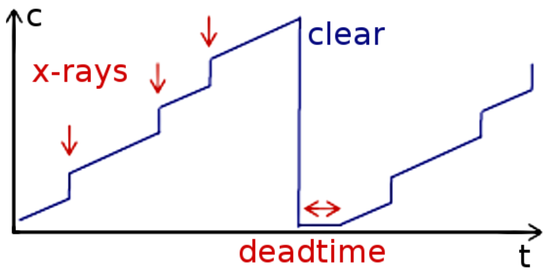


Figure 5.14
Anode charge in one cell over time. Note the constant increase due to dark currents. The arrows mark charge increases caused by X-ray photons that trigger a detector cell readout. After the readout the cell is not cleared, only after a certain threshold is reached the anode is cleared and inoperable for a short time.

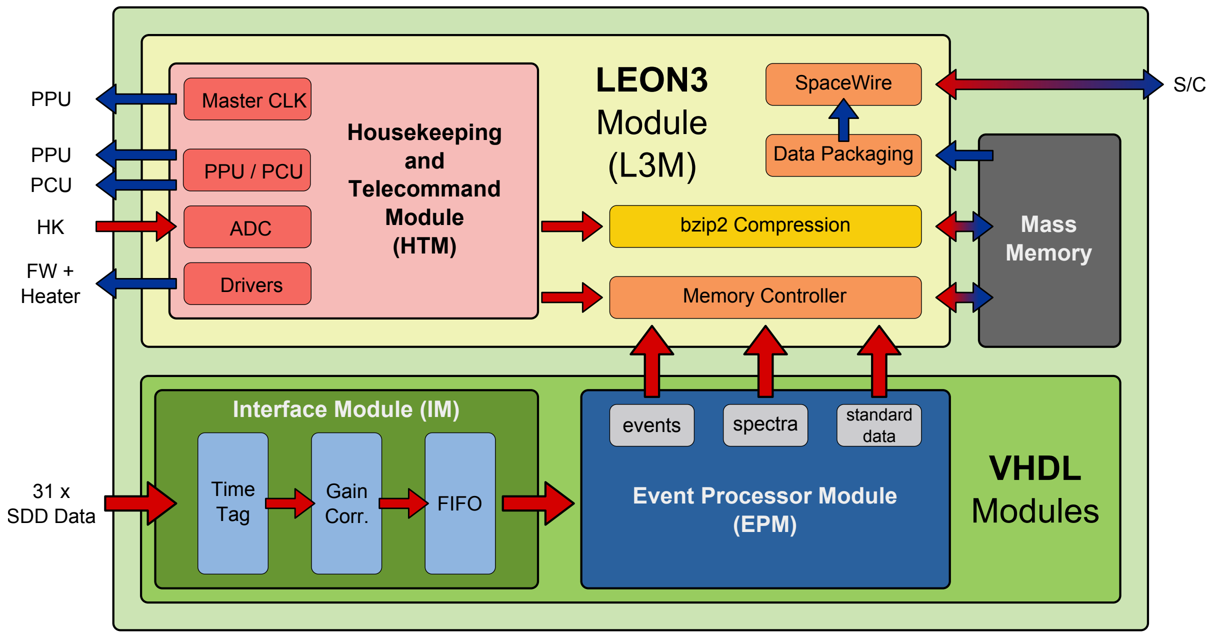


Figure 5.15
The two parts of the Data Processing Unit of the HTRS. The LEON3 VHDL microcontroller is implemented in the same FPGA as the IM and EPM.

5.6 HTRS Data Processing Unit

The HTRS Data Processing Unit (DPU) is the instrument's main control & command unit. It is responsible for controlling the detector chip and the read-out electronics, for power distribution, housekeeping, and for interfacing the spacecraft as well as operating a mass memory to store the scientific data until it can be linked down to ground. A very important task of the DPU is the reduction of the telemetry data rate handed to the spacecraft. Whenever the spacecraft requests data from the HTRS for downlink (usually during HTRS observations) the available telemetry rate is 0.75 Mbit/s. The reduction of the raw data rate of up to 82 Mbit/s by a factor of 100 is achieved by creating spectra instead of single events and further data compression. This process is described in detail in the following sections.

In a first step the data handling FPGA offers several configurable detector operation modes to reduce the amount of data while enabling the observer to optimize the scientific output of the observation. The second step is lossless on-board data compression and intelligent wrapping of the data done by a LEON3 CPU additionally implemented in the FPGA. Both units form the DPU of the HTRS instrument which is being developed in a two-part VHDL design as is shown in Fig. 5.15.

5.6.1 Data Rate Reduction

For dim sources with brightness of up to 0.1 Crab the single-event-mode can be used. When a single event is detected in one of the 31 independent SDD cells of the detector chip this event is received and processed by the Interface Module (IM). The IM will apply a time-tag and gain correction to the event and since it processes events from all 31 cells in parallel it will store all events in chronological order in a FIFO buffer (serialization). The events are then handed to the EPM that produces the configured scientific data product.

A single event that is detected in one of the detector cells contains energy information, detection time and a pixel id corresponding to the cell it was detected in. These information form the basic event packet and when operating in single-event mode these events are handed from the EPM to the DPU, bzip2 compression is applied to the event stream and the data is finally downlink via the spacecraft to ground.

Size of a single event package

```
  24 bit time information
+  5 bit pixel id
+ 12 bit energy information
```

Resulting data rate

$2 \cdot 10^6$ cts/s $\cdot$ 41 bit
= 82 Mbit/s

Available net telemetry rate for the HTRS

0.75 Mbit/s

For bright objects (> 0.1 Crab) the single-event data rate exceeds the telemetry limit by far and the resulting data rate for a source with 10 x the brightness of the Crab nebula is too high by a factor of 100. Therefore data reduction is indispensable for bright sources.

To reduce the sing-event data several highly configurable spectrum-modes have been implemented in the EPM. In theses modes a spectrum with given energy resolution is integrated on-board over a given time. The energy channels of the spectrum can be defined via uploading a channel matrix file that describes the energy range that is assigned to each individual channel of the spectrum. The integration time for one spectrum can also be set be defining the number of spectra per second as shown in Figure 5.16.

In spectrum-mode the EPM produces a constant data rate that is independent of the brightness of the source. The data rates of several spectrum-mode configurations are shown in Figure 5.16. Applying a lossless bzip2 block compression algorithm to the data enables the HTRS instrument to operate within the required telemetry limit of 0.75 Mbit/s. The resulting data rates and the achieved compression ratios are shown Figure 5.17.

All data rate estimations in spectrum-mode are based on simulations of the detection of the Crab nebula at 10x its brightness ($2 \cdot 10^6$ events/s = 82 Mbit/s) done at the IAAT in Tübingen as part of this Thesis.

For this purpose a HTRS simulator was written in C++ to simulate the functionality of the IM (event handling) and the EPM (spectrum generation) as well as the bzip2 compression algorithm and data handling applied by the DPU. For the simulation of the mirror and detector properties the work of Michael Martin was used (Martin 2009). The data compression was also implemented in the LEON3 microcontroller as described in Chapter 5.6.3 - DPU Prototype.

5.6.2 Channel-Bit-Width

The bit-width of the channels in a spectrum is a very important decision when the instrument is operated in spectrum-mode and has to be carefully considered. A large bit-width prevents integer overflows in the event-counter of a spectral channel and therefore reduces the complexity of the data handling at the cost of a linearly higher data rate.

To study the effect of the bzip2 compression on the bit-width of the spectral channels, we simulated source-spectra with constant energy distribution and a completely randomized, but limited, count rate per channel. This means that by limiting the count rate, only a fixed amount of (lower) bits per channel is used to store the random number of counts. The remaining, unused 'upper' bits of the channel all remain zero as it would be the case when observing a source with a given brightness. The values that are ultimately stored in the channel are randomized to negate the effect that compression might have on these parts of the data. The compression was applied to blocks of these spectra and the resulting data reduction is given in Table 5.2.

The first two columns in the table give the used and unused bits (always totaling 32). Ideally the compression (4th column) is the same as the relative amount of unused bits (3rd column). An ideal compression should simply cut away the unused upper bits as would the reduction of the bit-width of the used data type.

As can be seen from the table, the achieved data reduction is generally comparable to the direct cutting of the unused bits (i.e. using a smaller bit-width). Another result is shown in Figure 5.18. Here we studied the effect of the endianess (the order of the bits) of the used data type. The plot shows the deviation of the achieved compression from

	spectra per second					
	128	256	512	1024	2048	4096
channels						
4	0,01	0,02	0,03	0,06	0,13	0,25
8	0,02	0,03	0,06	0,13	0,25	0,50
16	0,03	0,06	0,13	0,25	0,50	1
32	0,06	0,13	0,25	0,50	1	2
64	0,13	0,25	0,50	1	2	4
128	0,25	0,50	1	2	4	8
256	0,50	1	2	4	8	16

Figure 5.16
Uncompressed data rates in spectrum-mode configuration. Rates are in Mbit/s.

	spectra per second					
channels	128	256	512	1024	2048	4096
128	0,05	0,08	0,13	0,21	0,32	0,47
256	0,08	0,13	0,20	0,31	0,46	0,64

	compression factor					
channels	128	256	512	1024	2048	4096
128	5,3	6,3	7,7	9,7	13	17
256	6,4	7,8	9,9	13	17	25

Figure 5.17
Compressed data rates and compression factor. Rates are in Mbit/s.

the ideal compression, which is similar to the direct cutting of the unused bits. Over a wide range the achieved compression is very good, only for four to eight unused bits in the spectral channels the compression decreases to 90 % of the ideal value. As can be seen from the plot, the effect of the endianess can be neglected.

The bzip2 compression thus allows the use of up to 32 bits per channel while still preserving a very efficient use of the available data rate. This large bit-width will enable the HTRS to observe highly variable sources over a wide range of intensities.

5.6.3 DPU Prototype

A fast and specialized FPGA (e.g. Virtex 4) is used to implement an interface module to the detector read-out electronics (IM) and the event processor module (EPM) that provides the different operation modes of the HTRS instrument. Since a microcontroller is needed to apply lossless bzip2 compression to the singe-event and spectrum-mode data, a LEON3 is implemented in the same FPGA and internally connected to the EPM. More details on the LEON3 can be found in Chapter 4.3 - LEON3.

The DPUs operating system is developed using RTEMS[3], a real-time executive that has a POSIX[4] API[5] which is required to build the bzip2 library.

The prototype board can be connected to a PC via a serial interface and spectrum-data can be sent directly to the LEON3 DPU. The DPU processes the data and applies the bzip2 compression algorithm before handing the data back to the PC. The prototype can also produce the various scientific products when raw event data is provided.

While the flight version of the DPU will utilize a Virtex 4 FPGA, we developed the DPU prototype on a Xilinx Spartan 3 XC3S on a Pender GR-X3CS development board that is shown in Figure 5.19.

Gaisler also provides an RTEMS implementation for the Virtex 4 and a LEON3 multiprocessor version. Since the bzip2 compression speed scales linearly with the number of processors this provides the possibility to further increase the DPUs performance if necessary.

Both parts of the DPU, the VHDL modules (IM, EPM) and the LEON3 part have been developed at our institute in Tübingen and a prototype board has been built to prove the feasibility of the proposed data handling procedures, and to identify the performance of the data compression.

[3]**R**eal-**T**ime **E**xecutive for **M**ultiprocessor **S**ystems
[4]**P**ortable **O**perating **S**ystem **I**nterface for Uni**x**
[5]**A**pplication **P**rogramming **I**nterface

Table 5.2
This table shows how effective the spectrum data can be compressed when a large data type (e.g. 32 bit) is used to store the channel counts. The 1st and 2nd columns show how many of the 32 bits are used to store the (random) channel count and how many are zero. The amount of unused bits is the amount of data rate reduction that can be achieved by using an optimized bit-width. The last column shows the data rate reduction achieved by the bzip2 compression which in most cases is very close to the ideal value.

Random Bits	Unused Bits	Unused Bits	Compression Strength
32	0	0 %	0 %
28	4	13 %	3 %
24	8	25 %	14 %
20	12	38 %	35 %
16	16	50 %	49 %
12	20	63 %	62 %
8	24	75 %	75 %
4	28	88 %	87 %
1	31	97 %	96 %

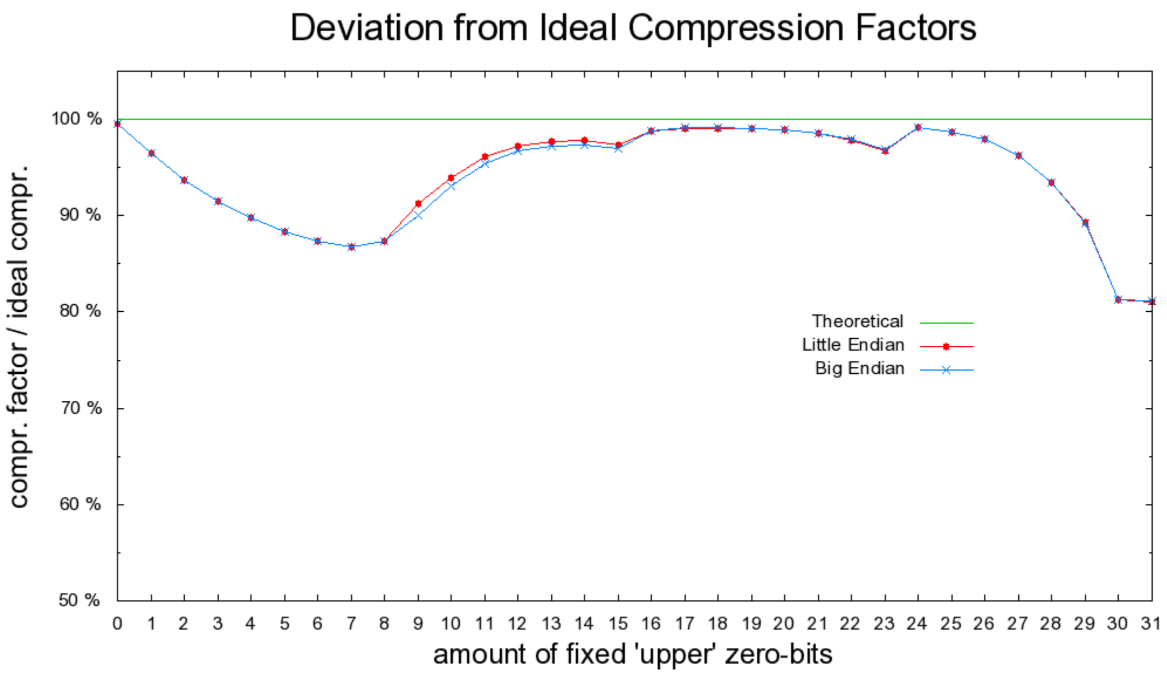


Figure 5.18
Shown is the deviation of the achieved bzip2 compression from the theoretical (ideal) compression, which is similar to the direct cutting of the unused bits. The effect of the endianess of the channel count can be neglected.

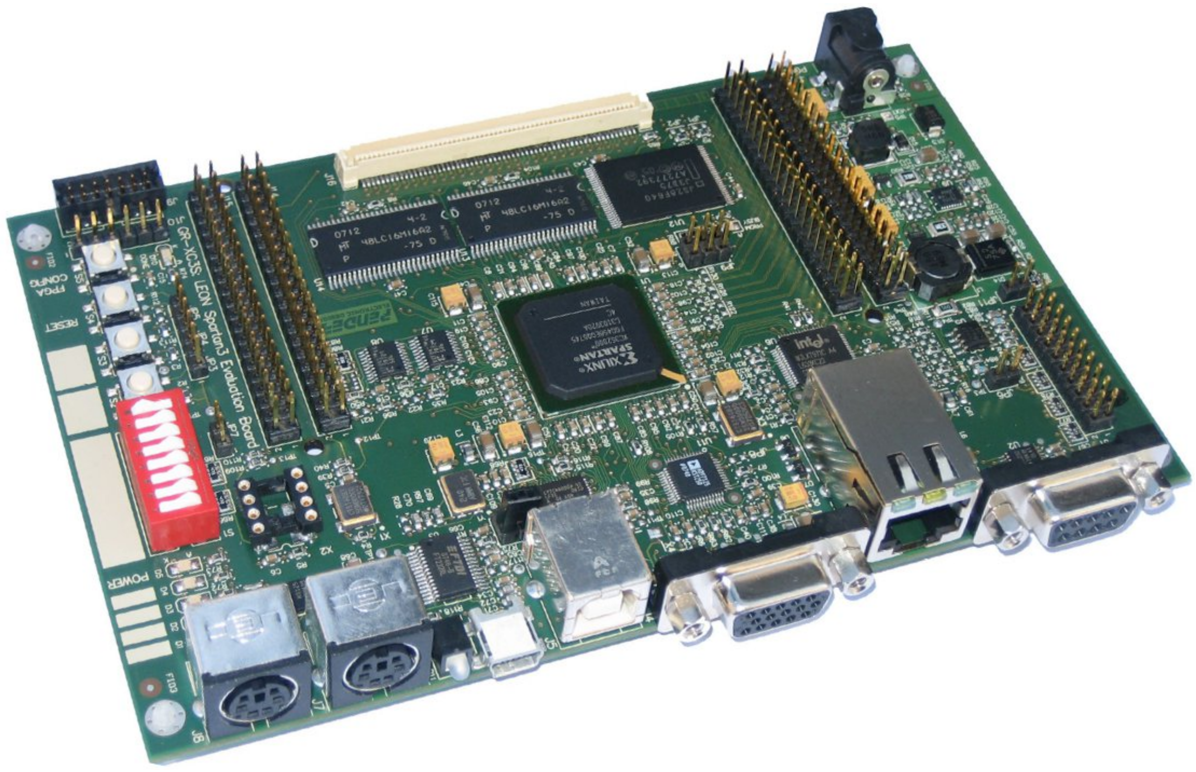


Figure 5.19
The Pender GR-XC3S development board carrying a Xilinx Spartan 3 FPGA. The LEON3 microcontroller VHDL design is implemented in the FPGA and is connected to a PC via one of the RS232 interfaces to send and receive commands. A mini-USB to RS232 adapter is connected to a second PC to simulate the connection to the HTRS detector.
Source: Pender Electronic Design GmbH, Switzerland.

5.6.4 HTRS Conclusions

In the context of this thesis the DPU operations (esp. spectrum generation and compression) have been simulated and studied in a simulation environment and with a prototype board. It has been concluded that the requirements on the telemetry rate can be met using the bzip2 compression algorithm. The performances of several other compression algorithms (gzip, zlib, lzma, paq9a) have been compared and it was found that bzip2 is well suited for several reasons such as compression strength and speed, data integrity verification, and possible parallelisation. In fact bzip2 was the only algorithm that achieved the desired compression strength while the compression could be performed in real-time.

We also implemented and successfully completed an exemplary test run of the compression on our LEON3 development board using a Xilinx Spartan 3 FPGA. An important improvement to the knowledge of the performance will be to experimentally operate a mass memory and to determine the resulting time constraints on DPU operations. We assume that I/O operations will have a significant impact on the compression speed. A study of the mass memory has not yet been carried out since after the the reformulation of IXO (see next chapter) the HTRS instrument is no longer part of the mission design.

5.7 From IXO to ATHENA

In 2011, ESA decided on a new way forward for the three L-class mission candidates to take account of developments with ESA's international partners. During the reformulation exercise IXO became ATHENA.

After the selection of JUICE (Jupiter Icy Moons Explorer) as current L1 mission by ESA in 2012 ATHENA was again revised to become a candidate for the 2013 selection of a science theme for an L2 or L3 mission and in September 2013 "The Hot and Energetic Universe" was in fact selected as science theme for an L2 mission with a possible launch in 2028. ATHENA is the current candidate for this mission slot.

ATHENA uses the same silicon pore optic development as IXO for its mirrors with an effective area of $2\,\mathrm{m}^2$ @ 1keV and $0.25\,\mathrm{m}^2$ @ 6keV. The mission now features two distinct scientific instruments.

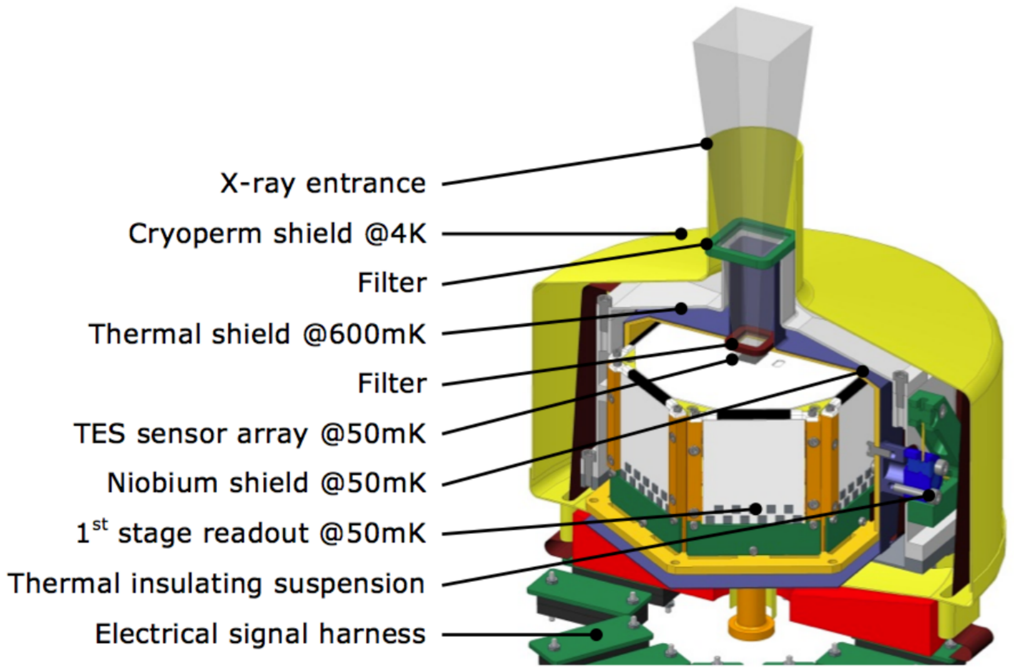


Figure 5.20
The X-ray Integral Field Unit (X-IFU) focal plane assembly. The outer shield is cooled down to 4 K. Between the detector, which is at 50 mK and the outer shield there is an intermediate thermal radiation shield at 0.6 K. The read-out electronics are mounted on the side walls of the focal plane assembly. Source: Barret et al. (2013)

One is the X-ray Integral Field Unit (X-IFU), a micro-calorimeter with 3840 Transition-Edge-Sensors (TES). These have a size of 250 µm each and a energy resolution of 2.5 eV @ 6 keV. This instrument is based on the IXO XMS instrument and operates in the range of 0.2 keV to 10 keV. It can observe sources with count rates ranging from 1 mCrab to 1 Crab with a time resolution of 10 µs. The main part of the instrument is shown in Figure 5.20.

The other instrument is the Wide-Field Imager (WFI) that is based on the WFI developed for IXO. This imaging instrument uses a DEPFET camera with 5 camera chips (256 x 256 pixel and 4 x 448 x 640 pixel) covering the energy range of 0.1 keV to 12 keV with an energy resolution of 150 eV @ 6 keV and an angular resolution of 5 arcsec in a field of view of 40′ x 40′. The instrument is shown in Figure 5.21 in its flight configuration.

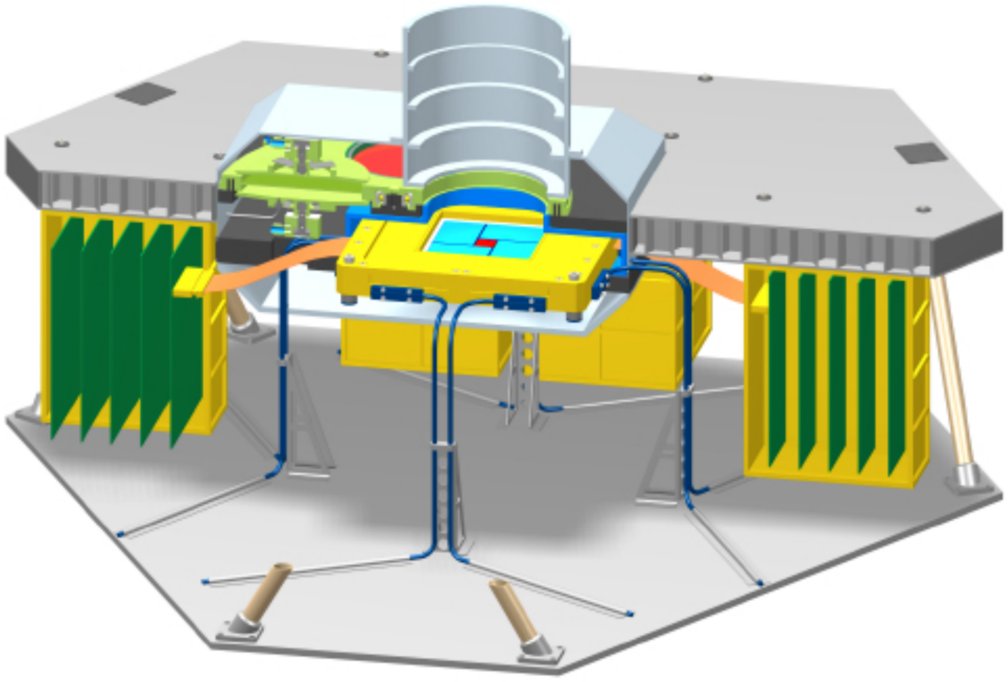


Figure 5.21
This cut through the WFI design shows the camera module at the center of the assembly. A X-ray baffle is placed on top of the filter wheel above the camera. Flexleads connect to the electronic boxes holding the read-out electronics. Source: Rau et al. (2013)

Chapter 6

LOFT

6.1 Introduction

The Large Observatory for X-ray Timing (LOFT) is one of five satellite mission candidates selected by ESA as a medium class (M3) mission in the context of their Cosmic Vision 2015-2025 programme[1] which is explained in more detail in Chapter 5.3. The five mission candidates selected for an assessment study are

- **EChO** - Exoplanet Characterisation Observatory
- **LOFT** - Large Observatory For X-ray Timing
- **MarcoPolo-R** - Asteroid Sample Return Mission
- **PLATO** - PLAnetary Transits and Oscillations of stars
- **STE-QUEST** - Space-Time Explorer and Quantum Equivalence Principle Space Test

Each of these missions addresses a certain theme of the Cosmic Vision 2015-2025 programme. The X-ray observations that can be carried out with LOFT address theme four of the programme "How did the Universe originate and what is it made of?"

In more detail LOFT, which is shown in see Figure 6.1, is going to answer fundamental questions about the motion of matter orbiting close to the event horizon of a black hole, and the state of matter in neutron stars. Therefore LOFT will detect their very rapid X-ray flux and spectral variability. To address the scientific goals of the mission that are presented in more detail in Chapter 6.3 the spacecraft is equipped with two distinct scientific instruments:

[1] The Cosmic Vision programme includes one small mission, three medium missions (M1, M2, M3) and one large mission. Only one of the five missions will be selected for the M3 mission slot. M1 and M2 missions have already been chosen with Solar Orbiter (launch 2017) and Euclid (launch 2020).

- **LAD**, the Large Area Detector with an effective area around 10 m^2 (far larger than any current space-borne X-ray observatory)

- **WFM**, the Wide Field Monitor that will monitor a large portion of the sky to provide interesting targets for follow-up observations with the LAD, and also to provide a rapid burst-alert function to the broader X-ray astronomy community.

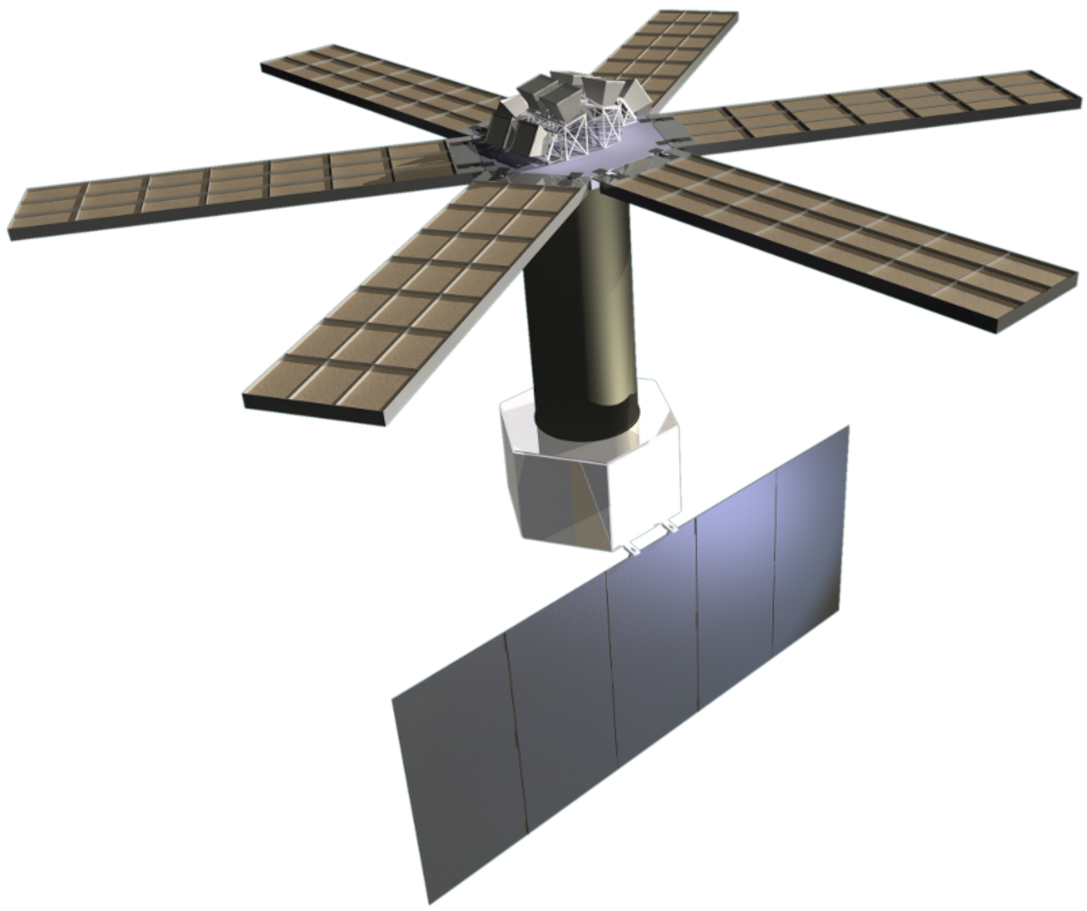


Figure 6.1
The Large Observatory for X-ray Timing (**LOFT**). The primary instrument, the Large Area Detector (**LAD**), consists of the six panels and includes nearly 2000 hand-sized Silicon Drift Detectors (SDD) grouped into 126 modules. The combined effective area is around 10 m^2 outperforming every other past/present/future X-ray mission. The 10 cameras on top of the satellite are part of the second instrument, the Wide Field Monitor (**WFM**). It can observe one third of the sky at all times, detect transient X-ray sources and immediately transmit their position to ground. Source: ESA (2013).

6.2 Scientific Background

High-time-resolution X-ray observations of compact objects provide direct access to strong-field gravity, black hole masses and spins, and the equation of state of ultra-dense matter. LOFT will be able to exploit the relevant diagnostics and answer two fundamental questions of ESA's Cosmic Vision Theme 4.3 "Matter under Extreme Conditions":

- Does matter orbiting close to the event horizon follow the predictions of general relativity?
- What is the equation of state of matter in neutron stars?

With an innovative design and the development of large monolithic Silicon Drift Detectors (SDD), the Large Area Detector (LAD), the primary instrument of the mission, will achieve an effective area of $\sim 10\,\mathrm{m}^2$ in the range 2–30 keV range (up to 80 keV in expanded mode). With this large area and a nominal spectral resolution of <240 eV over the entire band, LOFT will allow studies of collapsed objects in our Galaxy and of the brightest supermassive black holes in active galactic nuclei (AGN), which will yield information on strongly curved space-times and matter under extreme conditions of pressure and magnetic field strength. In addition to these core science goals, LOFT provides a 50 % allocation of the observing time to observatory science.

6.3 LOFT Scientific Requirements

The LOFT key science requirements are listed below. Titles are the same as in the *LOFT Assessment Study Report* (ESA 2013) and are used as references in the following chapters.

1. **EOS1**
   Constrain the equation of state of supranuclear-density matter by the measurement, using three complementary types of pulsations, of mass and radius of at least 4 neutron stars with an instrumental accuracy of 4 % in mass and 3 % in radius.
2. **EOS2**
   Provide an independent constraint on the equation of state by filling out the accreting neutron star spin distribution through discovering coherent pulsations down to an amplitude of about 0.4 % (2 %) rms for a 100 mCrab (10 mCrab) source in a time interval of 100 s, and oscillations during type I bursts down to typical amplitudes of 1 % (2.5 %) rms in the burst tail (rise) among 35 neutron stars covering a range of luminosities and inclinations.

3. **EOS3**
   Probe the interior structure of isolated neutron stars by observing seismic oscillations in Soft Gamma-ray Repeater intermediate flares when they occur with flux 1000 Crab through high energy photons ($> 20\,\mathrm{keV}$).

4. **SFG1**
   Detect strong-field GR effects by measuring epicyclic motions in high frequency QPOs from at least 3 black hole X-ray binaries and perform comparative studies in neutron stars.

5. **SFG2**
   Detect disk precession due to relativistic frame dragging with the Fe line variations in low frequency QPOs for 10 neutron stars and 5 black holes.

6. **SFG3**
   Detect kHz QPOs at their coherence time, measure the waveforms and quantify the distortions due to strong-field GR for 10 neutron stars covering different inclinations and luminosities.

7. **SFG4**
   Constrain fundamental properties of stellar mass black holes and of accretion flows in strong field gravity by (a) measuring the Fe-line profile and (b) carrying out reverberation mapping and (c) tomography of 5 black holes in binaries providing spins to an accuracy of 5 % of the maximum spin and do comparative studies in 10 neutron stars.

8. **SFG5**
   Constrain fundamental properties of supermassive black holes and of accretion flows in strong field gravity by (a) measuring the Fe-line profiles of 20 AGNs and for 6 AGNs (b) carry out reverberation mapping and (c) tomography, providing BH spins to an accuracy of 20 % of the maximum spin (10 % for fast spins) and measuring their masses with 30 % accuracy.

These requirements can only be met with a correlated set of instrumental and mission parameters. This means that e.g. the mission duration and the effective area can be balanced to some extend. The most important science parameter is the spectral-timing sensitivity. This can be optimized by the simultaneous optimization of all parameters, with the science impact varying very slowly and smoothly with each of them. The main parameters are:

- Effective Area
- Spectral Resolution
- Sky Visibility

- Mission Duration

Many parameters have potential margins which can be used to compensate a loss on another:

- 10 - 20 % on Effective Area
- 15 % on Sky Visibility
- 400 % on Radiation Damage
- 7 °C on Operating Temperature (hence Spectral Resolution)
- 15 % on Mission Lifetime

The final result of the optimization process is presented in Chapter 6.4.

6.4 LOFT Engineering Requirements

The requirements presented in this chapter reflect the state of the mission as presented in the *LOFT Assessment Study Report* (ESA 2013) published in December 2013.

6.4.1 Effective Area

The 10 m^2 effective area requirement for the core science is derived from two main observational needs: reducing Poisson noise for relatively bright sources to access weak timing features (e.g., EOS1-3, SFG1-3; see Chapter 6.3) and gathering high-quality spectra for phenomena occurring on intrinsically short time scales in weak (SFG5) or bright sources (SFG2,4). Examples of the first category are the measurement of epicyclic frequency QPOs (SFG1) and intermittent X-ray pulsars (EOS2) within the time that these signals persist unchanged (minutes). Examples of the second category are the direct observation of millisecond orbital motion close to stellar mass black holes (SFG4), and Fe-line tomography in AGNs to derive 10 - 20 % constraints to the spin of the supermassive black hole (SFG5). Figure 6.2 shows the anticipated LAD effective area, as compared to the past/ongoing/planned largest X-ray missions.

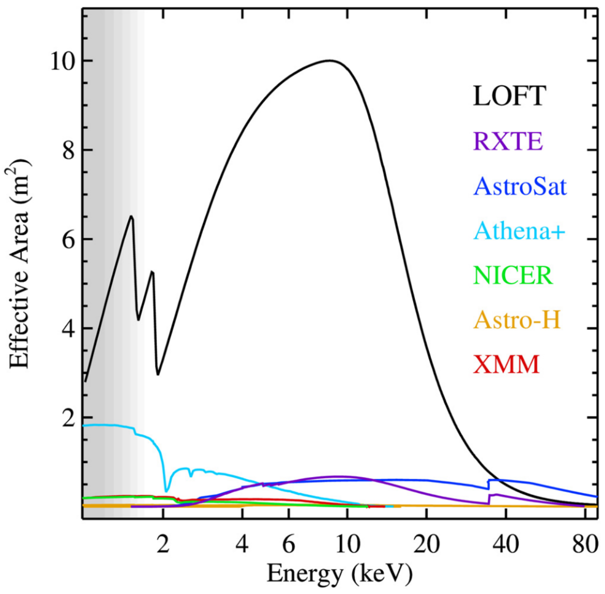


Figure 6.2
The effective area of LOFT/LAD as compared to the largest past, on-going and approved missions. The shaded area below 2 keV indicates the fact that the LAD will outperform its science requirement of 2 keV low energy threshold by extending its sensitivity down to 1 - 1.5 keV. Source: ESA (2013).

6.4.2 Spectral Resolution

A good spectral resolution is required for a number of LOFT core science objectives (but not all of them), namely strong-field gravity through studies of gravitationally broadened Fe Kα line. The requirement on the end-of-mission spectral resolution integrated over the full detector (240 eV FWHM @ 6 keV) was set as the resolution allowing to measure the Fe-line parameters (enabling SFG2, SFG4, SFG5), e.g. the BH spin with a minimum accuracy on a/M of 10 - 20 % in 20 AGN and 5 % in 5 stellar mass black holes.

6.4.3 Sensitivity

The enormous effective area of the LAD also requires very good counting statistics. The standard candle of X-ray astronomy, the Crab Nebula, is expected to provide a count rate as high as 240 000 cts/s. This means that the statistical LAD 3σ sensitivity for persistent sources is tremendously high (see Figure 6.3).

For longer integrations (> 10 ks) the statistical uncertainties become so small that systematics in the background subtraction dominate. In Figure 6.3 the LAD sensitivity curves are provided as a function of the exposure time, for different levels of required signal-to-noise ratio, under the assumption of the required 0.25 % background systematics as well as the expected value of 0.15 %. The sensitivity curves show that a 5σ detection of a 0.1 mCrab sources takes 100 s. A signal to noise of 100 such as required for sensitive spectral line studies is reached after 10 ks for a 0.5 - 1 mCrab source.

When observing rapidly variable (as compared to the orbital background variation) signals, the sensitivity is not affected by any incoherent residual background systematics and continues to improve at long integration times. This includes not only timing analysis but spectral variability as well, as long as it is rapid enough.

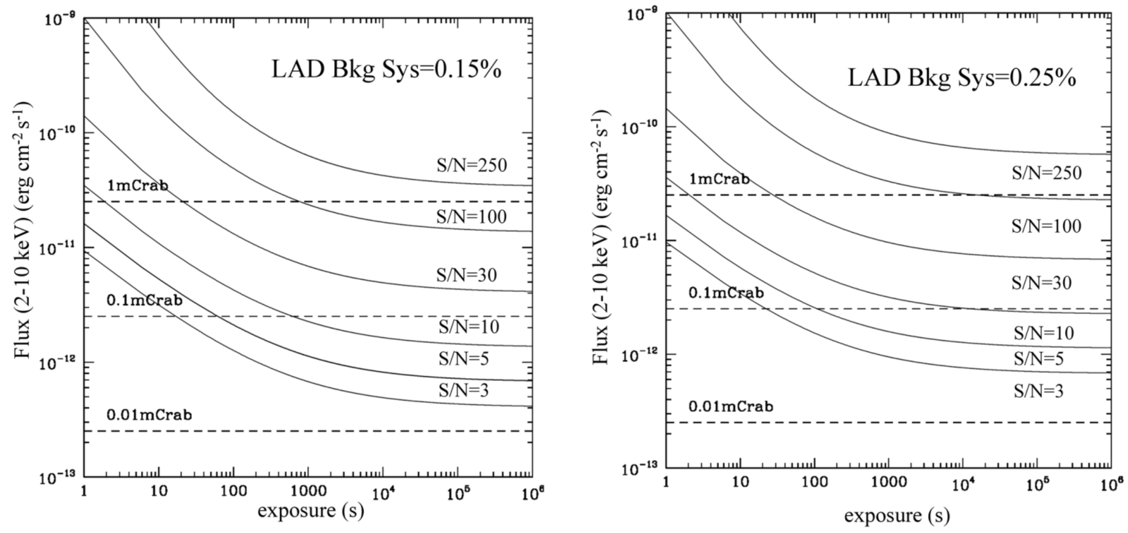


Figure 6.3
LAD sensitivity versus exposure time for different signal-to-noise levels. Both cases of the requirement (0.25 %) and expected (0.15 %) background systematics are provided. Source: ESA (2013).

6.4.4 Monitoring of Transient Sources

Many of the sources which are key to the success of the mission are variable sources and LOFT should observe these in the relevant bright state. Therefore the LOFT mission includes a Wide Field Monitor that will be used to monitor the state of the relevant sources but will also alert the wider scientific community about transient events in known sources as well as identify new and unexpected sources. These requirements are best achieved by optimizing the field of view. The baseline WFM design envisages a simultaneous sky coverage as large as 1/3 of the whole sky with > 20 % of its peak effective area. This responds to its main requirement of monitoring at least 50 % of the sky fraction accessible to the LAD at any time (for transient monitoring). The co-alignment of the WFM and LAD field of view will also allow use of the arcmin-resolution imaging capability of the WFM to monitor the field of view observed by the LAD at the same time. The shape of the field of view of the WFM is shown in Figure Figure 6.4.

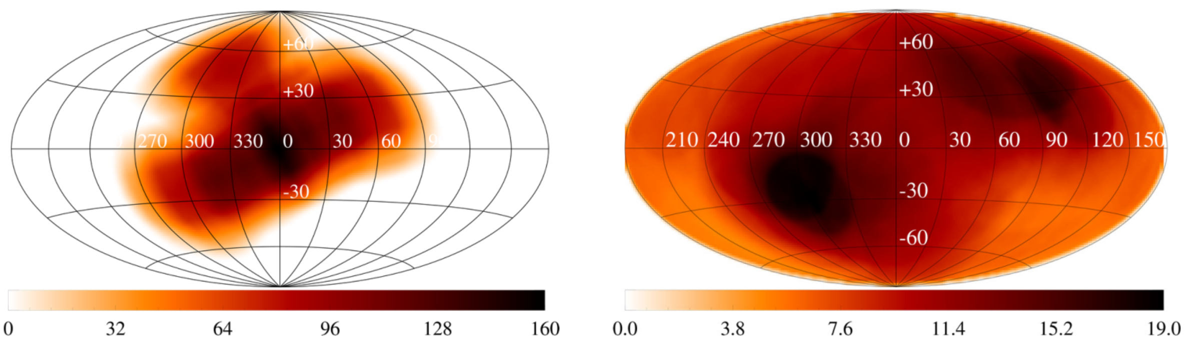


Figure 6.4
Left: galactic coordinates of the active detector area for a sample observation performed in the direction of the Galactic Center. Right: 1 year exposure map in galactic coordinates for positions covered by at least 10 cm^2 per observation. Map scale is given in Ms. Source: ESA (2013).

6.4.5 Mission Duration

The key elements for the requirement of the mission duration are (a) the overall observing time, (b) accessibility of required sources, (c) the probability to catch rare outbursts of transient black hole candidates (BHCT) and Accreting Millisecond X-ray Pulsars (AMXPs). A total exposure time of 25 Ms is required to execute the core program. Even with the minimum requirement of 40 % mission availability (while industrial studies showed an observing efficiency $> 60\,\%$) 25 Ms are available in less than 2 years. The effectively driving requirement comes from a $> 95\,\%$ probability to detect at least 3 outbursts from BHCT (in their intermediate states, where high-frequency QPOs required by SFG1 are detectable) and 3 AMXP outbursts. For the core-science observation of AMXP outbursts and for strong-gravity objective SFG1 the full energy resolution is not required and therefore the relevant sky visibility here is the eFOR. With a sky visibility of +30 / -70 degree the mission duration is 3 years. A reduced sky visibility of +30 / -50 degree could be compensated by a longer (4 year) required mission duration.

6.5 Wide Field Monitor

The Wide Field Monitor (WFM) is a coded mask camera designed on the heritage of the SuperAGILE experiment (Feroci et al. 2007). It provides solid state-class energy resolution through the use of Silicon Drift Detectors (SDDs) very similar to the LAD detectors. Since detectors provide accurate positions in one direction, while they deliver only rough positional information in the other direction, pairs of two orthogonal cameras are used to obtain precise two dimensional source positions (see Figure 6.6). The effective field of view of one camera pair is about 70° x 70°. The dimensions of each camera are chosen to match the required sensitivity and the location accuracy. To provide the full required sky coverage, 5 pairs of cameras are foreseen (Figure 6.6). The key instrument requirements are summarized in Table 6.1 along with the expected performance.

The working principle of the WFM is the classical sky encoding by coded masks (Fenimore and Cannon 1978) which is widely used in space borne instruments (e.g. INTEGRAL, RXTE/ASM, Swift/BAT). The mask shadow recorded by the position-sensitive detector can be deconvolved by using the proper procedures (in 't Zand 1996) and can therefore recover the image of the sky, with an angular resolution given by the ratio between the mask element and the mask-detector distance. In order to avoid losing imaging sensitivity, the mask element should not be smaller than twice the

Table 6.1
Requirements and expected performance of the LOFT Wide Field Monitor. Note that the FOV is given in steradian since the monitored area of all 5 camera it has a T-shape.
Source: ESA (2013).

Item	LOFT WFM Requirement	Expected Performance
Location Accuracy (2D)	< 1 arcmin	< 1 arcmin
Angular Resolution (2D)	< 5 arcmin	< 4.3 arcmin
Peak Sensitivity in LAD direction (5σ)	1 Crab (1 s) 5 mCrab (50 ks)	0.6 Crab (1 s) 2.1 mCrab (50 ks)
Field Of View	3.2 steradian	5.5 steradian
Energy Range	2 - 50 keV	2 - 50 keV
Energy Resolution	500 eV @ 6 keV	300 eV @ 6 keV
Absolute Time Accuracy	2 µs	1 µs
Availability of Trigger-Data	3 hours	< 3 hours
Broadcast Delay	30 sec after event	25 sec after event

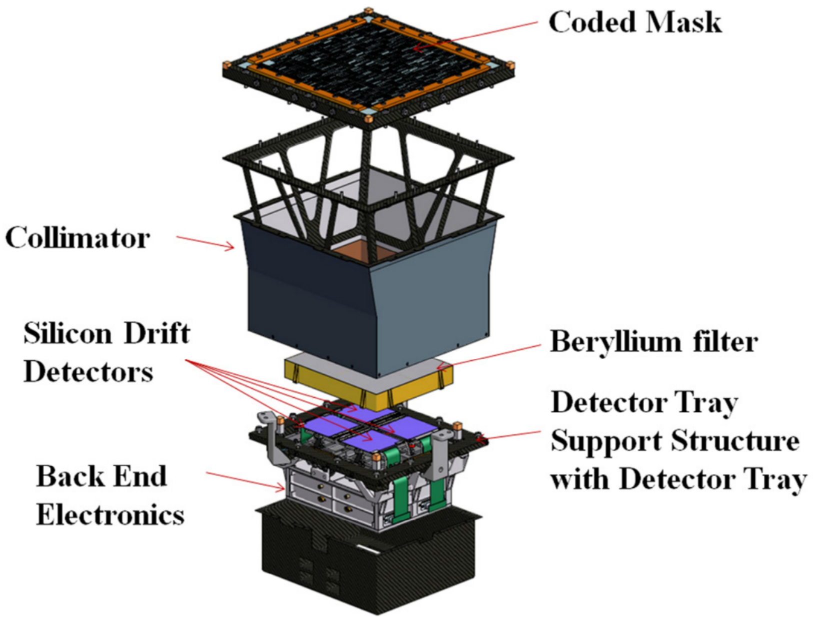


Figure 6.5
One WFM camera module consists of four Silicon Drift Detector behind a beryllium filter sitting at the end of the collimator with the coded mask on top. The Back-End-Electronics (one BEE per camera) which are very similar to the LAD MBEE have also been developed in Tübingen. Source: ESA (2013).

detector resolution element. By using a mask with mask elements of 250 µm in the fine resolution direction and a mask-detector separation of 200 mm the WFM can obtain an angular resolution < 5 arcmin. The coded mask imaging is the most effective technique to observe steradian-wide sky regions with arcmin angular resolution.

As a first approach, each WFM camera can be considered a one-dimensional coded mask imager. This means that after the proper deconvolution is applied to the detector images, the image of a sky region including a single point-like source will appear as a single peak over a flat background. The position of the peak corresponds to the projection of the sky coordinates onto the WFM reference frame. The width of the peak is the point spread function, of the order of a few arcmin in the LOFT WFM. If more than one source is present in the observed sky region, the image will show a corresponding number of individual peaks, whose amplitude will depend on the intensity of the source and on the exposed detector area at that specific sky location.

By observing the same sky region with two cameras oriented at 90° to each other simultaneously (such a pair composing one WFM Unit), the precise 2D position of the sources can be derived, by intersecting the two orthogonal 1D projections.

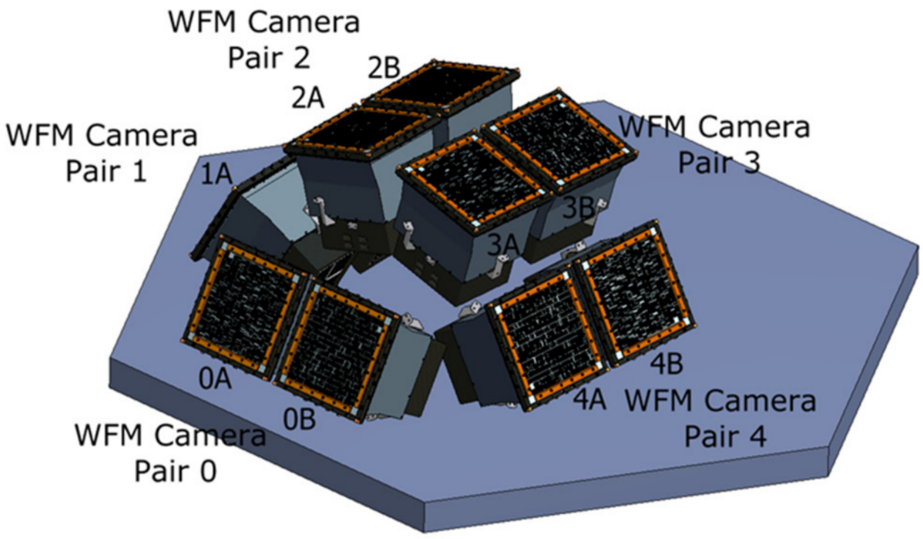


Figure 6.6
A combination of five camera pairs which are rotated 90° to each other gives the required sky coverage of 4.1 steradian. The rotation compensates for the fact the the detectors have a much better position accuracy in the x-direction. Source: ESA (2013).

6.6 Large Area Detector

The Large Area Detector (LAD) is designed as a classical collimated instrument. The LAD consists of 6 identical panels, each accommodating 21 detector modules as shown in Figure 6.8. Since there is no requirement on the number of panels the total number of 126 detector modules with a collecting area of 10 m^2 at 6 keV can be achieved with different panel numbers in a different layout as well. Still, 6 panels is the baseline design of the proposed mission.

Each of the 126 detector modules holds 16 monolithic Silicon Drift Detectors (SDD) (see Figure 6.7), which are split into and operated as two identical though independent detector halves. From a functional or operational point of view these detector halves can be considered to be the basic common unit of the LAD. One detector half is read out by 7 ASICs with 16 anodes each. The SDDs are 450 µm thick and sensitive to X-ray photons in the energy range 2 - 80 keV.

Table 6.2
Requirements and expected performance of the LOFT Large Area Detector. Note: eFOR is the extended field of regard for the LAD. High-energy photons can pass through the collimator from an extended FOV and can be detected by the SDDs. Source: ESA (2013).

Item	LOFT LAD Requirement	Expected Performance
Effective Area	3.8 m^2 @ 2 keV 7.6 m^2 @ 5 keV 9.5 m^2 @ 8 keV 0.95 m^2 @ 30 keV	4.4 m^2 @ 2 keV 9.0 m^2 @ 5 keV 9.8 m^2 @ 6 keV 1.3 m^2 @ 30 keV
Energy Range	2 - 30 keV nominal 30 - 80 keV extended (events outside LAD FOV)	1.5 - 30 keV (7.5σ) 30 - 100 keV
Energy Resolution (FOV)	240 eV @ 6 keV	180 eV @ 6 keV
Energy Resolution (eFOR)	400 eV @ 6 keV	210 eV @ 6 keV
Absolute Time Accuracy	2 µs	1 µs
Dead-Time	< 1 % @ 1 Crab	< 0.7 % @ 1 Crab
Background	10 mCrab	9 mCrab
Background Knowledge	0.25 % @ 5 - 10 keV	0.15 % @ 5 - 10 keV
Max. Flux (sustained)	500 mCrab	650 mCrab
Max. Flux (cont. 300 min)	15 Crab	15 Crab

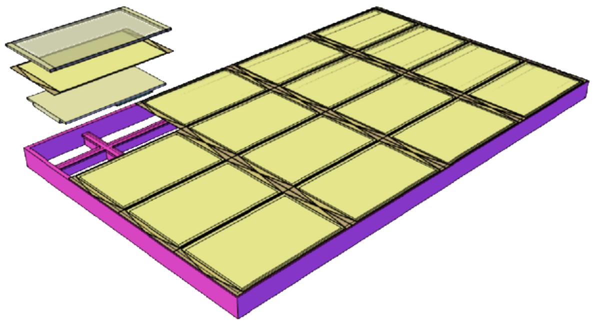


Figure 6.7
LAD Module Layout: Each module is comprised of 16 Silicon Drift Detectors. The detectors are mounted between the Front-End-Electronics and the collimator plate. The Front-End-Electronics contain the ASICs and are operated by the Module-Back-End-Electronics (not shown here) of which one is part of every module. Source: ESA (2013).

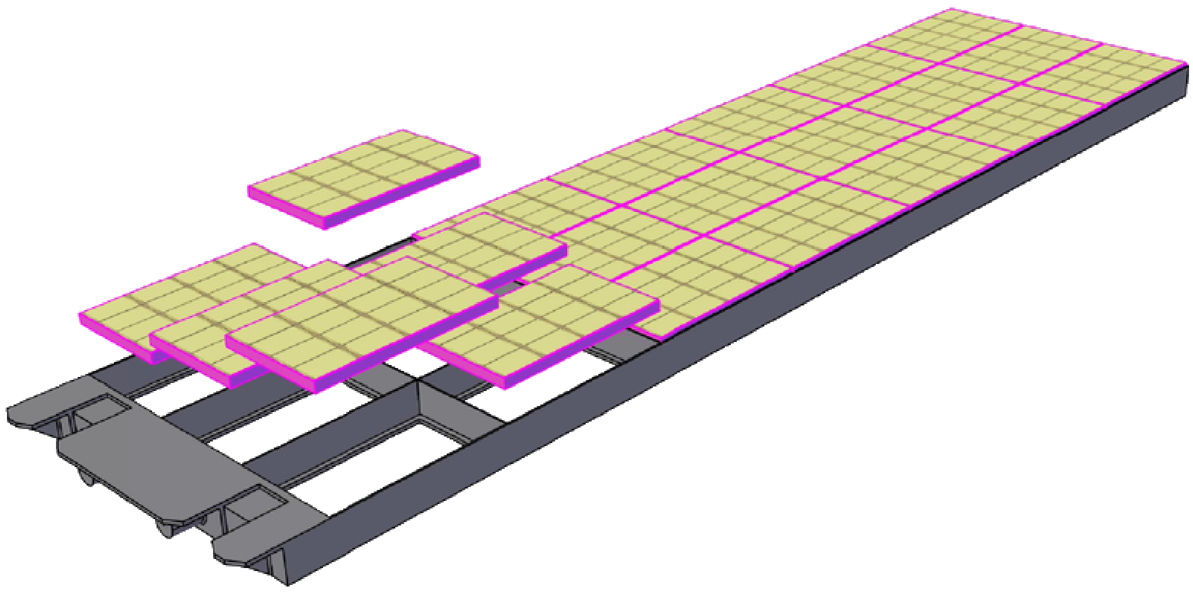


Figure 6.8
LAD Panel Layout: 21 modules are mounted on one panel. Each panel also carries a Panel-Back-End-Electronics (not shown here) that collects event data from all 21 Module-Back-End-Electronics. Small alignment errors between the modules are a desired effect since they will flatten the otherwise sharp, triangular collimator response. Source: ESA (2013).

Table 6.3
LOFT Silicon Drift Detector specifications for the LAD and the WFM.

Parameter	LAD	WFM
Active Area	108.5 mm x 70.0 mm	65.1 mm x 70.0 mm
Anode Pitch	970 µm	145 µm
Number of Anodes	2 x 212	2 x 448
Anode Capacitance	350 fF	85 fF

6.6.1 Silicon Drift Detectors

The detectors for the LAD and WFM are large-area Silicon Drift Detectors (SDD). The charge generated by the absorption of an X-ray photon is collected in the middle plane of the detector (thickness is 450 µm) and then drifted towards the read-out anodes at the edge of the detector by an electric field sustained through a series of cathodes on both faces of the detector. While drifting to the anodes, the charge cloud widens due to diffusion. The size of the cloud (which is basically energy-independent) reaching the anodes depends on the absorption point. For LOFT typical parameters (drift field 1300 V) the charge cloud reaches a maximum size of about 1 mm, when the photon is absorbed at the bottom of the 35 mm long drift channel. The drift time is about 7 µs. The working principle is depicted in Figure 6.9.

When the charge cloud is collected on more than one anode the signal-to-noise worsens, as the same charge has to confront the read-out noise of multiple anodes. On the other hand, it also encodes the distance to the absorption point, enabling a 2D position resolution (fine in the anode direction and coarse in the drift direction). For a non-imaging application as the LAD a large anode pitch is more favorable: an optimization including all relevant and read-out power parameters, resulted in a 970 µm anode pitch. With this choice, 40 % of the events are completely read-out on 1 anode (singles, with 200 eV energy resolution) and 60 % on 2 anodes (doubles, with 260 eV energy resolution) meeting the overall requirement of 240 eV.

In the WFM detector a wider charge division enables a higher positional accuracy, at the expenses of a larger noise and read-out power. A 145 µm anode pitch was selected. The size of the detector for the LAD optimizes the ratio between the active and geometric area (the integrated voltage divider has a fixed size, while the active area is a repetition of the anode pattern) for a 6-inch Si wafer, while keeping the drift length at 35 mm to preserve the heritage of the ALICE detectors. In Table 6.3 the main detector design parameters for the LAD and WFM are summarized. The WFM detectors are smaller and nearly squared, to optimize imaging.

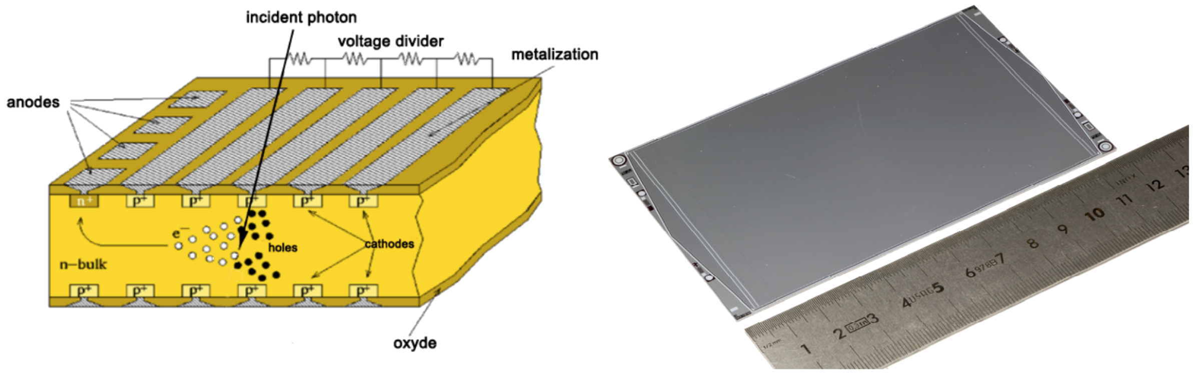


Figure 6.9
Left: working principle of the large-area SDD: charge generated by the absorption of an X-ray photon is collected in the middle plane and drifted side-ways to the read-out anodes.
Right: one LAD detector (full size). The lateral triangular structures identify the built-in voltage divider sustaining the drift of the charge. Source: ESA (2013).

6.6.2 ASICs

The front-end read-out of the LAD and WFM detectors will be based on mixed-signal ASIC technology. As the LAD and WFM uses the same type of detector, the ASIC design is the same, with the only differences being related to the different anode pitch, and consequently stray capacitance and leakage current. Adapting the LAD ASIC pre-amplifier to the lower leakage current and capacitance of the smaller WFM anodes allows to directly meet the noise requirements.

The complexity in the ASIC requirements comes mostly from the combination of low-noise and low-power on a mixed-signal ASIC. The number of channels and pitch are also lower when compared to ASICs equipping past experiments. The ASIC for LOFT has an outstanding heritage from the ESA StarX32 ASIC project offering a mixed-signal ASIC with about 18 - 19 e- rms noise (no detector) and 500 µW/channel power consumption in a 1024-channel matrix.

A working ASIC prototype, the Sirius V1, was produced by Dolphin Integration in Grenoble, France. This prototype has 4 read-out channels with 145 µm pitch and 4 read-out channels with 970 µm pitch. The performance of this ASIC prototype almost meets the LOFT requirements. The middle plot in Figure 6.10 shows the noise performance as a function of the temperature for a detector with no load. The power consumption during these tests also stayed well within performance.

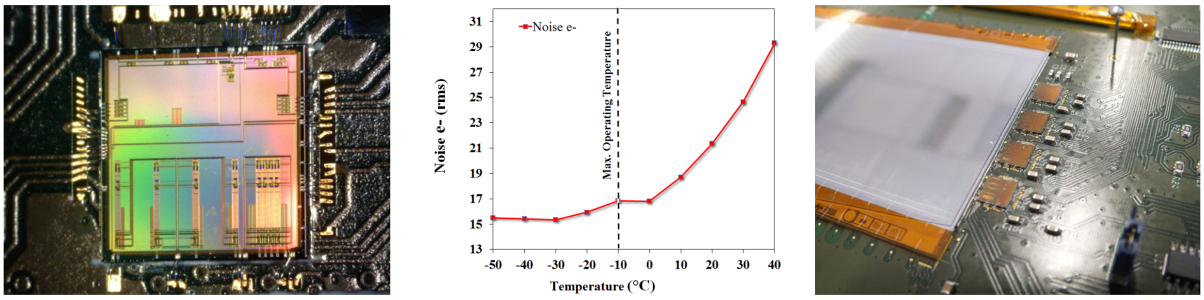


Figure 6.10
ASIC development for LOFT. Left: the Sirius ASIC V1. Middle: the performance as function of temperature. Right: the ASIC bonded to the SDD prototype. Source: ESA (2013).

6.6.3 Collimator

While the LAD is a non-imaging instrument, the unambiguous identification of a source is achieved by narrowing the field of view (FOV) of the detectors to 1° by placing a lead-glass capillary plate (CP) aperture collimator in front of each Silicon Drift Detector. The lower limit of the FOV is given by the requirements to to allow for pointing uncertainties, the upper limit by the reduction of the cosmic diffuse X-ray background and the risk of source confusion.

The collimators are composed of a 5 mm thick sheet of lead-glass, perforated by a huge number of micro-pores with a diameter of 83 µm and a wall thickness of 16 µm. The stopping power of the lead glass allows the collimators to effectively collimate X-rays below 30 keV.[2] A collimator prototype has been manufactured by Photonis (shown in Figure 6.11) which has also manufactured a similar MCP for the BepiColombo mission (MIXS instrument). The collimator requirements and the properties of the prototype are given in Table 6.4.

[2] The energy range from 30 keV to 80 keV is used only for exceedingly bright events from outside the instrument field of view, shining through the collimator walls.

Table 6.4
LOFT collimator specifications, as compared to BepiColombo/MIXS and the specific LOFT collimator prototype already available. All types use the same substrate (Lead glass, Philips 3502), and have the same pore-to-pore alignment (1 arcmin) and pore to surface alignment (1 arcmin).

Item	LOFT Requirement	LOFT Prototype	BepiColombo / MIXS
Unit Size	111 x 72.5 mm	50 x 50 mm	40 x 40 mm
Plate Thickness	5 mm	6 mm	0.8 - 2.5 mm
Aspect Ratio	60:1	60:1	55:1
Pore Size	83 µm, squared	100 µm, squared	20 µm, squared
Wall Thickness	16 µm	15 µm	6 µm
Open Area Ratio	70 %	60 %	60 %
Units	2016 (flight units)	2	40 (flight units)

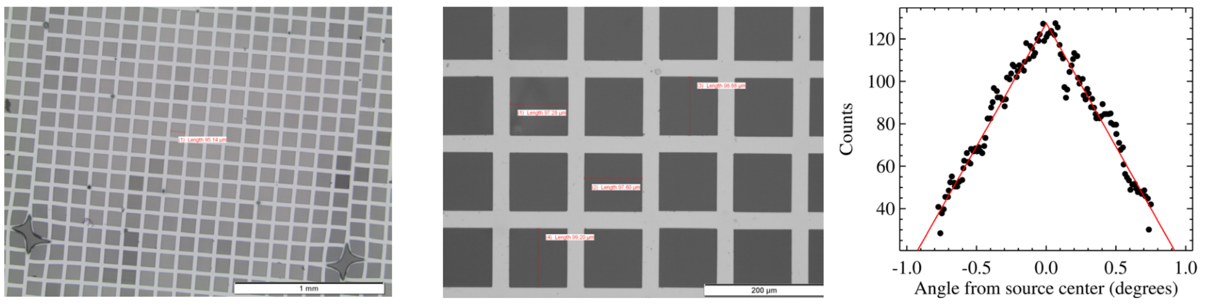


Figure 6.11
Left and middle: collimator produced at Photonis shown with different magnification. Right: measured response to 22 keV photons on the LAD CP prototype (6 mm thick, 100 µm pore). Source: ESA (2013).

6.6.4 MBEE

The MBEEs main task is to command the ASICs (Application-Specific Integrated Circuits) that are connected to the SDDs (Silicon Drift Detectors) and to receive and process events. While the MBEE sends calibration and configuration data as well as commands to the ASICs it primarily reads out the anodes of the ASICs after an X-ray photon has hit the detector to determine the total charge deposited and reconstruct the photon energy. The energy together with a corresponding timestamp and optionally information about the location of the detection is then combined into an "event" and sent to the PBEE.

Event Reconstruction

The MBEE reconstruction pipeline was implemented in VHDL and operated in an MBEE prototype at the IAAT as one distribution to the LOFT project. This is a short summary of the pipeline processing, a more detailed description can be found in *Development of the Module Back End Electronics for the Large Observatory For X-ray Timing* (Uter 2013).

1. **Time Tagging**
   For a timing mission it is crucial that each event is precisely tagged with the time of its occurrence. A 20-bit timestamp allows time tagging with microsecond resolution. The MBEE receives a stable 1 MHz clock from the PBEE which is synchronized once per second with a GPS Pulse Per Second signal distributed by the Data Handling Unit through the PBEE to all MBEEs.

2. **Trigger Validation and Filtering**
   Trigger can not only be caused by incident photons but also by minimum ionizing particles (MIPS), which makes it necessary to validate the trigger first. When a trigger occurs, the whole detector half initially is handled as one single unit. After a short coincidence window, all 7 ASICs are requested to send their 16-bit trigger map (1 bit per anode) to the MBEE, which generates an overall trigger map of 112 bits. Each bit stands for one of the 112 anodes of the detector half and indicates whether the respective anode has triggered. The event is considered valid if only one anode alone or two adjacent anodes had triggered. Otherwise the trigger is most likely caused by MIPS and will be rejected. If the trigger map is valid the MBEE commands the triggered ASICs to perform the A/D conversion and to send all the anode values to the MBEE.

3. **Pedestal Subtraction**
   Due to manufacturing tolerances each anode has a different pedestal value. As a first processing step the respective pedestal value is subtracted from each anode

value. Therefore one corresponding pedestal value for each anode is stored in the MBEE. These values can be updated by on-board calibration or by telecommand from ground.

4. **Common Mode Subtraction**
   Small variations in the supply voltage of the FEE (Front-End-Electronics) cause an offset, which is the same for all anodes of one ASIC. The common mode offset is calculated as the median of all non-triggered anodes from the respective ASIC and is subtracted from all its anode values.

5. **Gain Correction**
   The gain varies slightly from anode to anode due to manufacturing tolerances and is also temperature dependent. There are gain correction values for a certain calibrated temperature for each anode stored in the MBEE. The difference to the actual temperature is accounted for by a linear correction procedure. The storage values can be updated by on-board calibration or by telecommand from ground. The local temperature of the instrument is provided by a separate housekeeping board.

6. **Energy Reconstruction and Threshold**
   For a two-anodes event the energy must be reconstructed by adding up the two previously corrected anode values. A check is performed whether the reconstructed energy value is within the valid energy range of 2 keV to 80 keV. Otherwise the event will be marked as invalid and discarded later.

7. **Event Formatting**
   In the last step an event packet is built, which is then sent to the PBEE. The event packet consists of a 9-bit energy value, a 20-bit timestamp, and a 3-bit event ID which distinguishes between one- or two-anode events and non-science packets.

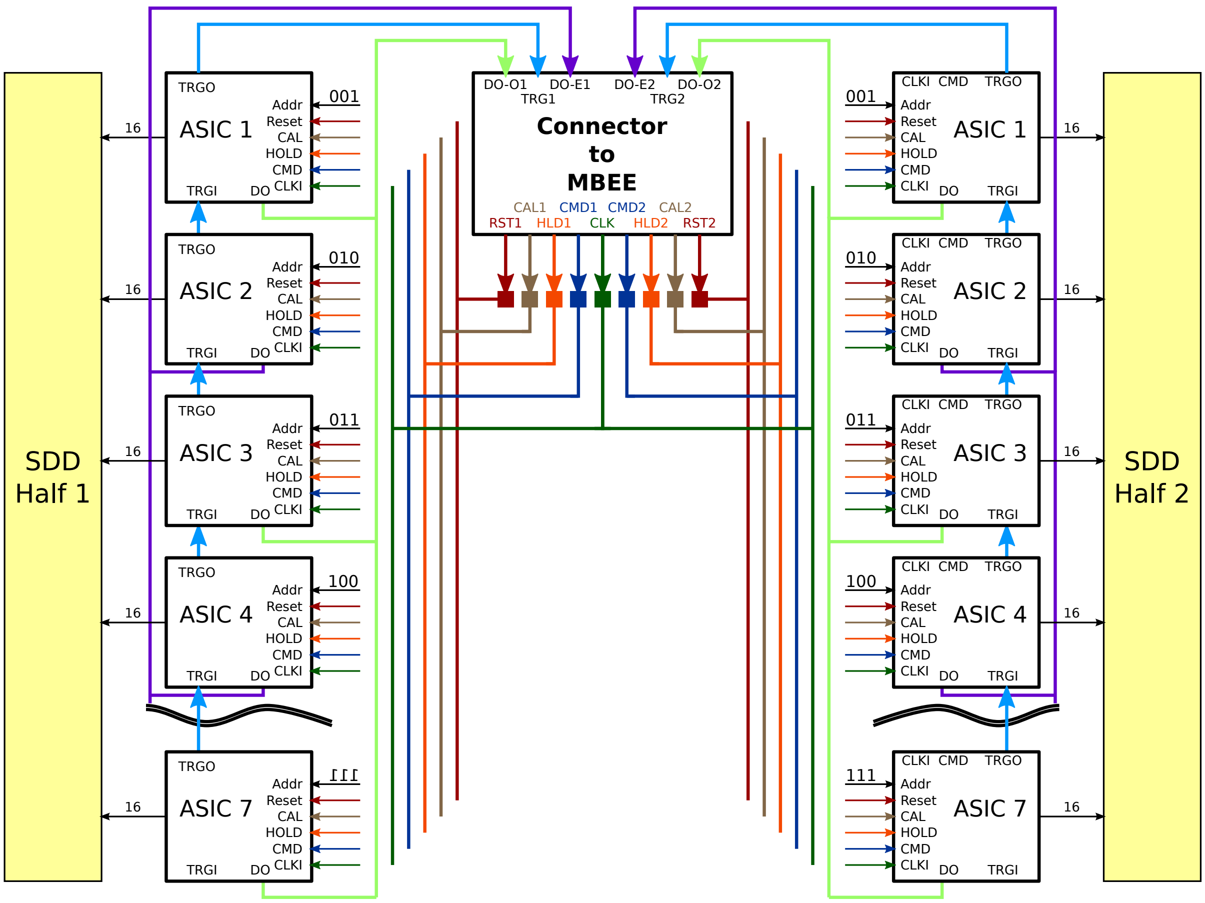


Figure 6.12
The diagram shows the interface lines on the FEE. Note the symmetry: One half of the signal lines is connected to the left detector half, the other half of the signal lines is connected to the right detector half. only the CLK signal is shared. See Table 3.1 for an explanation of the signals. Source: Yannick Favre, University of Geneva (2013).

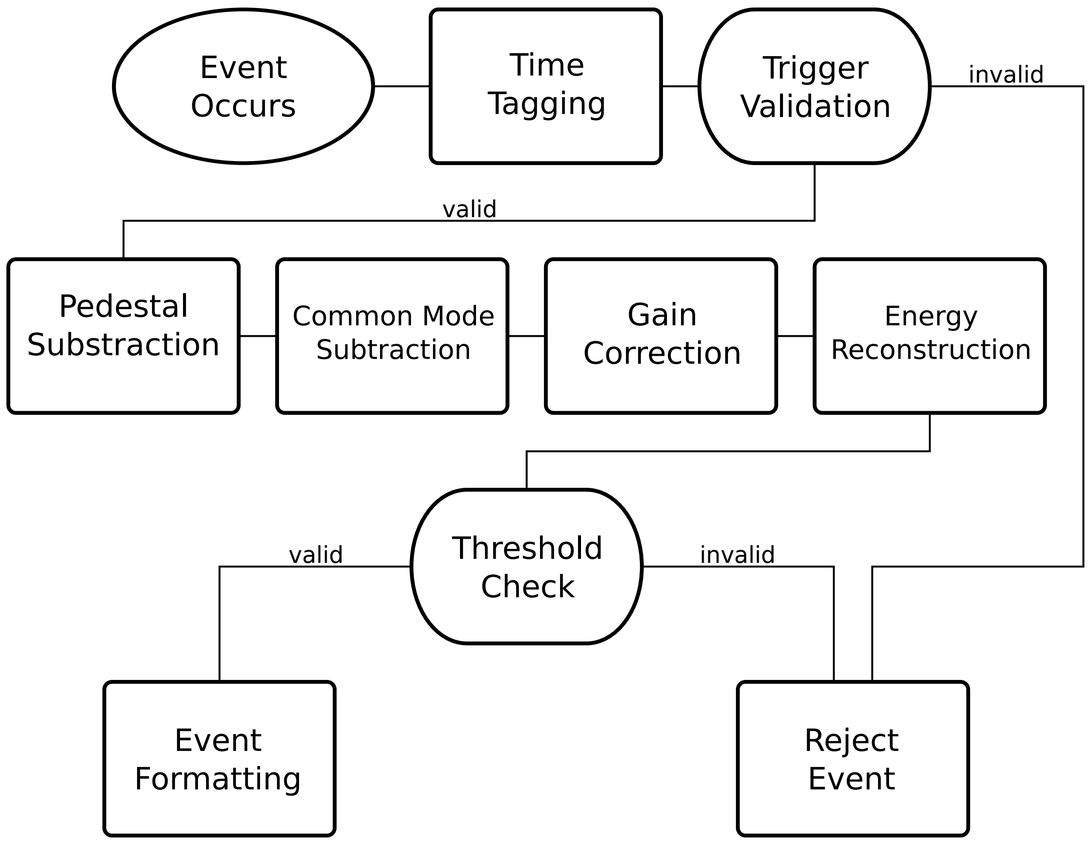


Figure 6.13
Flowchart of the MBEE Event Pipeline. The single steps of this chart are described in Chapter 6.6.4 - Event Reconstruction starting on page 98. After an event has passed the threshold check and has been formatted and prepared for sending, it is handed to the PBEE. Original figure: Uter (2013).

6.6.5 PBEE

After an event has been successfully detected and its energy has been reconstructed the event package is sent to the PBEE. The PBEE is connected to 21 MBEEs and can communicate to all of them at the same time, using 21 specialized parallel interfaces.

The PBEE implements the two main modes of the LAD instrument.

- **Event Mode**
  In this mode events are sorted chronologically, formatted as event stream, transferred to the DHU, and eventually sent to ground. This mode is used for sources with a brightness of up to 1 Crab.
- **Spectrum Mode**
  While for brighter sources the MBEE can still detect all single events and sent them to the PBEE, a transfer to ground is no longer possible due to the limiting telemetry bandwidth. Therefore events are "collected" by the PBEE for a configurable time span and integrated into a spectrum. This can - depending on the configuration of the spectrum - decrease the necessary bandwidth significantly.

Additional tasks of the PBEE include receiving commands from the Data Handling Unit (DHU) and distributing them to the MBEEs, distributing a stable 1 MHz clock and collecting housekeeping data from monitoring boards, integrated into the MBEE.

The development and testing of a PBEE prototype board was done as part of this thesis. Therefore the PBEE is presented in much more detail in Chapter 6.7 - PBEE Prototype Board starting on page 104.

6.6.6 DHU

Data is handed from the PBEE to the instruments Data Handling Uni (DHU), an FPGA based LEON3 microprocessor (see also Chapter 4.3 for an in-depth discussion of the LEON3). The interface used is a standard SpaceWire interface. The current design allows for the SpaceWire receiver to be located in the DHU but it is also possible to locate dedicated receivers directly at the panel hinges to reduce complexity of the cabling. The DHU also provides the Interface Control Unit (ICU) to the satellite bus.

The tasks of the DHU include: control of the overall instrument performance by analyzing the housekeeping data collected by the PBEEs, instrument configuration and mode selection, data compression and packaging of the scientific data into a telemetry stream suited for transmission, interfacing the mass memory to store data until it be transmitted to the ground, power supply of the PBEEs and MBEEs, clock distribution to the PBEEs. An overview of the hierarchical layout is shown in Figure 6.14.

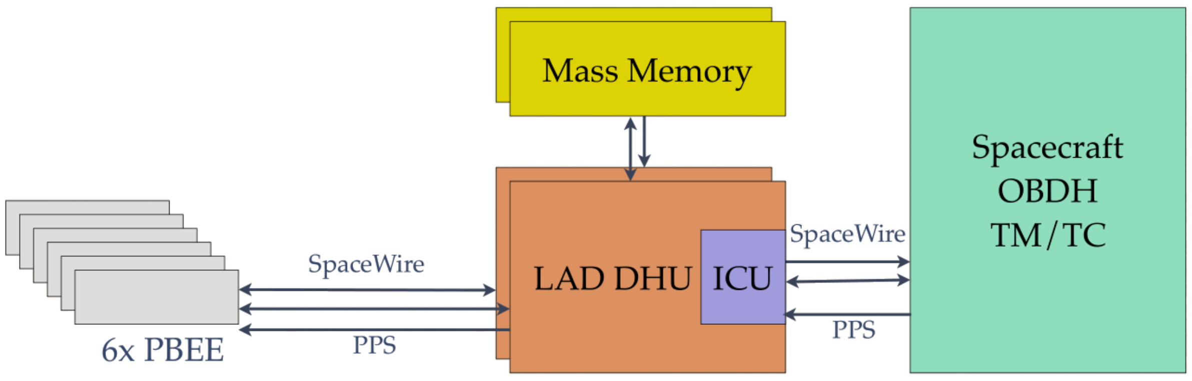


Figure 6.14
The six PBEEs are connected via SpaceWire to the Data Handling Unit (DHU). The DHU has access to a mass memory to store scientific data that is later handed to the On-Board Data Handling Unit (OBDH). Source: Tenzer (2012).

6.7 PBEE Prototype Board

Since the Large Area Detector has a geometric collecting area of 10 m^2 filled with 126 detector modules, each containing 16 Silicon Drift Detectors an intermediate stage between the modules and the Data Handling Unit (the instrument's main computer) is needed. To meet the requirement of being able to detect up to 28.570 events per second (@ 15 Crab) on each of the 126 modules in parallel, groups of 21 MBEEs are connected to and operated by one Panel-Back-End-Electronics (PBEE). The LAD has one PBEE on each of the six panels. Refer to Figure 6.7 and Figure 6.8 on page 93 for an overview of the module and panel layout.

Part of the work of this thesis was the development of a PBEE prototype board and a test setup to verify the function and the performance of the custom designed interface between the MBEE and the PBEE. Section 6.7.1 covers the bi-directional point-to-point custom interface and the communication protocol. Section 6.7.2 covers the operational modes and the internal processing steps of the PBEE and the subsequent Chapter 6.8 presents the prototype hardware and test setup built in Tübingen.

6.7.1 Interfaces

Each PBEE communicates with the 21 MBEEs located on the same panel via our custom interface, and with the DHU in the center of the LAD instrument via SpaceWire. It is also connected to a second DHU, the cold redundancy backup, so that communication can still be provided in case of a DHU system failure.

Housekeeping boards integrated into the MBEEs relay their communication through the MBEE-PBEE interface, too. The PBEE is also connected to a dedicated clock distribution network, to provide all 21 MBEEs with a synchronous one second timer reset that is used to generate the event timestamp inside the MBEE.

PBEE to DHU

The interface between PBEE and DHU is a standard SpaceWire interface as described in Chapter 4.4.1. The expected bandwidths between each PBEE and the DHU for a dim/bright source are:

PBEE to DHU bandwidth

```
500 mCrab source:   19,950 events/s = 480 kbit/s
 15 Crab source:   600,000 events/s = 14 Mbit/s
```

The PBEE Virtex 4 FPGA is connected to external 2.5 V LVDS-driver and uses a single-ended SpaceWire core to implement the interface to the DHU. The baseline design allows for either a direct point-to-point connection to the DHU but also for a routed connection through dedicated SpaceWire hub that is connected to all six panels as well as the DHU.

The 1 Hz clock signal that is distributed from the DHU through the PBEEs to the MBEEs and that is needed to reset the internal 1 MHz counter used for time-tagging the events is routed through a dedicated clock network that is not part of the SpaceWire interface.

MBEE to PBEE

Using SpaceWire also for the interface between MBEE and PBEE has the drawback that the total harness per LAD panel reaches a weight of up to 100 kg which increases instrument mass beyond acceptable levels. Due to its small size this RTAX FPGA used in the MBEE also does not support the implementation of a fully functional SpaceWire interface between the PBEE and the MBEE. Additional SpaceWire hardware components would have increased weight, complexity and power consumption of the MBEE.

Therefore a specialized interface with 40 bit data packets is used to transfer the event and housekeeping data from the MBEE to the PBEE and to send telecommands from the DHU via the PBEE to the MBEE. The MBEE and PBEE prototypes use a 9-pole Sub-D connector, where the data and clock signals are transferred with a low voltage differential signal (LVDS). In the final design, the communication will be performed with a 9 way CBR connector.

Communication is performed at 5 MHz which allows for to a transmission rate of up to 40k events/second/module. The housekeeping data is very small in comparison to the event data and can be neglected. The bandwidth provided by the interface will be sufficient for sources with a brightness of up to 20 Crab. Bandwidths for a typical dim source and the brightest source intensity LOFT is specified to observe are:

MBEE to PBEE bandwidth

500 mCrab source: 950 events/s = **23 kbit/s**

15 Crab source: 28,570 events/s = **886 kbit/s**

Event Packaging (MBEE to PBEE)

The MBEE creates different types of data packets for an event, depending on its operational mode. All data packets always have a size of 40 bit, with an 8 bit header and a 32 bit body. The different data packets are:

- Scientific Mode: 40 bit/event with 20 bit time, 9 bit energy, 3 bit ID, and 8 bit header.
- Engineering Mode: The first packet will be a regular scientific event (40 bit), followed by a number of packets (40 bit each) with the anode values and the ASIC numbers containing the raw data of the event.
- Calibration Mode: Multiple 40 bit packets, depending on the calibration program.
- Timer Overflow Event: If the 1 MHz event timer reaches its maximum, a timer overflow event will be sent to the PBEE. The distribution of a 1 Hz clock reset signal from the PBEE to the MBEE should prevent a timer overflow.
- Housekeeping Event: At regular intervals housekeeping data is sent, identified by its 8 bit header.

Sending Events (MBEE to PBEE)

The communication between the MBEE and the PBEE is performed in a master-slave scheme, where the MBEE is the master at all times. Every communication is initiated by the MBEE by sending a single HIGH bit followed immediately by the 40 bit data packet. Then the MBEE waits for a response which is triggered by the PBEE by sending a HIGH bit followed by the 40 bit response. A flowchart of this scheme is presented in Figure 6.15.

Most of the time (when observing) detector events are sent to the PBEE. If no events are detected or processed by the MBEE housekeeping events are sent to the PBEE at regular intervals. These are not timer overflow events as described in the section before, but regular events that contain housekeeping data. Each packet sent to the PBEE is usually acknowledged with a 40 bit ACK response packet. If the MBEE operates in engineering mode a data stream is sent to the PBEE with every single packet of the stream still acknowledged by the PBEE.

Each packet sent in either direction is followed by a parity bit and a parity check is performed when the packet is received on either side. If the PBEE detects a parity bit error, an error response is sent instead of ACK to the MBEE. The MBEE can then decide to either send the last packet again or to discard the event in the case that a new event is already awaiting transmission.

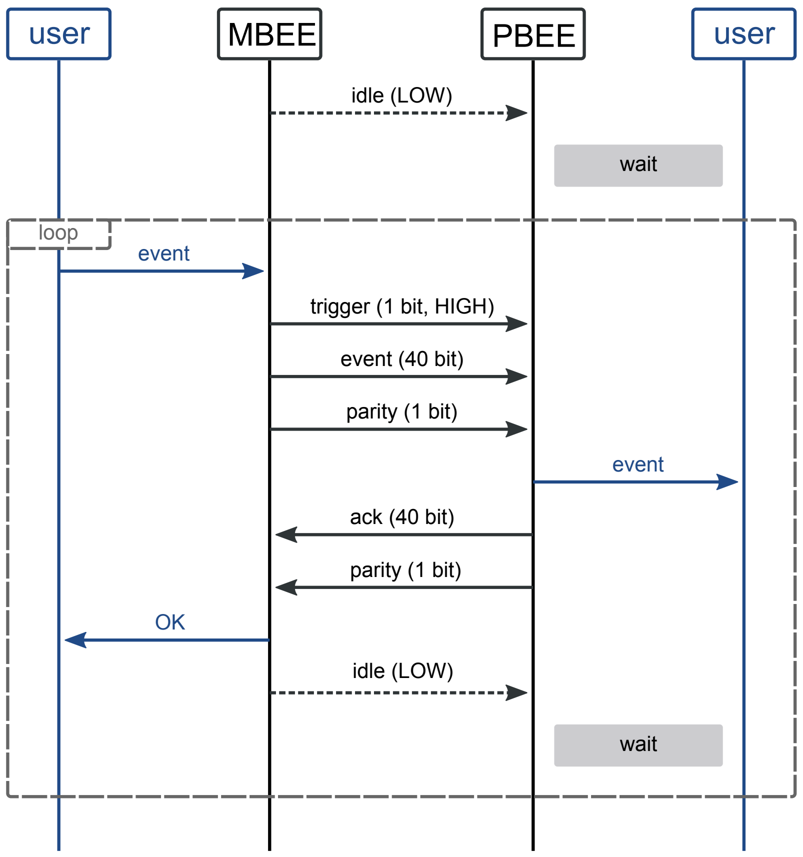


Figure 6.15
MBEE sending events to the PBEE. The line is idle (LOW) until triggered by the MBEE. The event is transmitted followed by a parity bit. If no regular events are transmitted, time events are sent to allow the PBEE the response with a command. See Figure 6.16 on page 109 for details on how to transmit data from PBEE to MBEE.

If the MBEE detects a parity bit error while receiving the ACK (or error response) from the PBEE, and therefore can not determine the state of the most recent transmission, it will retry sending the original event with an altered header so the PBEE can identify the situation. If a transmission fails a configurable number of times, the MBEE will enter a defined safe-mode and the PBEE will signal the situation to the DHU but is still able to continue normal operation and communication to the other MBEEs.

Sending Commands (PBEE to MBEE)

Whenever a command should be sent from the PBEE to the MBEE the direction of transmission has to be reversed. Since the MBEE is the master at all times the PBEE first has to wait for a packet sent by the MBEE. This could be either a scientific event or e.g. a housekeeping event, such events are sent at regular intervals.

Instead of responding with an ACK response packet the PBEE sends a special command telling the MBEE to reverse the direction of the communication. The MBEE acknowledges this command with an ACK response and from that point on the PBEE can send packets to the MBEE while the MBEE will respond with ACK.

It is important to note that the master slave scheme of the communication is still in place. Technically the communication is triggered by sending a high bit from the MBEE to the PBEE, but this high pit is part of the ACK response to the last command. The PBEE then "responds" by sending the next command. A flowchart of this scheme is presented in Figure 6.16.

If the transfer of data or commands from the PBEE to the MBEE is complete the PBEE sends a dedicated command telling the MBEE that the direction is again reversed. The MBEE does not respond to this command but simply waits until the time reserved for both units waiting for the MBEEs response has passed. When this happens, the PBEE and the MBEE reset the logical direction of the transmission and the MBEE starts sending regular events or time/housekeeping events again.

The advantage of using this mechanism of reversing the direction of the communication is that only two lines are needed for the interface. A clock line that is provided by the PBEE and that is also used to synchronize the internal FPGA clock domains of the PBEE and the MBEE and one data line for the communication. Considering that a total of six PBEEs are connected to a total of 126 MBEEs this considerably reduces the instrument's harness complexity and weight.

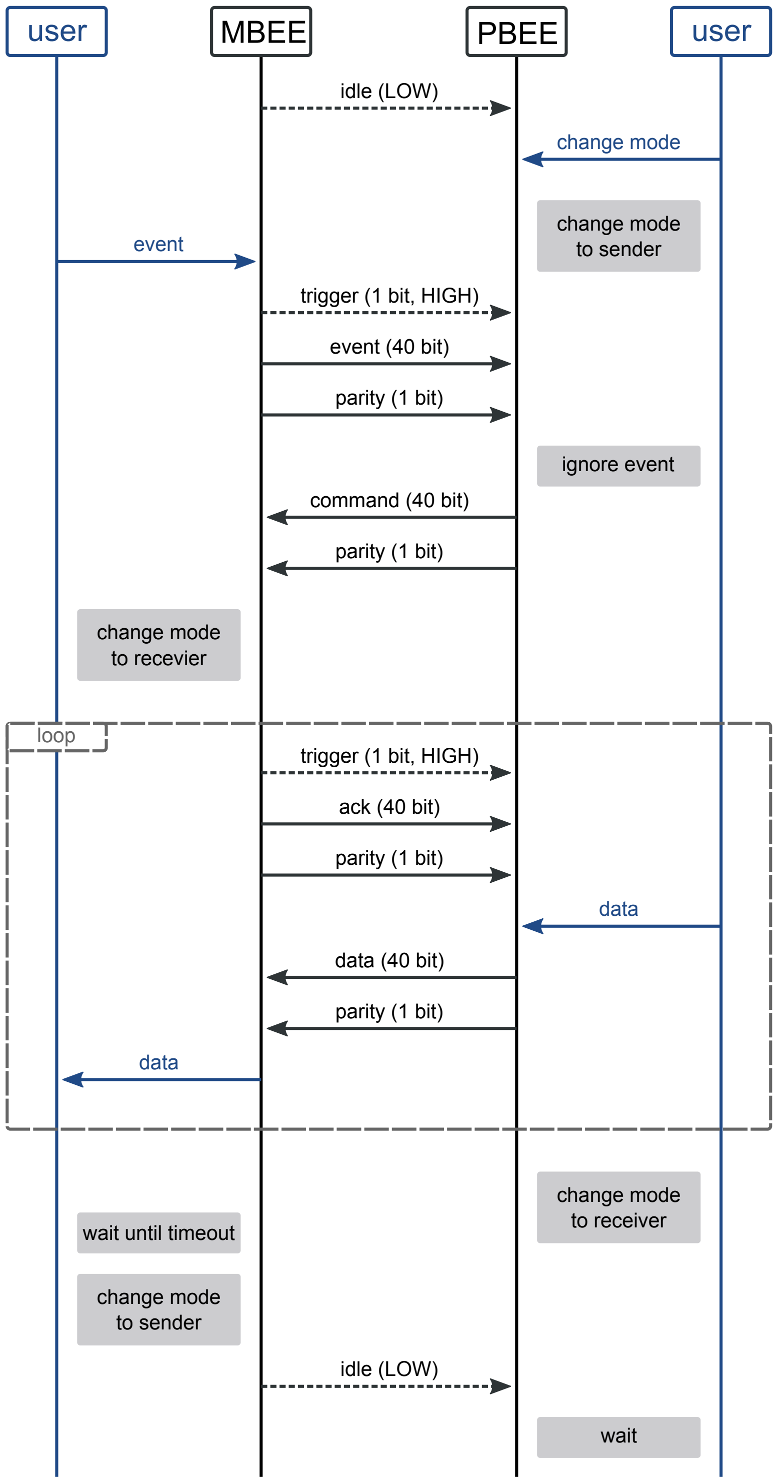


Figure 6.16
After reversing the logical direction of the transfer, the PBEE can respond to the MBEEs ACK packet with the next data packet.

6.7.2 Event Processing

All 2016 Silicon Drift Detectors operate independently and can detect X-ray photons at all times, except for a very short dead-time of 100 ns after the detection of the previous event. The MBEE reads out the event, processes it as described in Chapter 6.6.4 and then sends it to the PBEE. If multiple events are detected in parallel on multiple detectors, the MBEE can process them in parallel, buffers the events and sends them to the PBEE in a serial fashion. The maximum number of events that can be processed is 40,000 events/s while the LOFT mission requirement is 28,570 events/s for a 15 Crab source.

The PBEE receives events from each individual MBEE by a dedicated point-to-point connection and buffers the incoming events for a short time. It selects the next event in chronological order by comparing the relative timestamps (since the last 1 Hz clock reset signal). The timestamp of the event is removed an replaced by a relative time to the last event to reduce the size of the timing information from 20 bit to e.g. 12 bit. The reduction is configurable but requires the creation of frame events that can be used as a relative reference.

To ensure data consistency even over the corruption of single events and in the case that no new events arrive for some time, special frame events that contain an absolute time stamp and an unambiguous panel identifier are automatically inserted into the data stream. This procedure is not repeated on the level of the DHU where the six data streams from all panels are merged together and processed for telemetry.

Timestamp Sorting

Events processed by the 21 MBEEs are sent to the PBEE but do not necessarily arrive in chronological order. Therefore the PBEE implements 21 FIFO buffers to store received events. Two conditions can trigger the processing of the 21 buffers. As soon as all buffers contain at least one event, the “next” event can be processed. The other condition arises if one of the buffers contains a given number of events and it is therefore considered unlikely that any event that has not yet arrived at the PBEE might have an earlier timestamp.

whenever the processing of the next event is triggered the timestamps of all events are compared and the next event is integrated into the event stream to be handed to the DHU. Integration into the event stream includes the conversion of the MBEE timestamp into a relative time to the preceding event.

6.8 LAD @ IAAT

The Institute for Astronomy and Astrophysics (IAAT) of the University of Tübingen is contributing in several ways to the LOFT mission. One project in which the author of this work was involved was the definition, design and development of the Panel-Back-End-Electronics (PBEE) and the Module-Back-End-Electronics (MBEE) of the LAD instrument together with the interfaces and protocols used for communication between these units.

A functional description of the MBEE board has been presented in Section 6.6.4. The development and the hardware of the MBEE prototype PCB is presented in Section 6.8.1. Similarly the PBEE tasks and operations have been presented in Section 6.6.5 and more detailed information on the interfaces and data handling procedures can be found in Section 6.7. The PBEE prototype PCB the hardware components are presented in Chapter 6.8.2.

The test setup used to validate the designs and the results that were obtained as part of the work of this thesis are presented in Section 6.8.3.

6.8.1 MBEE Prototype

One contribution to the LOFT mission at the IAAT was the development, design and building of the MBEE prototype which is shown in Figure 6.18. Four detectors or detector simulators can be attached to the MBEE prototype board via four SUB-D 25 connectors. The prototype board also contains a Spartan 3 that is used as an on-board ASIC simulator and is able to simulate the FEE–ASICs of four detectors. An USB port is connected to the ASIC simulator, which allows to feed external event data into it.

The clock signal that can be received by the board via a single ended connection or via LVDS is distributed by an LVDS clock buffer to the RTAX-Adapter that contains the MBEE, the Spartan 3 FPGA (ASIC simulator), and the four external FEE interfaces simultaneously. The clock buffer has two selectable inputs, one is connected to a 40 MHz oscillator included in the PCB design. The other one is connected to the SUB-D 9 connector, which is used to interface the PBEE.

Figure 6.17 shows the PCB layout and Table 6.5 describes the six PCB layers of the MBEE prototype board. The top and bottom layers are used to connect single ended signals, whereas the differential signals (LVDS) are routed in the inner two layers, which are shielded by a ground layer beneath and above. Using two layers for differential signals allows to route the two signal lines of one pair on top of each other, which is reducing the electromagnetic susceptibility and ensuring that both lines have the same length and signal travel time.

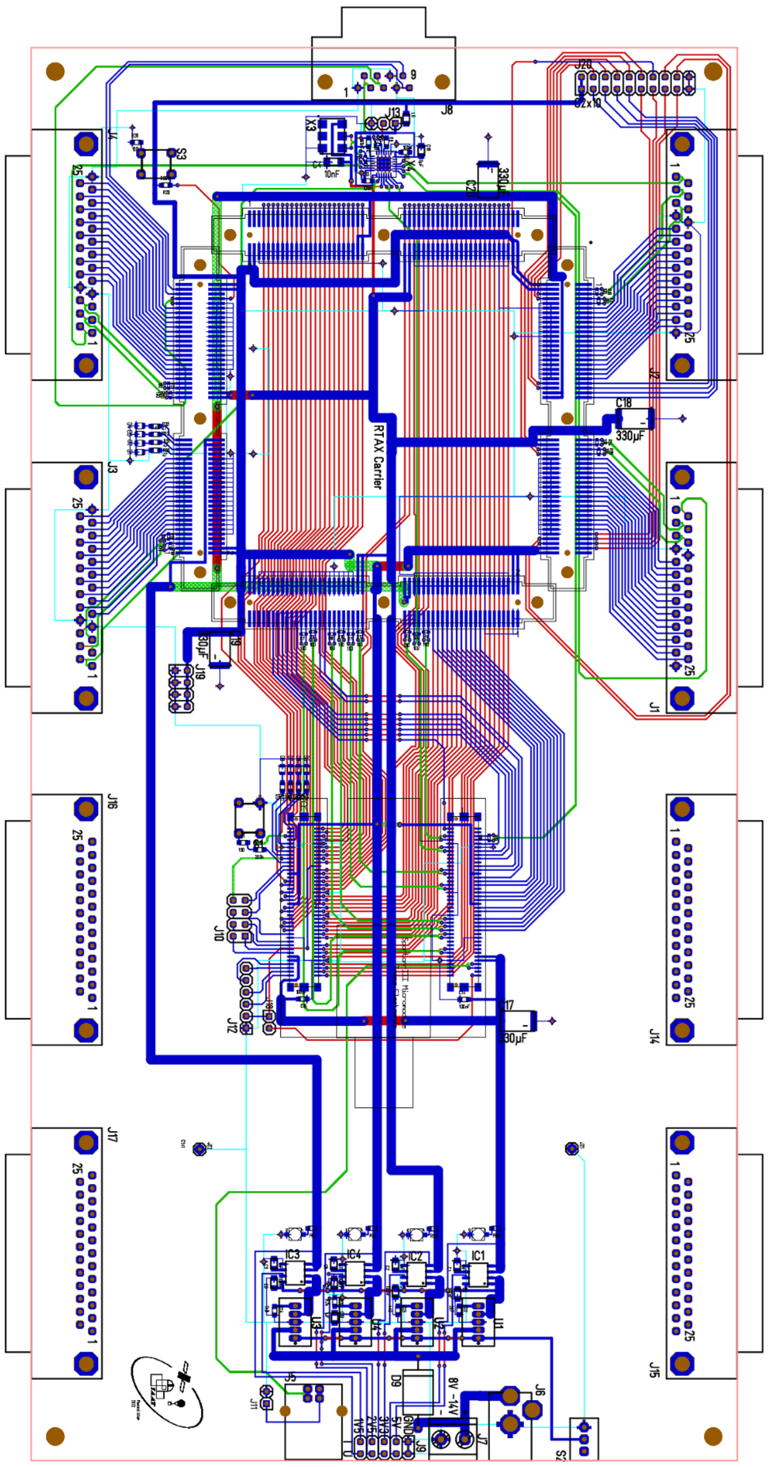


Figure 6.17
MBEE Prototype PCB Layout. Layers: 1 (TOP) blue, 2 (GND) hidden, 3 (LVDS_P) green, 4 (LVDS_N) brown, 5 (GND) hidden, 6 (BOT) red. Size: 150 mm × 300 mm. Source: Uter (2013).

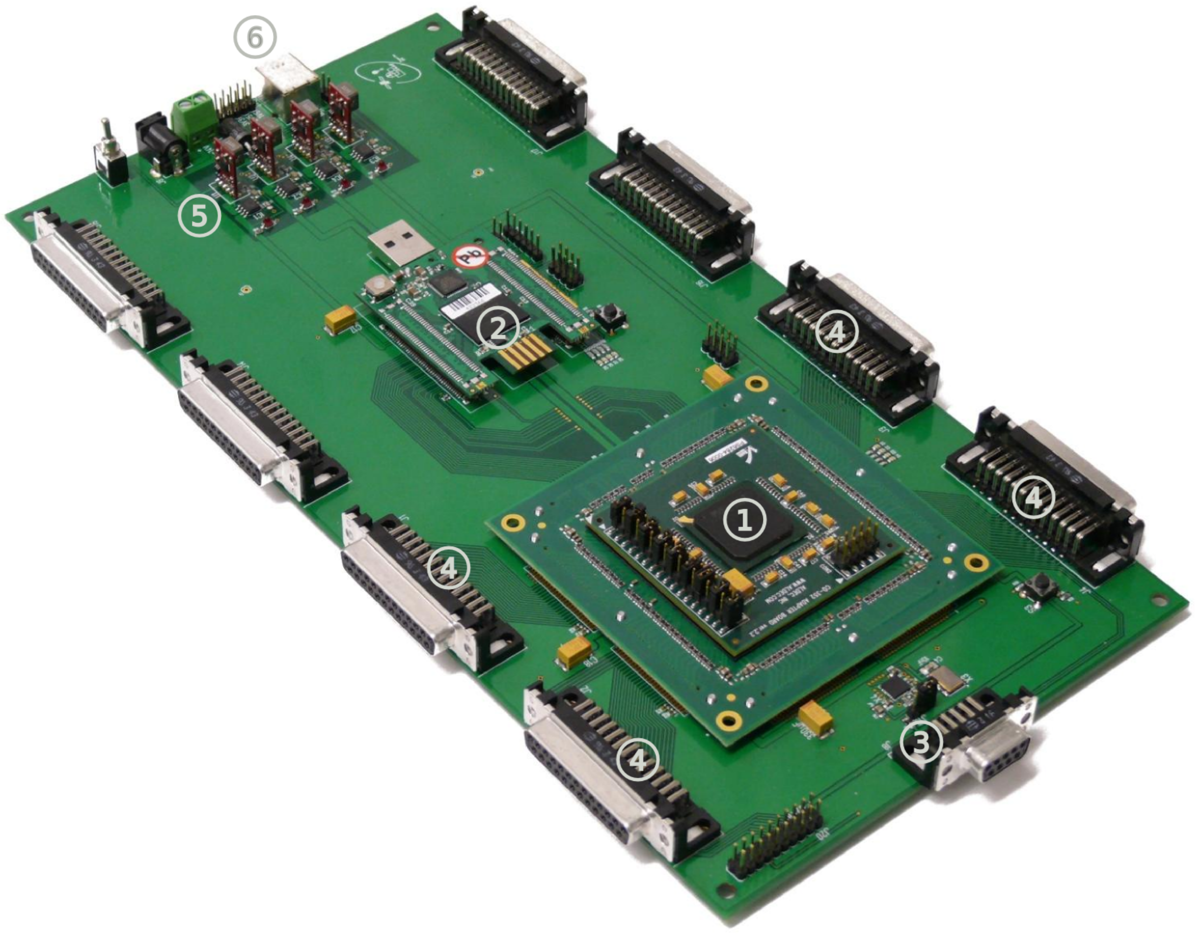


Figure 6.18
MBEE prototype board. 1: Pro-ASIC FPGA on the RTAX-Adapter; 2: Spartan 3 FPGA simulating the detector FEE; 3: PBEE interface connector; 4: FEE interface connectors; 5: on-board voltage generation; 6: USB port to load events into the FEE simulator. Source: Uter (2013).

Table 6.5
Layers of the MBEE Prototype PCB.

Layer	Name	Description
1	TOP	Power supply / connecting RTAX-Adapter with FEE interfaces.
2	GND	GND layer shielding the LVDS connections.
3	LVDS_P	Mainly used for LVDS connections.
4	LVDS_N	Mainly used for LVDS connections.
5	GND	GND layer shielding the LVDS connections.
6	BOT	Connects RTAX-Adapter with ASIC simulator.

6.8.2 PBEE Prototype

As part of the work for this thesis a PBEE reference design was developed, implemented in VHDL, and simulated in a VHDL test-bench. A PCB prototype board was designed, and built, carrying a Virtex 4 and a Spartan 3 FPGA. The PCB layout is shown in Figure 6.19 and the prototype board in Figure 6.20. Together with Michael Gschwender who continued the work on the MBEE after the MBEE prototype board was built both prototypes were connected and successfully demonstrated the custom interface and communication protocol by sending events from the MBEE to the PBEE and by transferring data and commands from the PBEE to the MBEE.

The PCB layout was created with Target 3001!, a CAD (Computer-Aided Design) program for PCB design and EDA (Electronic Design Automation, sometimes called ECAD) that is developed by the German company Ing.-Büro Friedrich in Eichenzell. Design files for several electronic parts used in the PCB design were created from scratch or imported from other projects created at our Institute in the past, for example, the QFSS Samtec sockets.

The design uses two layers, almost all signals are routed on the top layer while the bottom layer provides a common ground for all components. The board is designed for an external support voltage of 5 V, that is routed to the Virtex 4 and Spartan 3 add-on boards. All other components (LEDs, switches, drivers) use 2.5 V provided by an on-board voltage converter. The input and output banks of both FPGAs also operate with 2.5 V, including the I/O lines to the MBEE interface.

A Virtex 4 SX35 add-on-board from IAF GmbH is used to implement the PBEE in a Virtex 4 FPGA and is connected to the PCB via QFSS Samtec sockets. It is shown in Figure 6.21. A second FPGA, a Spartan 3 XC3S1000-4FT256C on a Spartan 3 FPGA Micromodule from Trenz Electronic GmbH is used to allow the PBEE to be tested with an on-board MBEE simulator (see Figure 6.22). This is the same module that is also used as a detector simulator on the MBEE prototype board.

The connector layout for the custom MBEE - PBEE interface that is used for our test setup is shown in Figure 6.23. For the prototype boards we used standard D-Sub connectors (DE-9). Since the Pro-ASIC FPGA used on the MBEE board does not support bi-directional communication on it's LVDS pins, we alternatively used to signal lines for uni-directional communication from MBEE to PBEE and vice versa. This imposes only minimal changes in the VHDL design since the translation from bi-directional to two uni-directional signals takes place in the topmost VHDL entity only.

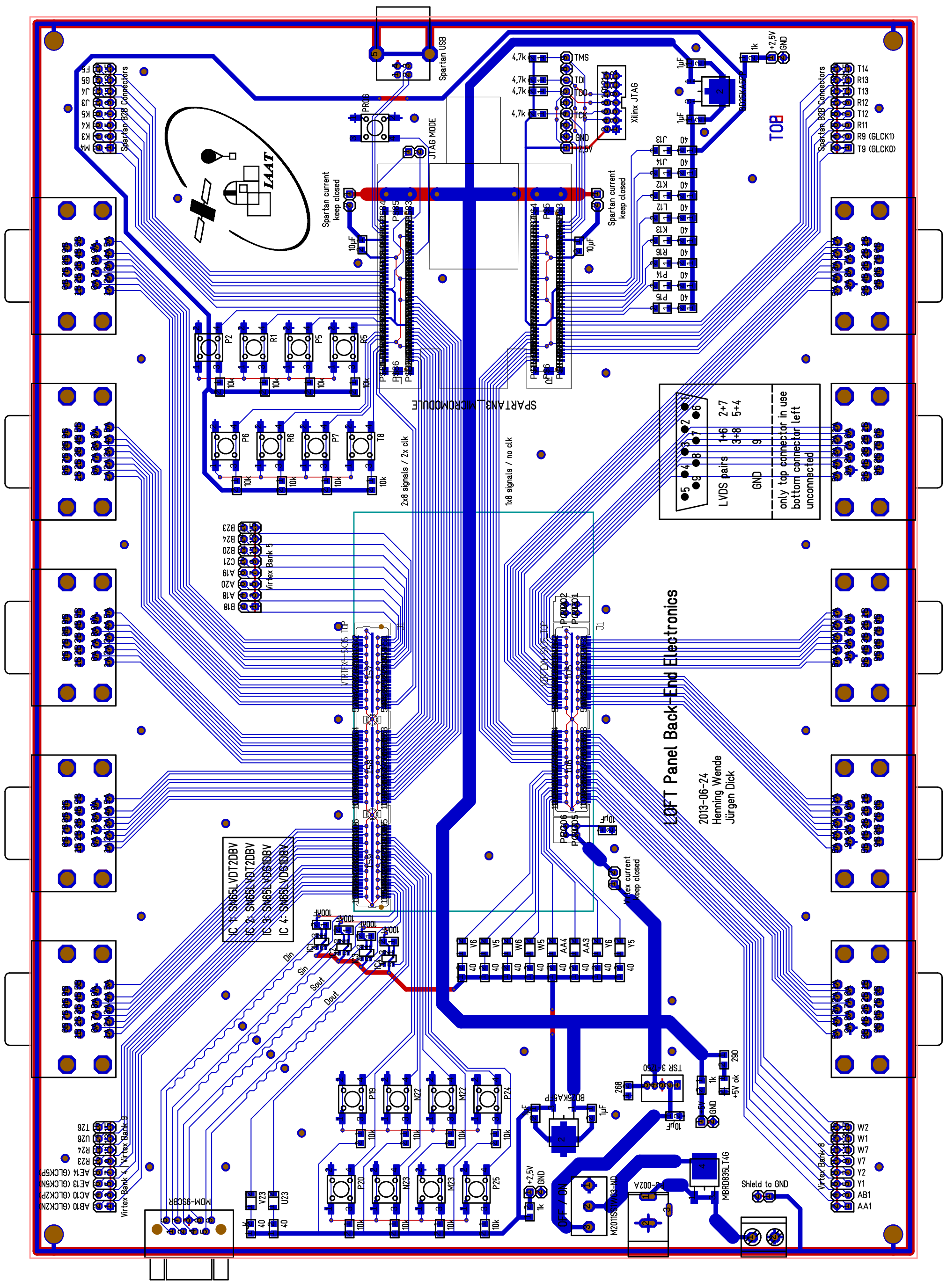


Figure 6.19
PBEE prototype PCB layout. The Virtex 4 with the PBEE design is located in the center of the board. A Spartan 3 FPGA is located above the Virtex 4 and is used to simulate two MBEEs that send stored events. Data can be transferred to the MBEE simulator via USB.

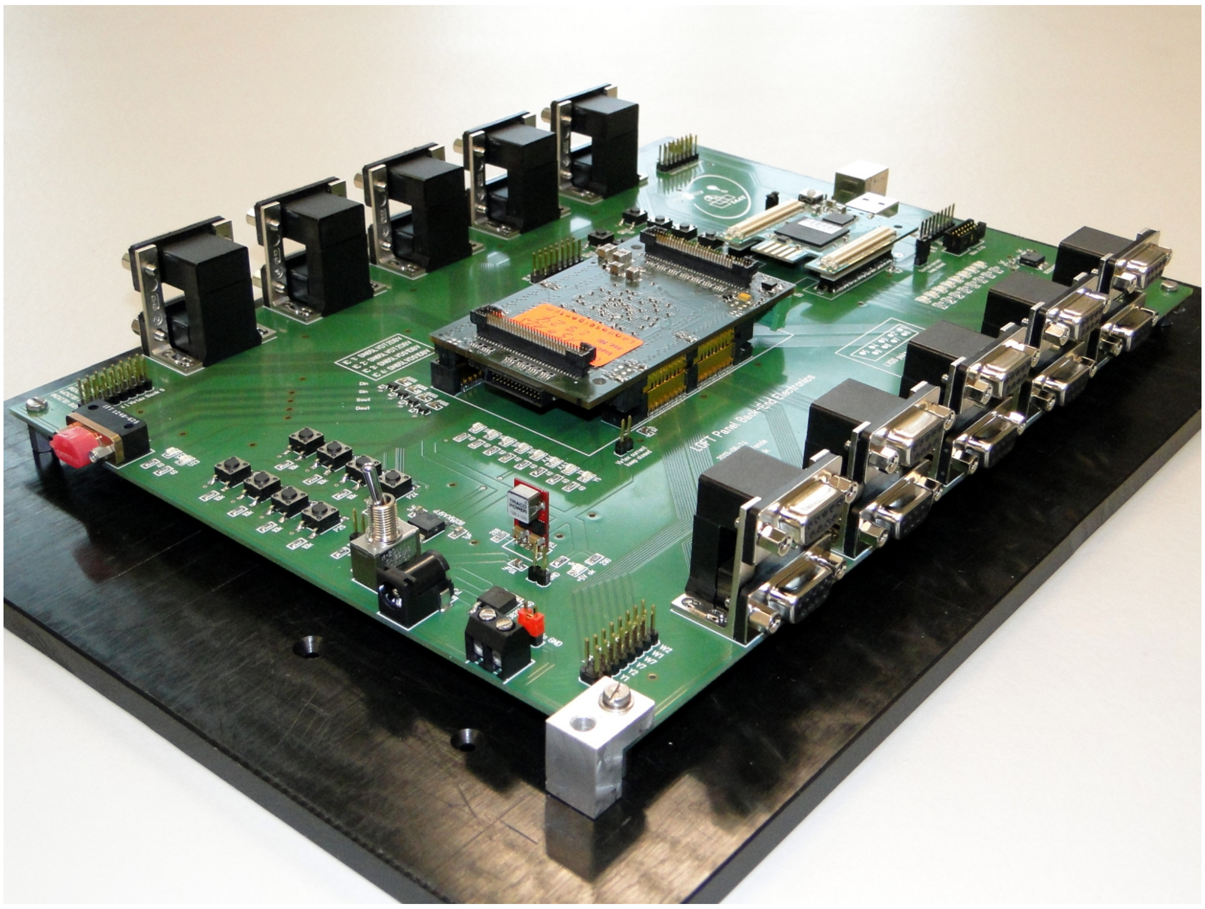


Figure 6.20
PBEE prototype board. The PCB features 20 connectors for MBEE boards and two on-board MBEE simulators via an additional Spartan 3 FPGA hardwired to the Virtex 4 containing the PBEE. Both FPGAs are connected to a set of eight LEDs and buttons for testing and debugging. The board also includes low-voltage SpaceWire LVDS drivers and a SpaceWire connector used for the interface to the DHU.

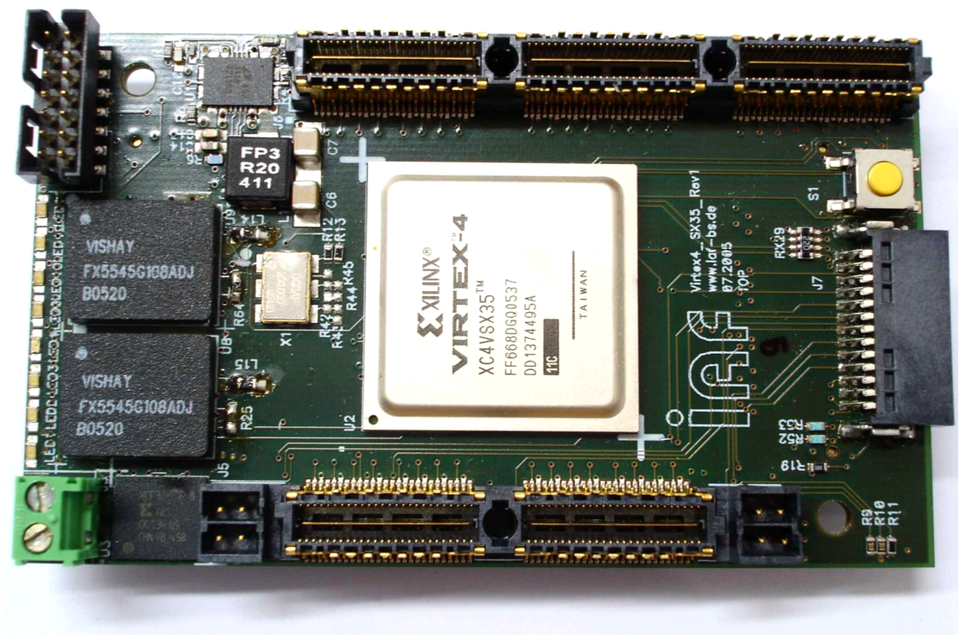


Figure 6.21
The IAF Virtex 4 SX35 add-on-board used to integrate the Virtex 4 FPGA into the PBEE prototype. It features an external 5 V power supply but also allows the board to be powered through the two QFSS Samtec sockets. It further includes an on-board oscillator that can be configured to provide 100 MHz (used) or 200 MHz. The voltage on the FPGA's I/O banks can be set through soldering connections to 2.5 V (used) or 3.3 V. Source: IAF GmbH (2006)

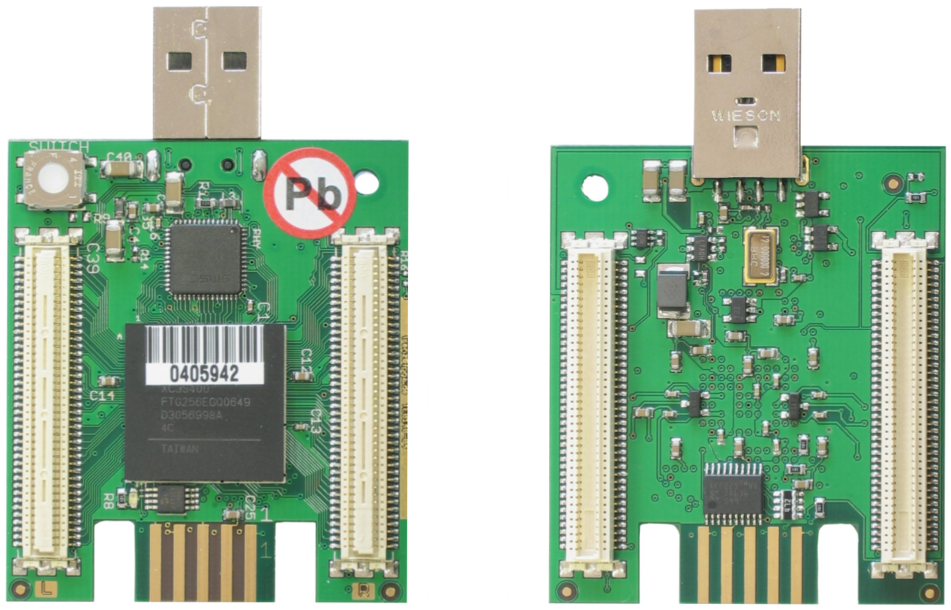


Figure 6.22
The Trenz Spartan 3 FPGA Micromodule is powered by a 5 V power source fed through the B2B connectors. While a 40 MHz oscillator is included on the module, we use a clock signal generated by the PBEE to ensure synchronous communication. The USB port can be used to send pre-generated events from a PC to the MBEE simulator.
The micromodule comes with one LED and one bush button and the PBEE prototype PCB features eight additional LEDs and buttons around the B2B connectors useful for debugging and testing.
A total of 16 signals and two clocks are routed on-board from the Virtex 4 to the Spartan 3 module, enabling the implementation and operation of two independent MBEEs in the micromodule. Source: Trenz Electronic GmbH (2008)

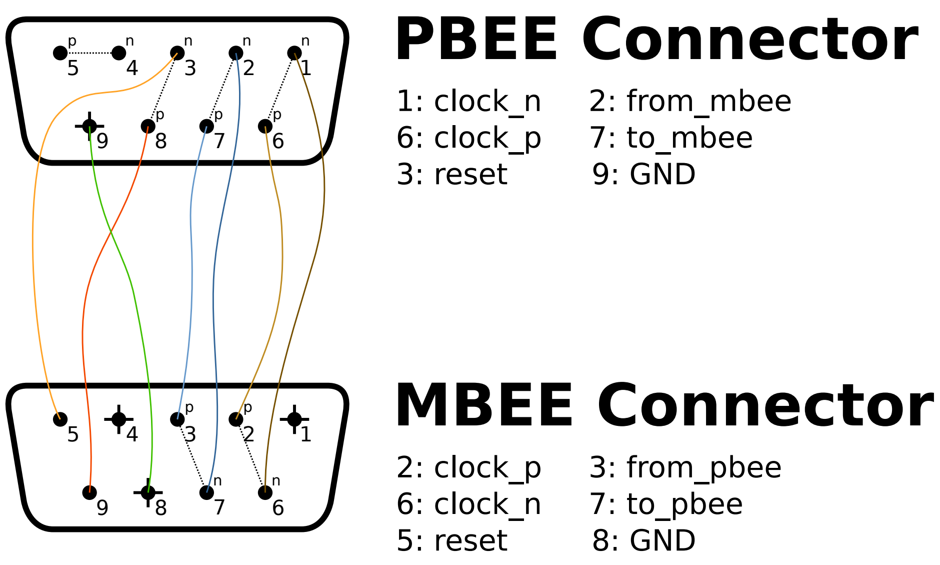


Figure 6.23
The connector layout used for the MBEE - PBEE prototype interface. For the prototypes we used standard 9 pin D-Sub connectors. The flight version of the PBEE board will use space qualified Micro-D connectors.
Since the PBEE prototype PCB is a two-layer design some of the on-board connectors have a different pinout and therefore require different pin assignments than the one shown. Detailed information on the pinout of each connector can be found in the PBEE design universal constraint file.

6.8.3 Test Setup and Verification

Simulation

During the development of the MBEE and the PBEE prototype VHDL designs, we tested all components of the two modular designs in the Xilinx ISE Simulator (ISim). For each component an individual test-bench is created and used in ISim to test the interfaces and the internal processes of the design. This is a standard procedure during VHDL development and is very useful to re-validate a component every time a change is made during the entire development process.

The MBEE components that compose the event reconstruction pipeline have been simulated and validated by Pascal Uter (Uter 2013). The components of the MBEE-PBEE interface and the components of the PBEE data handling procedures have been simulated and validated as part of this thesis.

Hardware Verification

To demonstrate and verify the performance and the correctness of the interface developed for MBEE-PBEE communication, and to validate the data handling procedures implemented in the PBEE we arranged a test setup with the PBEE prototype, the MBEE prototype, and the two MBEE simulators included in the PBEE prototype board.

In this setup pre-arranged events were uploaded into the three MBEEs and then sent according to their timestamps to the PBEE. The events were received and handled according to the specifications given in Section 6.7.2. The resulting event stream was sent from the PBEE to a PC and compared to the expected results that had been created with an event handling simulator on the PC.

The clock frequency used for the interface clock was 5 MHz, the MBEE clock frequency was 40 MHz and the PBEE was running at a clock frequency of 100 MHz. The MBEE and interface clocks were generated in the PBEE and distributed to the MBEEs.

The tests show that the custom designed interface handles the transmission of an event from the MBEE to the PBEE including the response from PBEE and the overhead transmission processing time in 24 µs. The required performance is 25 µs resulting in a net data rate of 40 000 events/s or 1.6 Mbit/s (with 40 bit/event). The communication is shown in Figure 6.24 during an interface simulation performed in ISim.

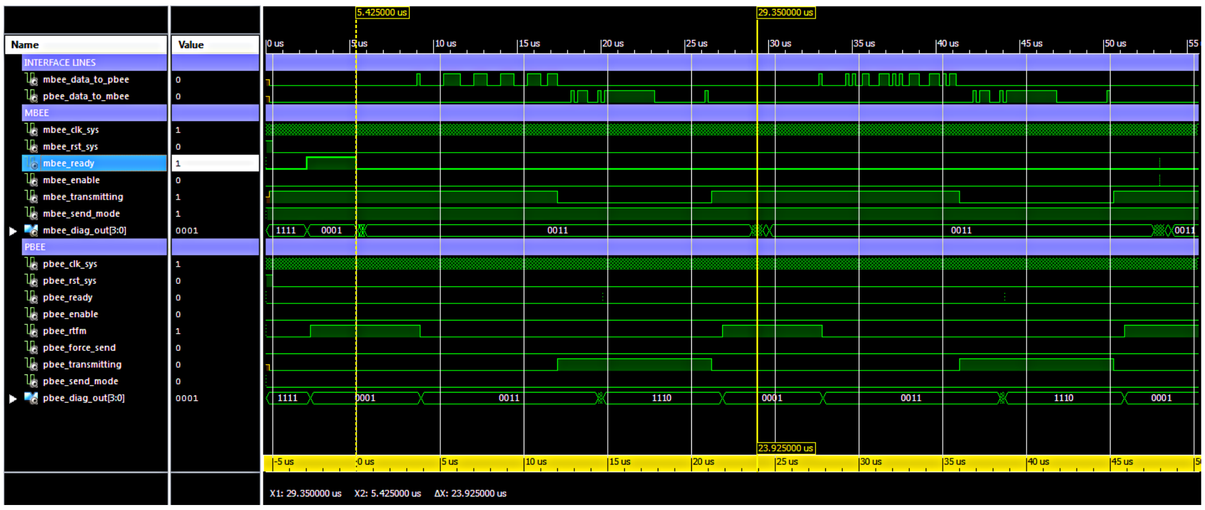


Figure 6.24
Simulation of the MBEE-PBEE custom designed interface when sending events to the PBEE. In the simulator the bi-directional data line is split into two separate lines, the two topmost lines in the simulator. The time from MBEE ready-to-ready is measured as 23.925 µs. Required is a time of 25 µs resulting in a net event data rate of 40 000 events/s or 1.6 Mbit/s.

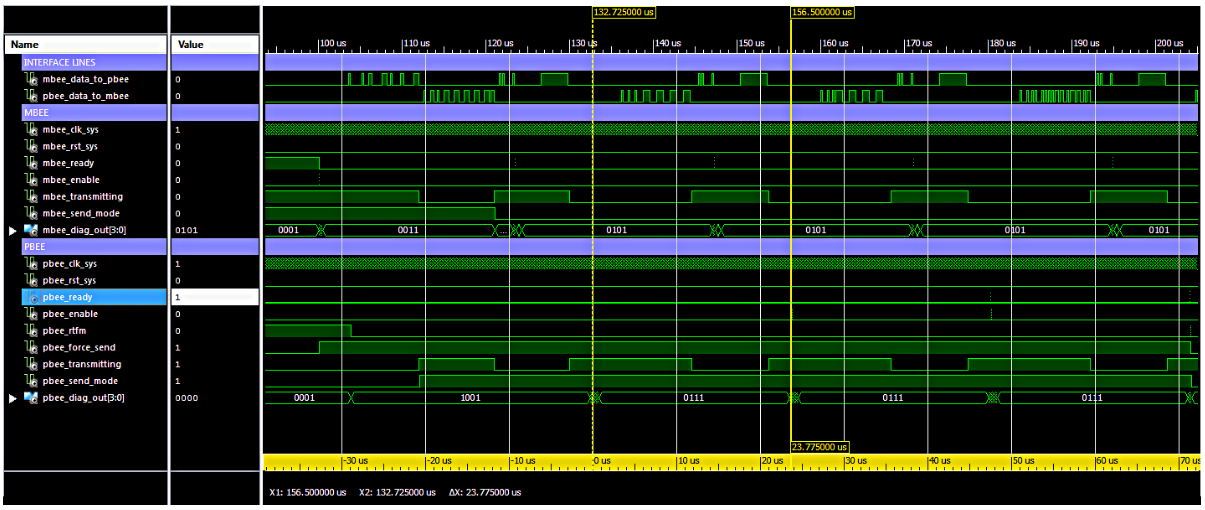


Figure 6.25
Simulation of the MBEE-PBEE custom designed interface when sending commands to the MBEE. After an event has been sent to the PBEE (100-110 µs) the PBEE responds with the command to reverse the logical direction (110-120 µs) which is acknowledged by the MBEE (120-130 µs). After that data can be sent from the PBEE to the MBEE.

Another test was the transmission of commands from the PBEE to the MBEE. As described in Section 6.7.1 the logical direction of the transfer has to be reversed, then commands and data can be transmitted to the MBEE, and finally the direction is reset to normal. The initiation of this reversed communication and the transmission of the first data packets is shown in Figure 6.25 during an interface simulation performed in ISim. This process was demonstrated successfully by sending configuration data to the MBEE and transmitting commands to start and end the event transmission for the bandwidth tests. Although the MBEE could receive commands, the MBEE prototype design did not implement different modes of operation at the time these tests were performed. MBEE modes have since been implemented in the context of the diploma thesis of Michael Gschwender (Gschwender 2014).

Chapter 7

Summary & Outlook

IXO

The first part of the work done in the context of this thesis was the development of the Data Handling Unit (DPU) for the High Time Resolution Spectrometer (HTRS). The HTRS is one of six instruments aboard the International X-ray Observatory (IXO). The prototype is based on the LEON3, a VHDL design of a microprocessor implemented in a Virtex 4 FPGA. The work included definition, development and validation of an operational prototype implemented on a Pender GR-X3CS development board.

Simulations of the DPU operations have been performed. The results obtained have lead to the conclusion that the requirements on the HTRS telemetry rate can be met using the bzip2 compression algorithm. The performances of several different compression algorithms (such as gzip, zlib, lzma, paq9a, and bzip2) have been analyzed and it was found that bzip2 is well suited for the task. The evaluated parameters were compression strength and speed, data integrity verification, and readiness for parallelism.

The LEON3 VHDL design was implemented in an FPGA and operated with the prototype DPU software. The software is based on RTEMS and its development was also part of the work. An exemplary run of the compression on the LEON3 development board was completed with the expected compression ratios.

An important improvement to the knowledge of the final DPU performance will come from the inclusion of an external mass memory device into the DPU. The DPU functionality and performance can be extended with additional custom IP cores integrated into the LEON3 design. Compression strength can be further increased by using a multi-core LEON3 VHDL design. An increased compression strength would enable the creation of spectra with improved energy resolution for brighter sources.

LOFT

The second part of the work was the development of the Panel-Back-End-Electronics (PBEE) for the Large Area Detector (LAD) of the Large Observatory for X-ray Timing (LOFT). A prototype PCB has been built and the PBEE VHDL design was implemented in a Virtex 4 FPGA. This design contains the interface to 21 MBEEs and the event data handling procedures required to create a serialized event stream that can be sent to the Data Handling Unit (DHU). In addition a custom interface with very low hardware complexity was developed for the communication between the MBEEs and the PBEE.

The correctness of the data handling procedures was verified with simulations and a hardware test setup consisting of the PBEE prototype, the MBEE prototype and two MBEE simulators integrated on the PBEE prototype PCB. The expected performance of the interface was demonstrated with simulations and the same test setup. The results have been presented in Chapter 6.8.3 - Test Setup and Verification.

The MBEE and PBEE data handling procedures as well as the custom designed interface were found to meet the specified performance and timing requirements. However, a full LAD data processing chain from the detection of a photon to the interface to the spacecraft can only be demonstrated with additional components, such as the Data Handling Unit (DHU) and FEE (Front-End-Electronics) simulators.

An MBBE prototype and two simulators have already been used to demonstrate the capability of the PBEE to handle multiple MBEEs in parallel and the PBEE design provides interfaces for a total number of 21 MBEEs, but a full scale communication test should be performed to validate timing and performance for the flight configuration. To demonstrate the parallel operation of multiple PBEEs, another PBEE prototype has to be built and the DHU or at least an interposed Multi Interface Control Unit (MICU) has to be developed.

Part IV

Appendix

LEON3 Configuration and Implementation Guide

A New Project in Xilinx ISE

This guide shows you how to create a new VHDL-project in Xilinx ISE. This includes setting up a project-folder-structure, writing the source code for a minimal example and synthesizing the design.

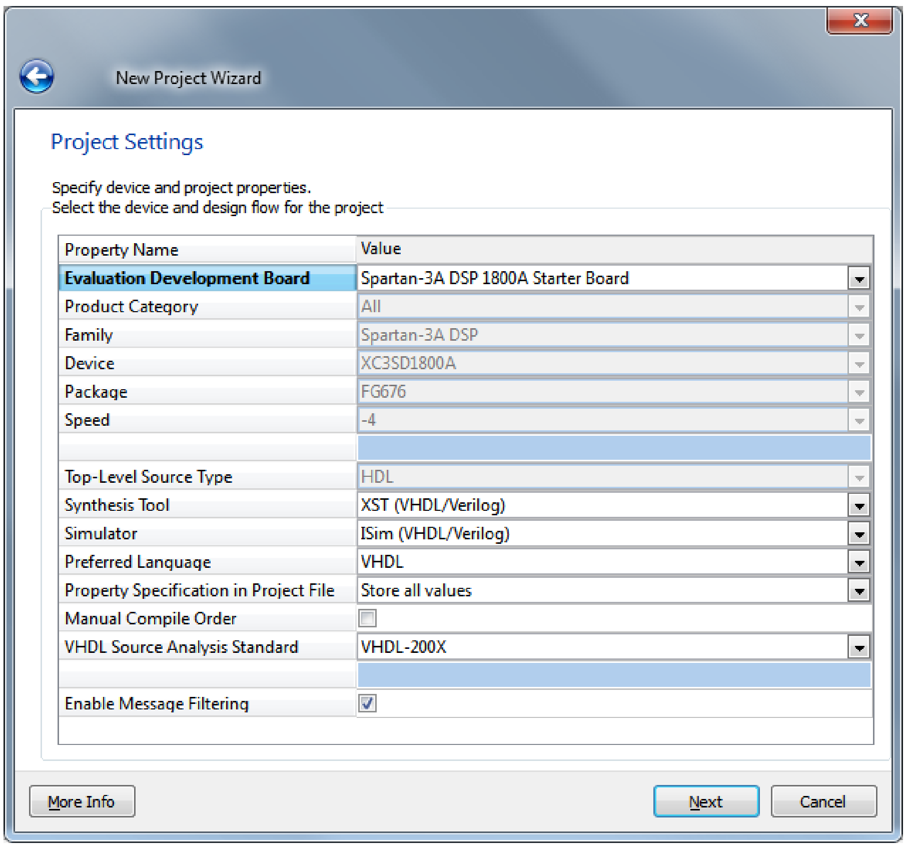
1. First, create a folder for all your VHDL-projects. This guide assumes you have such a folder and that you work inside it.
2. Create a folder `PushLED`[1].
3. Create the files `PushLED.vhd` and `PushLED.ucf` in this folder.
4. Now start the program **ISE Design Suite**.
5. Click **New Project** and enter "PushLED_ISE" as "Name". Then specify your folder `PushLED` as "Location". You will end up with two folders: One is your project folder which will contain all important files such as the source code and the user constraints file. The other folder (inside your project folder) has the suffix `_ISE` and will contain a large number of files that you don't really need to see.
6. For "Top-level source type" select "HDL" and click **Next**.
7. Configure your Project Settings as shown in Figure A.1 and click **Next**.
8. In the next step you see the "Project Summary". After you clicked **Finish** you see your working area. This should look as shown in Figure A.2. If your "ISE Design Suite Info Center" is open, please close it.
9. To your left in the "Hierarchy" panel you see the project tree starting with your device (xc3sd1800a-4fg676). Right click on the device and choose **Add Source**. Now browse to the folder `PushLED` (not the `_ISE` one) and add your VHDL dummy file `PushLED.vhd`. In the "Addings Source Files..." dialog click **OK**.
10. Do the same to add you User Constraints File `PushLED.ucf`.
11. Doubleclick on **PushLED.vhd** in the "Hierarchie" panel and add some example code. You can use Code Example A.1 on page 134.

[1] Let's assume a "PushLED" is what we want to design.

12. Now add the example code from Code Example A.2 on page 134 to the ucf-file in a similar way.
13. To start synthesizing the design click on your top module **PushLED - Behavioral (PushLED.vhd)** in the "Hierarchie" panel and then doubleclick **Synthesize - XST** in the process panel below. The synthesis should run for some time until the "Console" at the bottom displays "Process 'Synthesize - XST' completed successfully".
14. Now doubleclick **Implement Design** and wait for the process to finish.
15. Doubleclick **Generate Programming File** and wait once more.[2]
16. In your `PushLED_ISE` folder you will now find a file called `pushled.bit`[3] (for no reason Xilinx ISE uses random capitalization on your filenames). That's the file you can now load into your FPGA.
    You're done!

[2] Later you can do all of these three steps at once by clicking the green arrow to the upper left of the process panel.

[3] When you're done developing you might want to save a copy of this file in your `PushLED` folder to keep it. You can than safely delete the folder `PushLED_ISE` if you no longer need it.

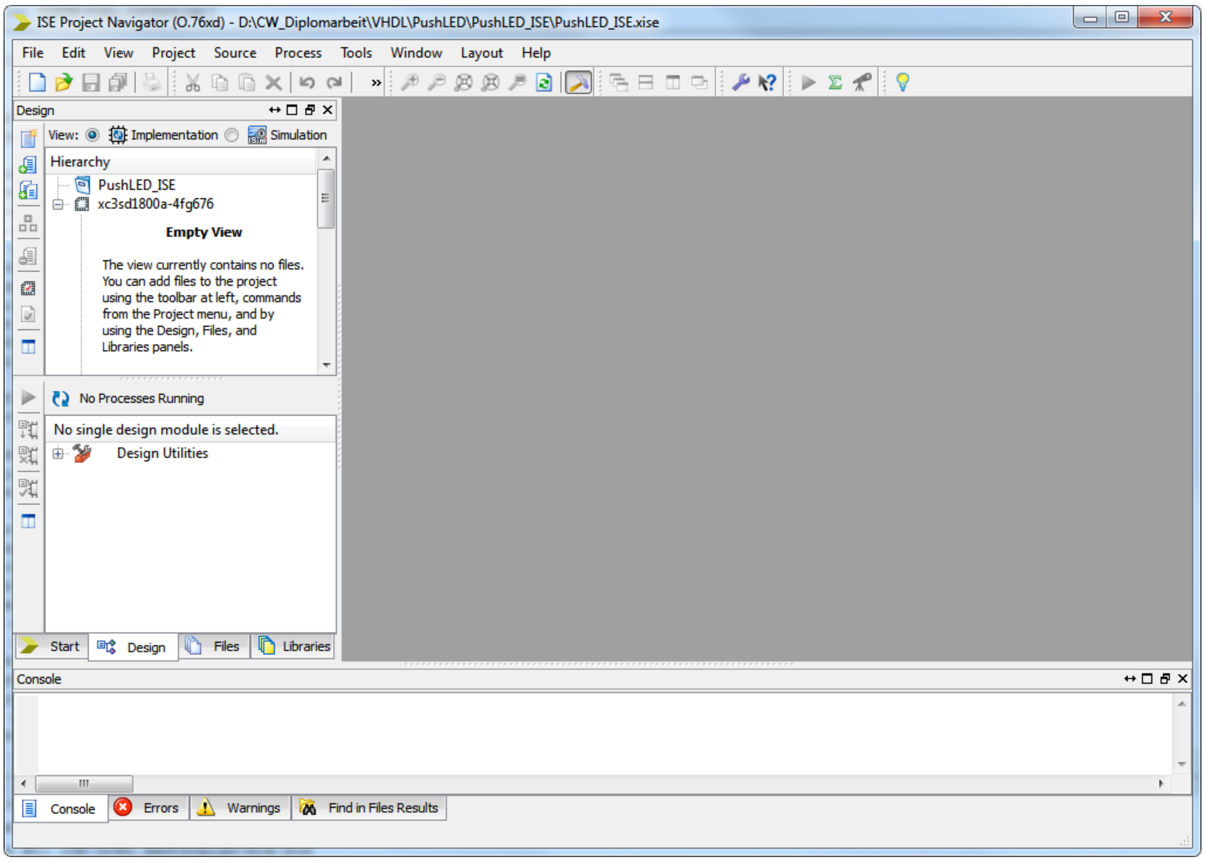


Project Settings

Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	Spartan-3A DSP 1800A Starter Board
Product Category	All
Family	Spartan-3A DSP
Device	XC3SD1800A
Package	FG676
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-200X
Enable Message Filtering	✓

Figure A.1
The “Project Settings” dialog as displayed during the creation of a new project. This setup is used for projects targeting the Xtreme DSP 1800A Starter Platform.


ISE Project Navigator (O.76xd) - D:\CW_Diplomarbeit\VHDL\PushLED\PushLED_ISE\PushLED_ISE.xise

File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
PushLED_ISE
xc3sd1800a-4fg676

Empty View

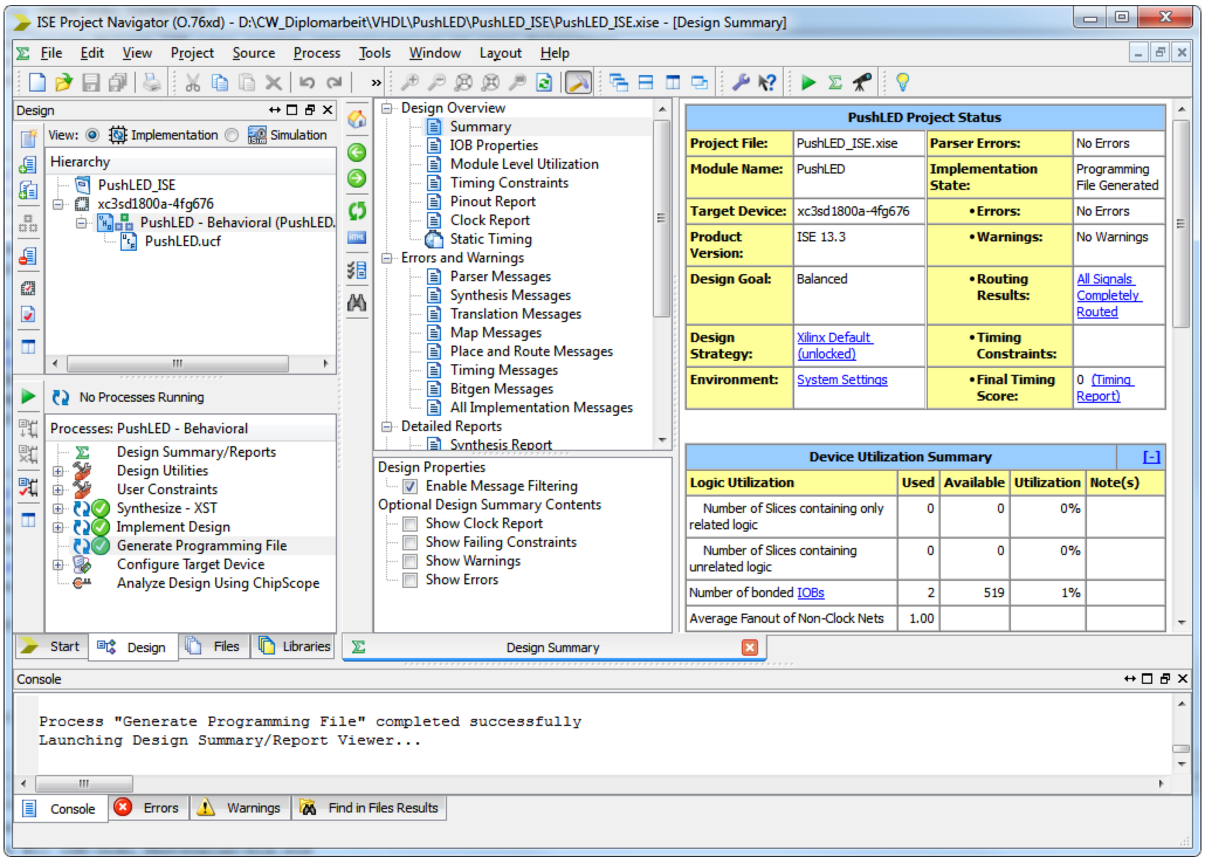
The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.

No Processes Running

No single design module is selected.

Design Utilities
Start Design Files Libraries
Console
Console Errors Warnings Find in Files Results

Figure A.2
The empty working area as seen after creating a new project.



Design Properties
[✓] Enable Message Filtering
Optional Design Summary Contents
[ ] Show Clock Report
[ ] Show Failing Constraints
[ ] Show Warnings
[ ] Show Errors

PushLED Project Status			
Project File:	PushLED_ISE.xise	**Parser Errors:**	No Errors
Module Name:	PushLED	**Implementation State:**	Programming File Generated
Target Device:	xc3sd1800a-4fg676	**•Errors:**	No Errors
Product Version:	ISE 13.3	**•Warnings:**	No Warnings
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slices containing only related logic	0	0	0%	
Number of Slices containing unrelated logic	0	0	0%	
Number of bonded IOBs	2	519	1%	
Average Fanout of Non-Clock Nets	1.00			

```
Process "Generate Programming File" completed successfully
Launching Design Summary/Report Viewer...
```

Figure A.3
The working area after running a successful implementation.
The “Hierarchy” panel shows the target device (xc3sd1800a-4fg676) and the associated files. There is one vhdl-file that contains the design for an entity called “PushLED”. Since this appears to be the top-level module there is another ucf-file below it. This User Constraints File associates the top-level signals (defined in the vhdl-file and used in the design) with pins that are specific to the target device.

PushLED.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

--------------------------------------------------------------------------------
-- E N T I T Y
--------------------------------------------------------------------------------
entity PushLED is
    port(
        Button : in  std_logic;
        LED    : out std_logic
    );
end entity;

--------------------------------------------------------------------------------
-- A R C H I T E C T U R E
--------------------------------------------------------------------------------
architecture Behavioral of PushLED is
begin
    LED <= Button;
end architecture;
```

Code Example A.1
VHDL code for the PushLED project used in the guide A New Project in Xilinx ISE. The PushLED entity uses one input and one output of type std_logic. In the behavioral description of the entity the input is directly mapped to the output. Note that only the ucf-file maps the input to a button and the output to an LED.

PushLED.ucf

```
#### Buttons
Net Button LOC = "J17" | IOSTANDARD = LVTTL;

#### LEDs
Net LED    LOC = "D25" | IOSTANDARD = LVTTL;
```

Code Example A.2
Each top-level input and output in the design has to be mapped onto some hardware. The User Constraints File specifies FPGA pins (e.g. LOC = “D25”) and since the “Place & Route” process knows what FPGA to use (specified in the Project Settings) it can create the correct netlist.

Using iMPACT to Configure a Target Device

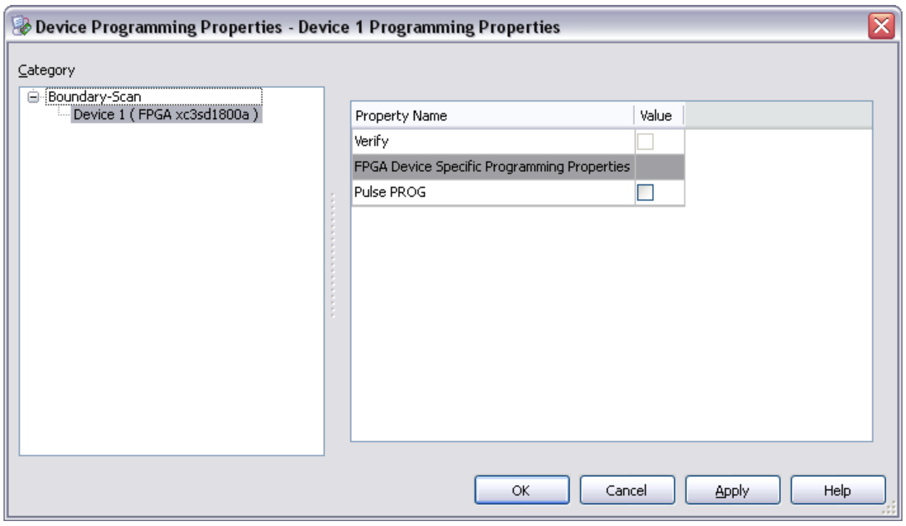
Follow these instructions to load your bit-file into an FPGA using Xilinx iMPACT.

1. You can launch iMPACT either from the windows start menu or from within the ISE Design Suite by clicking on **Configure Target Device**[1] in the process panel.
2. Doubleclick on **Boundary Scan** in the top left "iMPACT Flows" panel.
3. Now make sure your board is connected and powered up, then use the green chain-like button or **File** > **Initialize Chain**.
4. In the "Auto Assign Configuration Files Query Dialog" answer **No**[2].
5. Click **OK** if your "Device Programming Properties" look as shown in Figure A.4.
6. Figure A.5 shows iMPACT after successfully identifying the FPGA and the on-board SPI/BPI Flash.
7. Now take a break and look what has happened. iMPACT autodetected your JTAG cable (USB or parallel) and used the JTAG interface to connect to your FPGA. iMPACT is also able to use the JTAG interface to connect to the on-board flash memory. Everything it finds is displayed in the main window. There you should see your FPGA and in the case of the Xtreme DSP 1800A Starter Platform you should also see a blue box labeled "SPI/BPI" above the FPGA. This indicates that flash memory is available through an indirect programming method using the FPGA with a proprietary IP core. This is explained in detail in the guide Writing a PROM File to the BPI Flash.
8. Now rightclick the FPGA and choose **Assign New Configuration File** and then select the previously generated bit-file `PushLED.bit`.
9. When presented with the choice to "Attach SPI or BPI PROM" answer **No**.
10. Rightclick the FPGA again and select **Program**.

[1] Usually you get the warning that "No iMPACT project file exists". You can change this later, so simply press **OK**.

[2] Do this for now. When you are familiar with the process you can of course use this shortcut.

11. Figure A.6 shows iMPACT after successfully programming the FPGA. Your design is now active and the FPGA should operate accordingly.
    You're done!



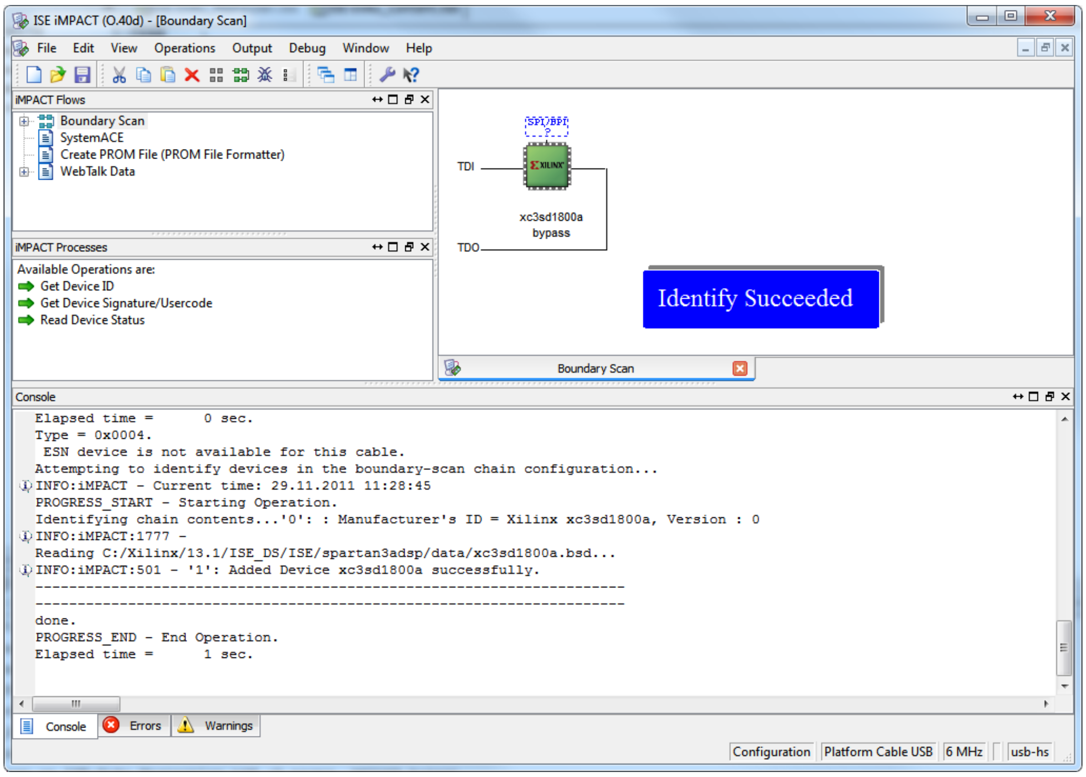
Device Programming Properties - Device 1 Programming Properties

Category

Boundary-Scan
Device 1 (FPGA xc3sd1800a)

Property Name	Value
Verify	☐
FPGA Device Specific Programming Properties	
Pulse PROG	☐

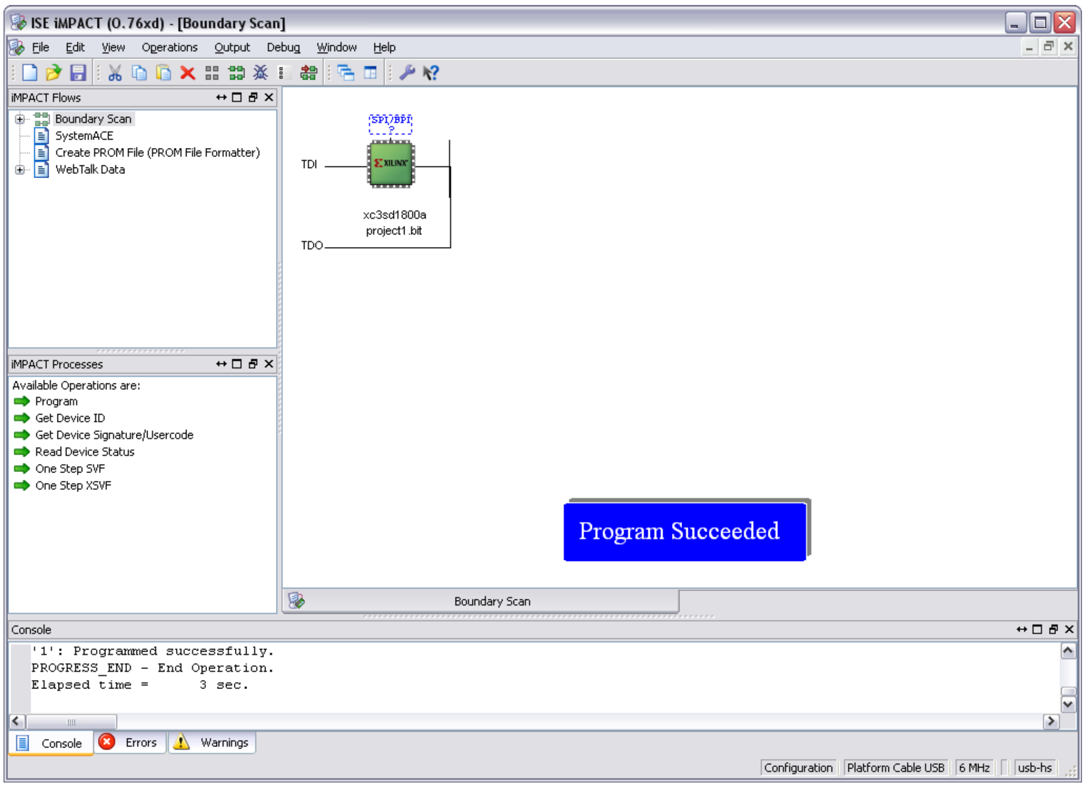
Figure A.4
This dialog is shown in iMPACT every time a boundary chain scan is performed. The programming properties of the target device can be set, e.g. verifying the correct transfer of the data after the programming process.



Available Operations are:
Get Device ID
Get Device Signature/Usercode
Read Device Status

```
Elapsed time =      0 sec.
Type = 0x0004.
 ESN device is not available for this cable.
Attempting to identify devices in the boundary-scan chain configuration...
INFO:iMPACT - Current time: 29.11.2011 11:28:45
PROGRESS_START - Starting Operation.
Identifying chain contents...'0': : Manufacturer's ID = Xilinx xc3sd1800a, Version : 0
INFO:iMPACT:1777 -
Reading C:/Xilinx/13.1/ISE_DS/ISE/spartan3adsp/data/xc3sd1800a.bsd...
INFO:iMPACT:501 - '1': Added Device xc3sd1800a successfully.
----------------------------------------------------------------------
----------------------------------------------------------------------
done.
PROGRESS_END - End Operation.
Elapsed time =      1 sec.
```

Figure A.5
iMPACT displaying a successfully intialized chain.



```
'1': Programmed successfully.
PROGRESS_END - End Operation.
Elapsed time =      3 sec.
```

Figure A.6
iMPACT after successfully programming the FPGA with the bit-file `PushLED.bit`. The console output gives detailed information about the process and shows that the programming took just 3 seconds.

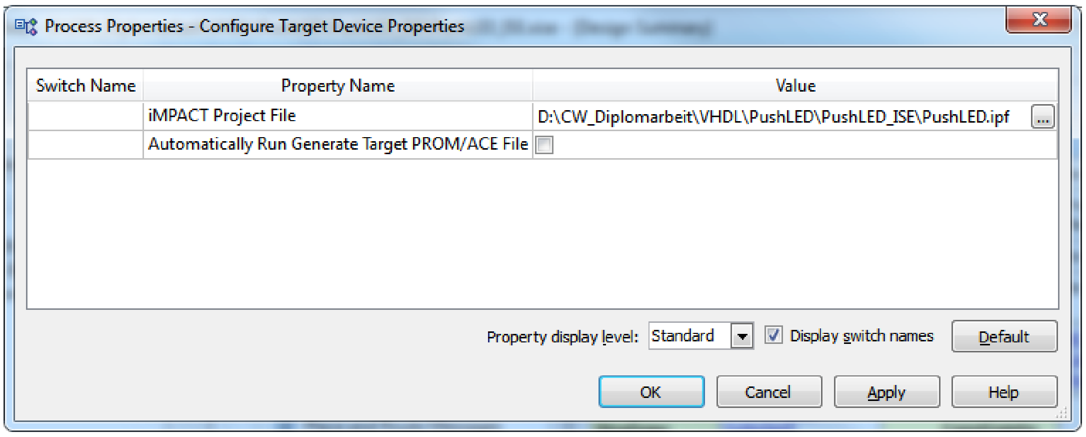
Using ISE Design Suite to Configure a Target Device

This section describes how to configure iMPACT and create a project file so that programming the target device with a bit-file can be done from within Xilinx ISE Design Suite without actually launching iMPACT.

Important: Please start by reading Using iMPACT to Configure a Target Device and perform all the steps described there. Then return to this section and carry out the instructions as follows:

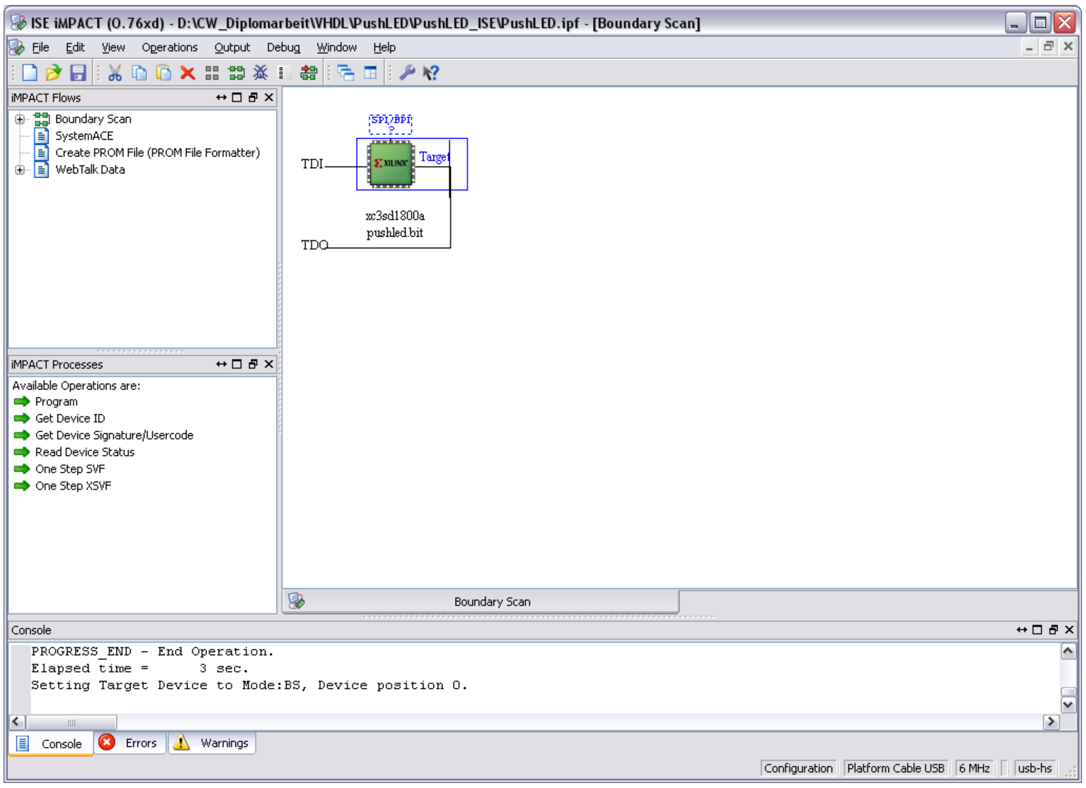
1. Rightclick the FPGA again and select **Set Target Device**. Now your window should look as shown in Figure A.8.
2. Click on the blue disk project symbol or click **File** > **Save Project** to create a project file and name it “PushLED.ipf”[1].
3. Close iMPACT and return to Xilinx ISE Design Suite.
4. Rightclick on **Configure Target Device** in the process window and select **Process Properties**.
5. Specify the newly created iMPACT project file `PushLED.ipf` (including its full path). This is shown in Figure A.7.
6. Click **OK** to close the dialog.
7. By doubleclicking on **Configure Target Device** the program iMPACT is no longer launched but the design is automatically loaded into the FPGA (see Figure A.9).
8. Please note that you can no longer launch iMPACT from within ISE Design Suite but you can use a shortcut from the windows start menu.
   You’re done!

[1] Assuming you still work with the example project from the guide A New Project in Xilinx ISE.



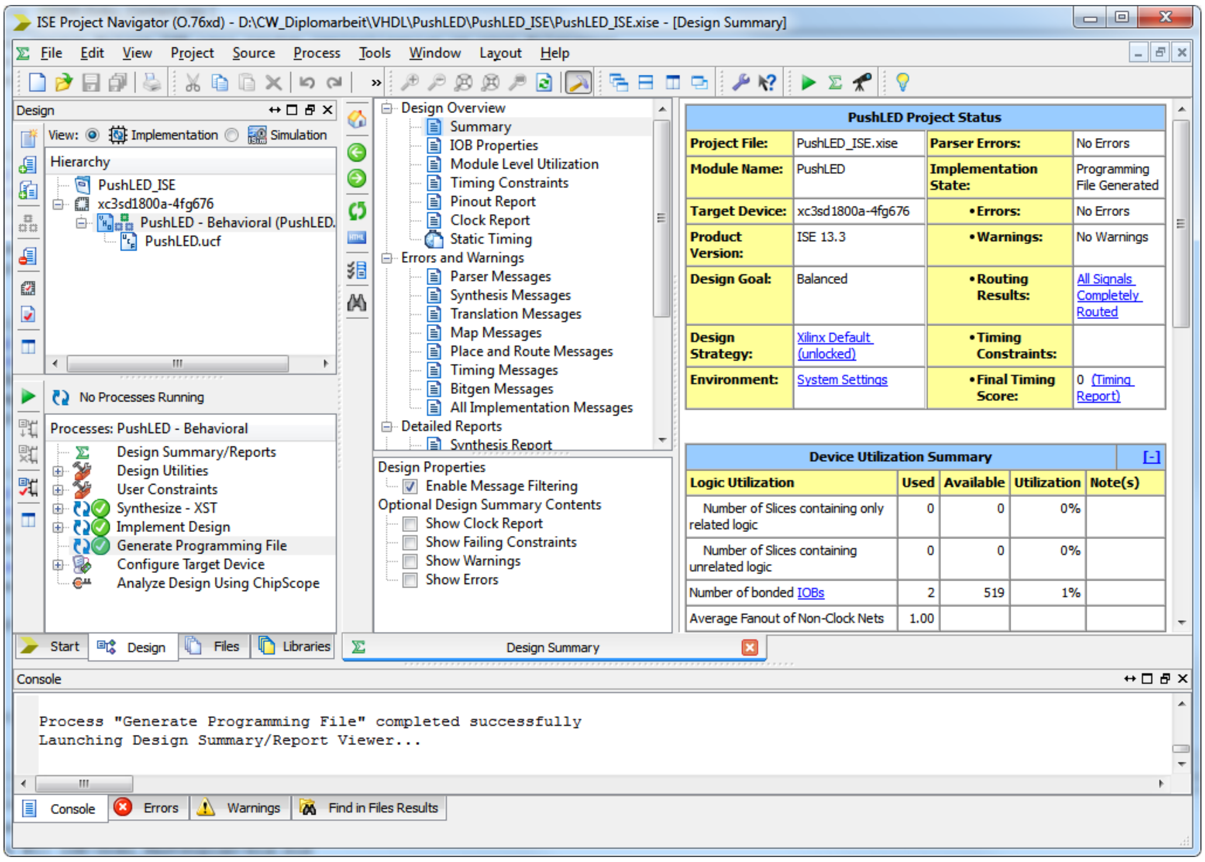
Switch Name	Property Name	Value
	iMPACT Project File	D:\CW_Diplomarbeit\VHDL\PushLED\PushLED_ISE\PushLED.ipf
	Automatically Run Generate Target PROM/ACE File	☐

Figure A.7
Specification of the newly created iMPACT project file.



```
PROGRESS_END - End Operation.
Elapsed time =      3 sec.
Setting Target Device to Mode:BS, Device position 0.
```

Figure A.8
iMPACT after the creation and configuration of a project.
A bitfile is assigned to the FPGA and the target device is specified.



PushLED Project Status

Project File:	PushLED_ISE.xise	**Parser Errors:**	No Errors
Module Name:	PushLED	**Implementation State:**	Programming File Generated
Target Device:	xc3sd1800a-4fg676	**•Errors:**	No Errors
Product Version:	ISE 13.3	**•Warnings:**	No Warnings
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary [-]

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slices containing only related logic	0	0	0%	
Number of Slices containing unrelated logic	0	0	0%	
Number of bonded IOBs	2	519	1%	
Average Fanout of Non-Clock Nets	1.00			

```
Process "Generate Programming File" completed successfully
Launching Design Summary/Report Viewer...
```

Figure A.9
The design has now been fully synthesized, placed and routed, and the bit-file successfully loaded into the FPGA.

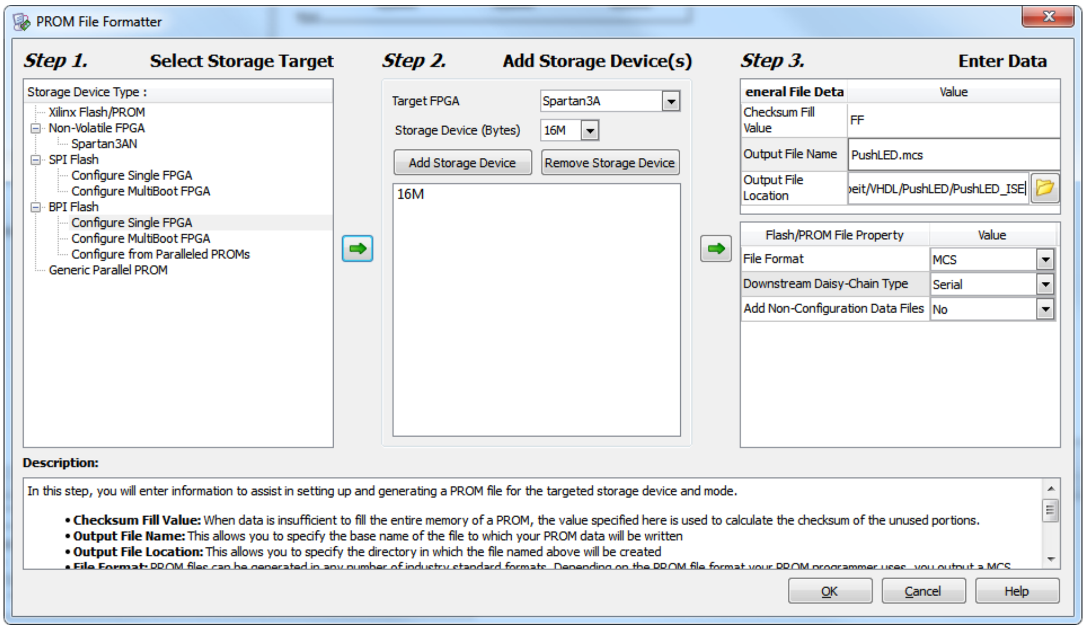
Creating a PROM File for the BPI Flash

Follow these instructions to create an mcs-file to be loaded into the BPI Flash.

1. Start iMPACT from the windows start menu. Starting iMPACT from within the ISE Design Suite will only work if it has not (yet) been configured as described in the guide Using ISE Design Suite to Configure a Target Device.
2. Do not create a project file and do not open a project file.
3. Doubleclick on **Create PROM File (PROM File Formatter)** in the "iMPACT Flows" panel.
4. Select **BPI Flash** > **Configure Single FPGA** and click the green arrow.
5. For "Target FPGA" select **Spartan 3A**.
6. For "Storage Device (bits)" select **16M**.
7. Click **Add Storage Device**.
8. Click the green arrow to proceed.
9. Now enter "PushLED.mcs"[1] as "Output File Name".
10. Make sure the "Output File Location" points to your `PushLED_ISE` folder.
11. Configure the "Flash/PROM File Properties" as shown in Figure A.10.
12. After clicking **OK** you have to specify which bit-file to use. Select the previously created bit-file.
13. If asked again, you do not want to add another device.
14. Click **OK** when presented with other questions and dialogs (which dialogs are shown might vary).
15. Now doubleclick **Generate File...** in the "iMPACT Processes" panel. The result should look as shown in Figure A.11.

[1] Assuming you still work with the example project from A New Project in Xilinx ISE.

16. The “Console” output will show that three files were written. `PushLED.mcs` is the PROM file that can now be loaded into the BPI Flash.
    You’re done!



Target FPGA	Spartan3A
Storage Device (Bytes)	16M

16M

eneral File Deta	Value
Checksum Fill Value	FF
Output File Name	PushLED.mcs
Output File Location	peit/VHDL/PushLED/PushLED_ISE

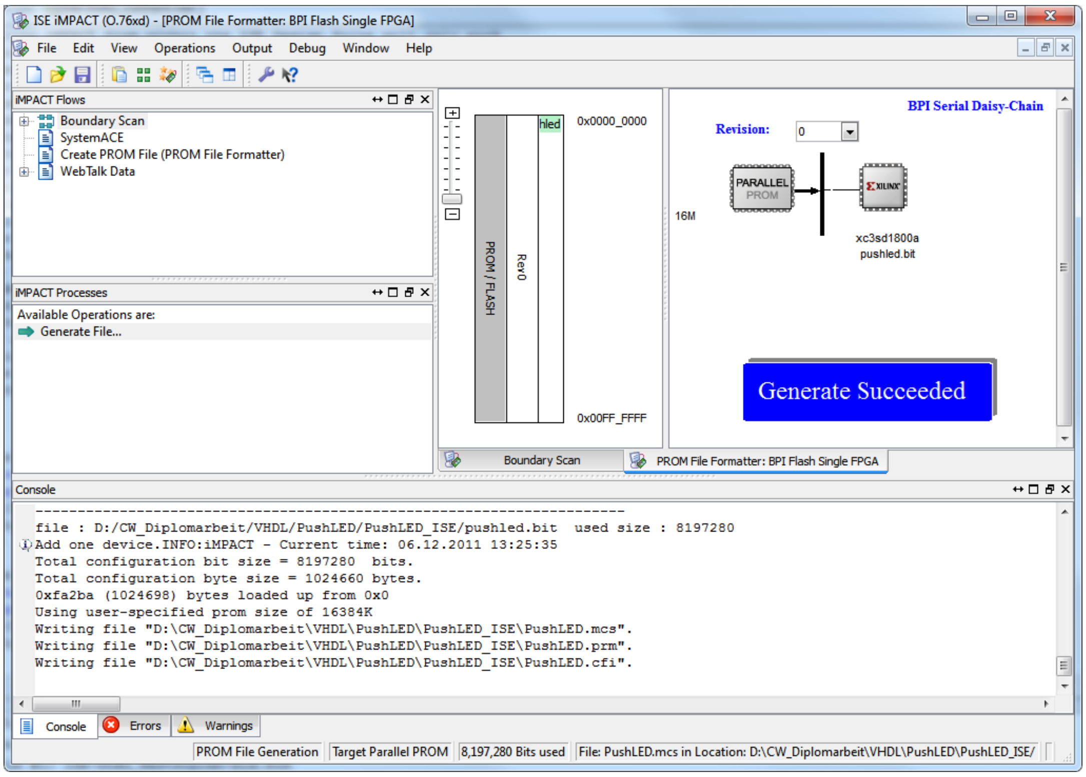
Flash/PROM File Property	Value
File Format	MCS
Downstream Daisy-Chain Type	Serial
Add Non-Configuration Data Files	No

Description:

In this step, you will enter information to assist in setting up and generating a PROM file for the targeted storage device and mode.

- **Checksum Fill Value:** When data is insufficient to fill the entire memory of a PROM, the value specified here is used to calculate the checksum of the unused portions.
- **Output File Name:** This allows you to specify the base name of the file to which your PROM data will be written
- **Output File Location:** This allows you to specify the directory in which the file named above will be created

Figure A.10
PROM File Formatter configured for BPI Flash.



```
-----------------------------------------------------------------------
file : D:/CW_Diplomarbeit/VHDL/PushLED/PushLED_ISE/pushled.bit  used size : 8197280
Add one device.INFO:iMPACT - Current time: 06.12.2011 13:25:35
Total configuration bit size = 8197280  bits.
Total configuration byte size = 1024660 bytes.
0xfa2ba (1024698) bytes loaded up from 0x0
Using user-specified prom size of 16384K
Writing file "D:\CW_Diplomarbeit\VHDL\PushLED\PushLED_ISE\PushLED.mcs".
Writing file "D:\CW_Diplomarbeit\VHDL\PushLED\PushLED_ISE\PushLED.prm".
Writing file "D:\CW_Diplomarbeit\VHDL\PushLED\PushLED_ISE\PushLED.cfi".
```

PROM File Generation | Target Parallel PROM | 8,197,280 Bits used | File: PushLED.mcs in Location: D:\CW_Diplomarbeit\VHDL\PushLED\PushLED_ISE/

Figure A.11
iMPACT after the successful generation of a BPI Flash PROM file.

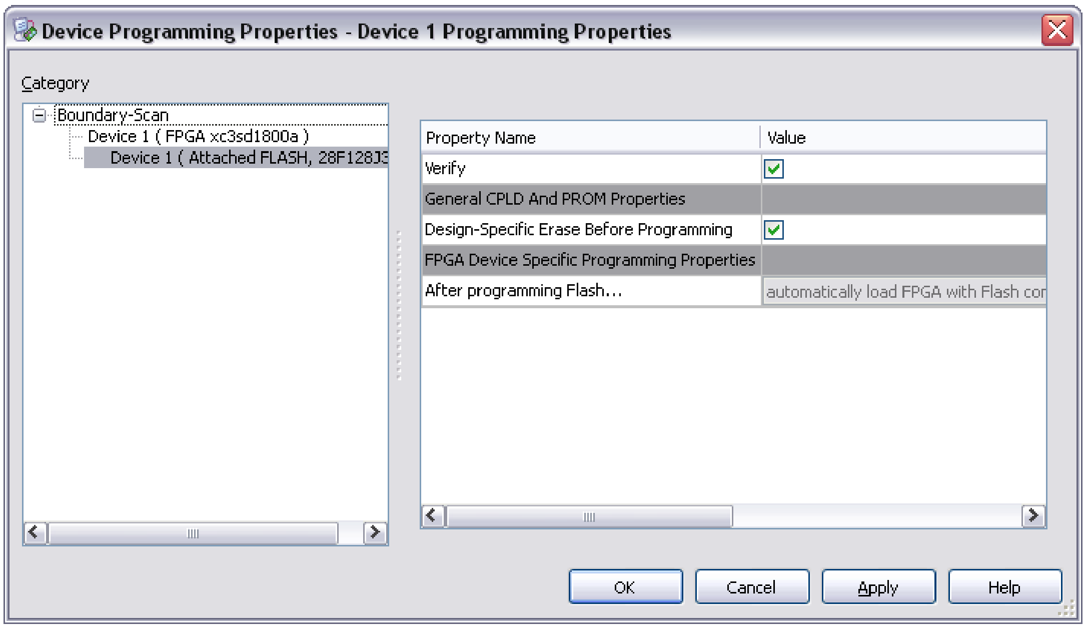
Writing a PROM File to the BPI Flash

Follow these instructions to write an mcs-file to the BPI Flash.

1. If there is no mcs-file, you can create one by following the instructions given in the guide Creating a PROM File for the BPI Flash on how to create such a file using iMPACT.
2. Make sure that all jumpers on the Spartan 3A DSP 1800A development board are set correctly:
   - JP7 (`PROG_B`) has to be open.
   - The configuration mode is selected via the jumpers on JP9. **M0 and M2 are closed**, M1 and the last jumper are open.
   - JP8 (flash write protection) has to be open.
3. Make sure that the board is connectedt via the USB JTAG-cable and powered up.
4. Launch iMPACT.
5. Doubleclick on **Boundary Scan** in the top left "iMPACT Flows" panel.
6. Now use the green chain-like button or **File** > **Initialize Chain**.
7. In the "Auto Assign Configuration Files Query Dialog" answer **No**[1].
8. Click **OK** if your "Device Programming Properties" look as shown in Figure A.4 on page 137.
9. Now rightclick the blue dashed area "SPI/BPI" shown above the FPGA and select "Add SPI/BPI Flash"
10. Select the previously created mcs-file `PushLED.mcs`.
11. For "BPI PROM" select **28F128J3D**. See the section "Intel J3 Parallel Flash" in Xilinx 2009 for more details.
12. Rightlick the now available "FLASH" part and select **Program**.
13. Set your Device Programming Properties as shown in Figure A.12 and click **OK**.

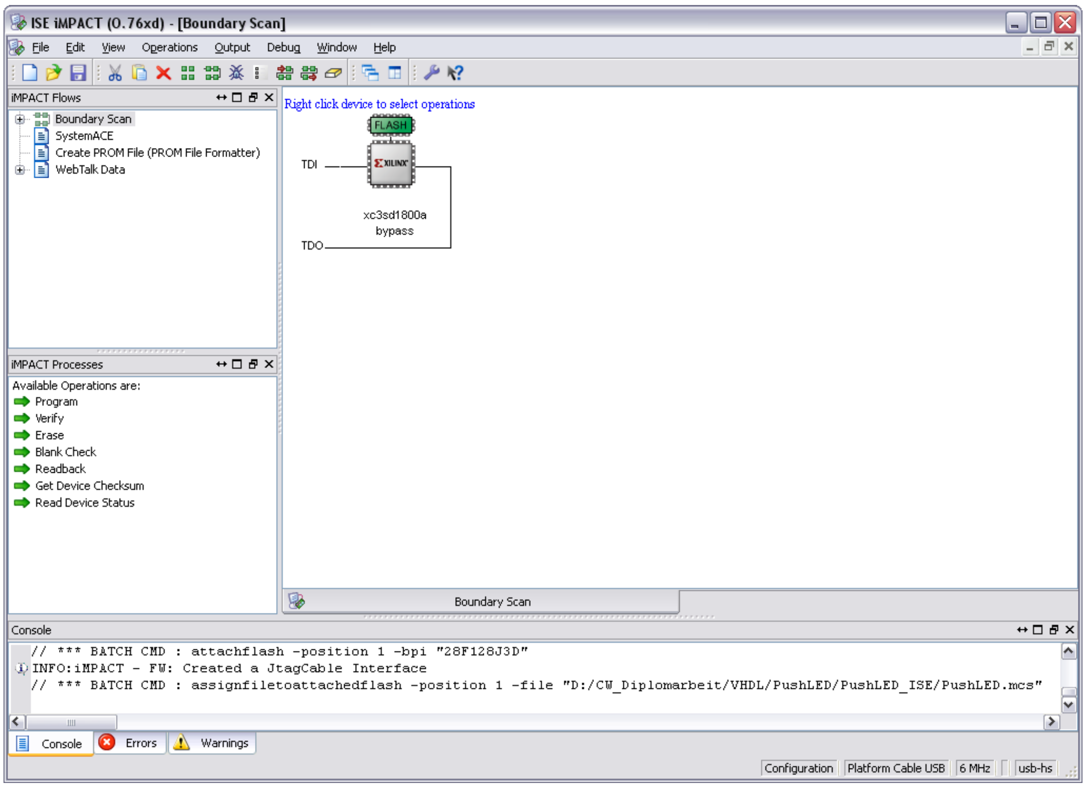
[1] Do this for now. When you are familiar with the process you can of course use this shortcut.

14. Figure A.13 shows iMPACT after successfully programming the BPI Flash. Please note that you have to maintain the jumper positions as described at the beginning of this guide to configure the FPGA from the BPI Flash automatically at power up. You're done!


Device Programming Properties - Device 1 Programming Properties
Category
Boundary-Scan
Device 1 (FPGA xc3sd1800a)
Device 1 (Attached FLASH, 28F128J3
OK
Cancel
Apply
Help

Property Name	Value
Verify	✓
General CPLD And PROM Properties	
Design-Specific Erase Before Programming	✓
FPGA Device Specific Programming Properties	
After programming Flash...	automatically load FPGA with Flash cor

Figure A.12
Programming properties for the BPI Flash.



```
// *** BATCH CMD : attachflash -position 1 -bpi "28F128J3D"
INFO:iMPACT - FW: Created a JtagCable Interface
// *** BATCH CMD : assignfiletoattachedflash -position 1 -file "D:/CW_Diplomarbeit/VHDL/PushLED/PushLED_ISE/PushLED.mcs"
```

Figure A.13
iMPACT after successfully programming the BPI Flash.

Implementing the LEON3 Design

Follow these instructions to configure the LEON3 model and to generate a bit-file. Note: The guides Creating a PROM File for the BPI Flash and Writing a PROM File to the BPI Flash describe how the bit-file is loaded into the FPGA.

1. Download the GRLIB IP Library from Aeroflex Gaisler[1] and extract it to e.g. C:\grlib-gpl-1.1.0-b4108.

2. Start your Cygwin shell.

3. Navigate to the folder
   grlib-gpl-1.1.0-b4108\designs\leon3-xilinx-xc3sd-1800.
   Note: Cygwin mounts your local partitions under /cygdrive/.

4. Depending on your type of Cygwin installation it might now be necessary to start the X-Server with the **startxwin** command. (It is necessary if the next step fails!)

5. Enter **make xconfig**.

6. Configure your LEON3 design using the GUI.

7. Finally click **Save and Exit**. Cygwin should print the line **config.vhd created** below the command you just entered.

8. Now enter **make ise-launch**. This will create a project file and automatically launch your installation of the ISE Design Suite. At this point the Cygwin prompt should look as shown in Figure A.14.

9. Since the automatically generated project file was most likely created for an older version of ISE, it will ask if you would like to migrate your project (see Figure A.15). Click **Backup and Migrate**.

10. In the “Hierarchy” panel (upper left) click on the top level module named **leon3mp - rtl (leon3mp.vhd)**. This should enable the actions in the “Processes” panel and the ISE should look as shown in Figure A.16.

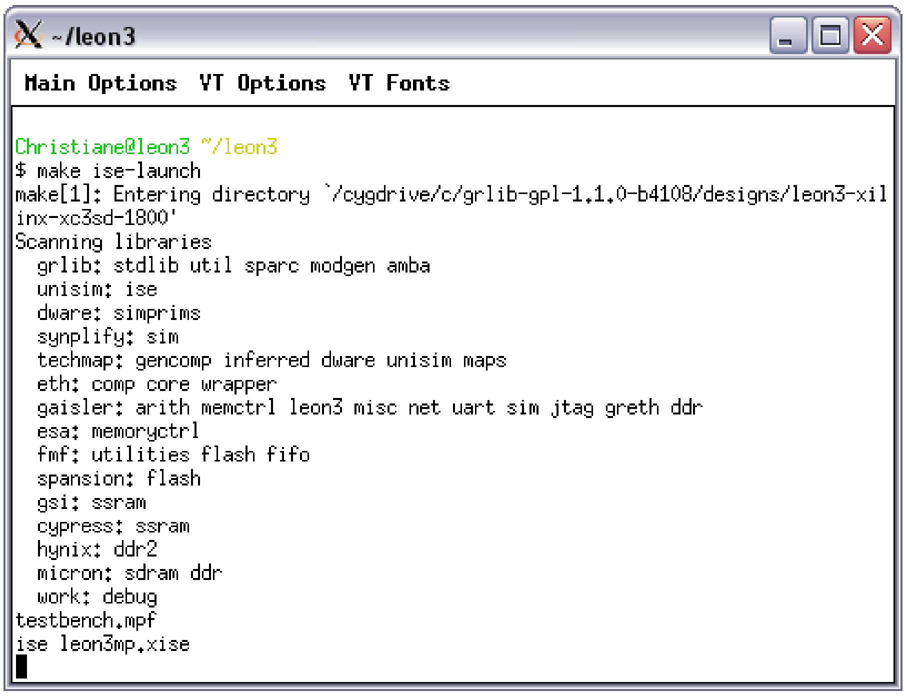
[1] http://www.gaisler.com/cms/index.php?task=view&id=156&Itemid=104

11. Now click the green arrow to the upper left of the process panel. This will start the Synthesize process and thereafter the Place & Route process. Since LEON3 is a large design these processes may take several hours on a modern PC (2012). Unfortunately ISE does not support multi-core processing.

12. When both processes are finished, doubleclick on **Generate Programming File** (Processes panel) to generate the bit-file.

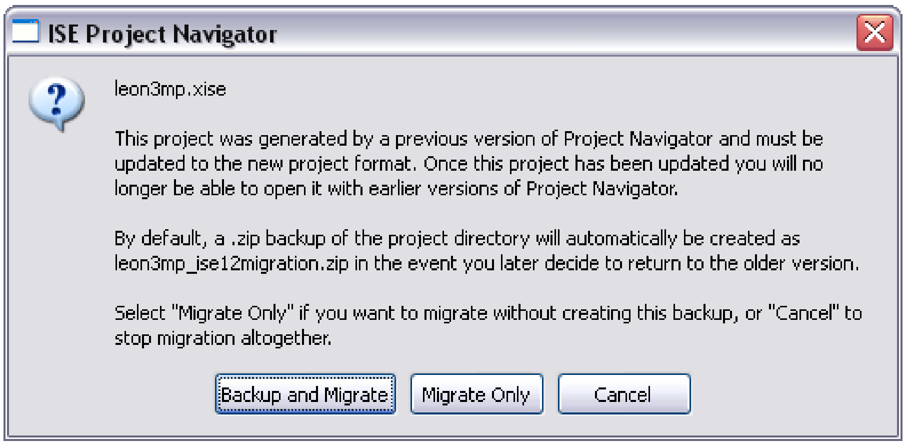
13. The **Design Summary (Implemented)** should look similar to Figure A.17. If you study the **Device Utilization Summary** you will see that the standard design (no FPU!) used for this guide utilizes 66 % of the available slices.

14. Close ISE, type **exit** into your cygwin prompt and manually close the X-Server if necessary. The bit-file in the folder leon3-xilinx-xc3sd-1800 can now be used to create a PROM-file. Please refer to the guides Creating a PROM File for the BPI Flash and Writing a PROM File to the BPI Flash.
    You're done!



```
Christiane@leon3 ~/leon3
$ make ise-launch
make[1]: Entering directory `/cygdrive/c/grlib-gpl-1.1.0-b4108/designs/leon3-xil
inx-xc3sd-1800'
Scanning libraries
  grlib: stdlib util sparc modgen amba
  unisim: ise
  dware: simprims
  synplify: sim
  techmap: gencomp inferred dware unisim maps
  eth: comp core wrapper
  gaisler: arith memctrl leon3 misc net uart sim jtag greth ddr
  esa: memoryctrl
  fmf: utilities flash fifo
  spansion: flash
  gsi: ssram
  cypress: ssram
  hynix: ddr2
  micron: sdram ddr
  work: debug
testbench.mpf
ise leon3mp.xise
```

Figure A.14
Cygwin X-Server terminal output when launching ISE. Step-by-Step the makefile-procedure adds VHDL-Cores as specified in the configuration.



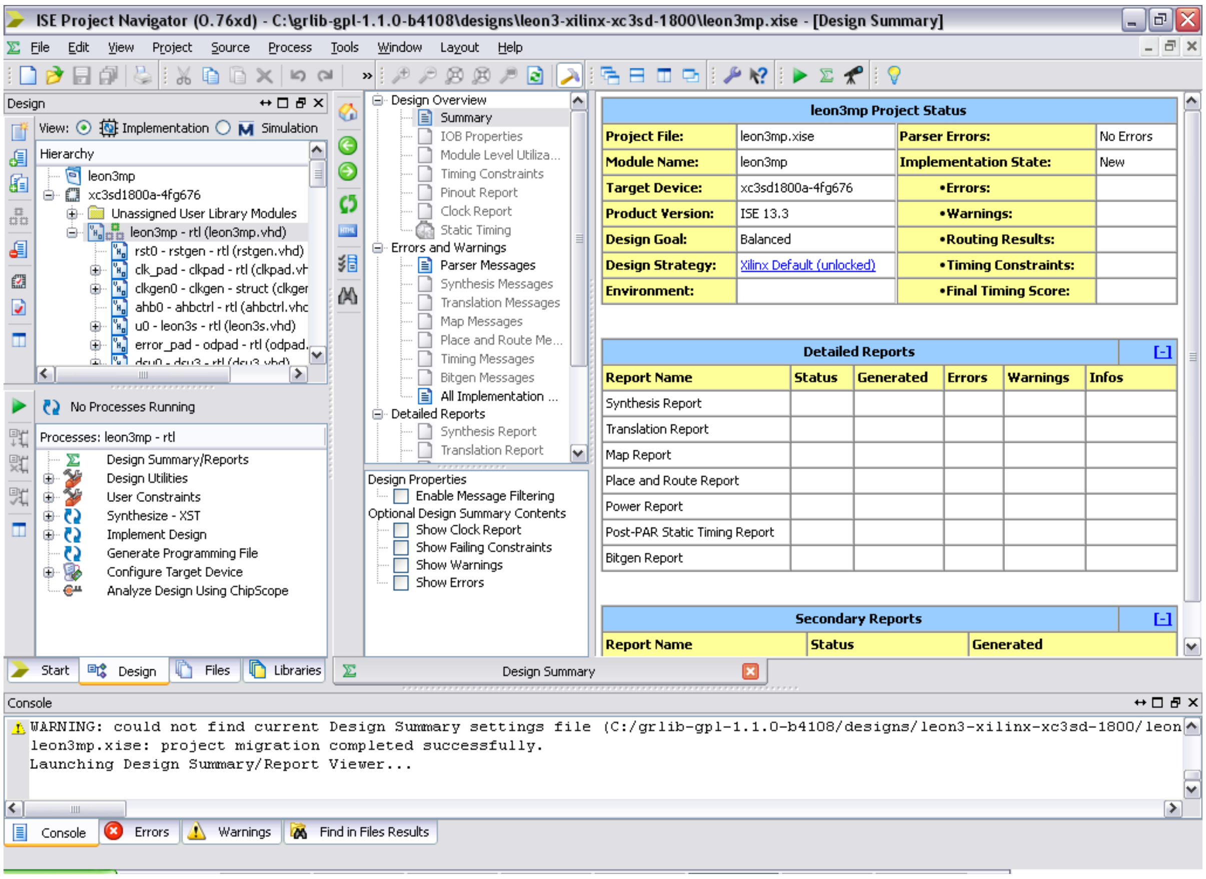
leon3mp.xise

This project was generated by a previous version of Project Navigator and must be updated to the new project format. Once this project has been updated you will no longer be able to open it with earlier versions of Project Navigator.

By default, a .zip backup of the project directory will automatically be created as leon3mp_ise12migration.zip in the event you later decide to return to the older version.

Select "Migrate Only" if you want to migrate without creating this backup, or "Cancel" to stop migration altogether.

Figure A.15
The project migration message box as shown after launching the ISE project for the first time.



leon3mp Project Status			
Project File:	leon3mp.xise	**Parser Errors:**	No Errors
Module Name:	leon3mp	**Implementation State:**	New
Target Device:	xc3sd1800a-4fg676	**•Errors:**	
Product Version:	ISE 13.3	**•Warnings:**	
Design Goal:	Balanced	**•Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:		**•Final Timing Score:**	

Detailed Reports

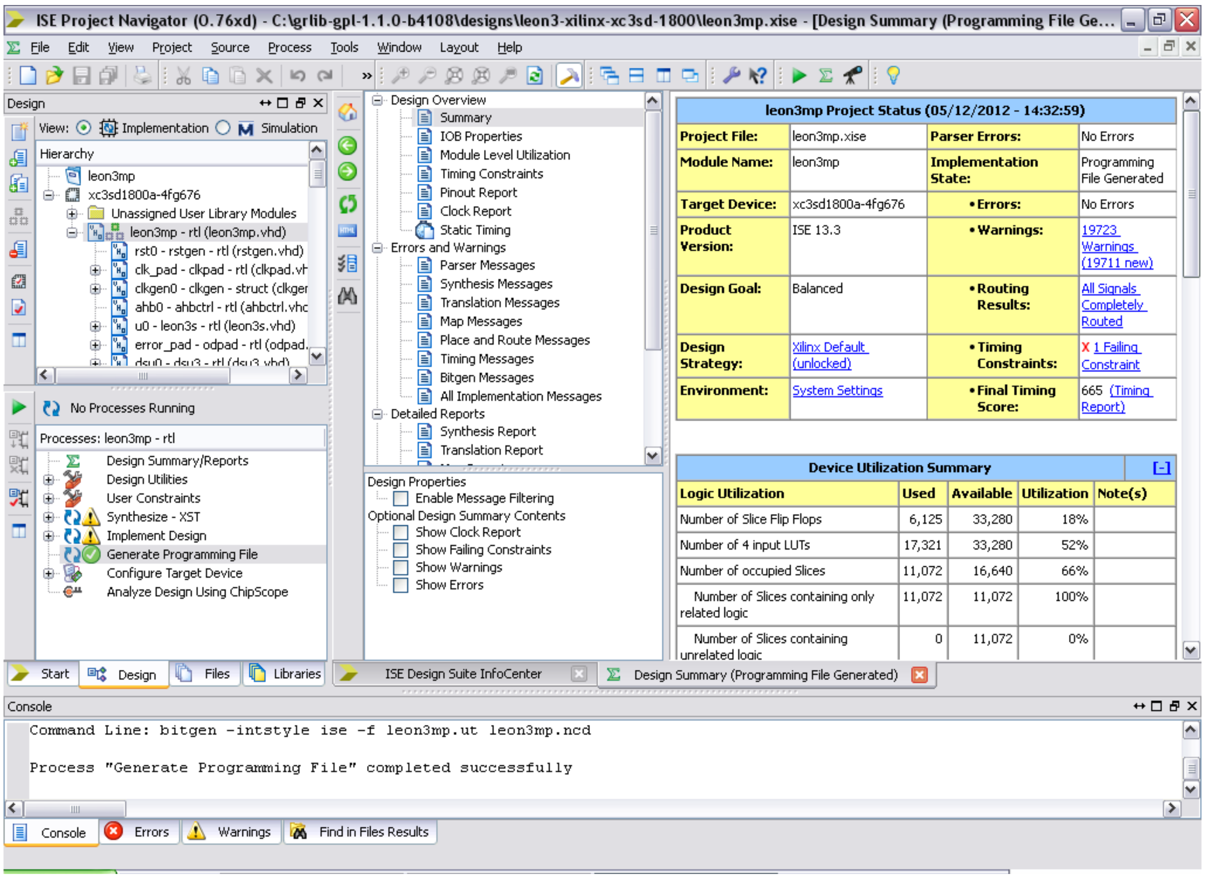
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report					
Translation Report					
Map Report					
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

Secondary Reports

Report Name	Status	Generated

```
WARNING: could not find current Design Summary settings file (C:/grlib-gpl-1.1.0-b4108/designs/leon3-xilinx-xc3sd-1800/leon
leon3mp.xise: project migration completed successfully.
Launching Design Summary/Report Viewer...
```

Figure A.16
Main screen of the ISE Design Suite after opening the newly generated LEON3 project.



leon3mp Project Status (05/12/2012 - 14:32:59)			
Project File:	leon3mp.xise	**Parser Errors:**	No Errors
Module Name:	leon3mp	**Implementation State:**	Programming File Generated
Target Device:	xc3sd1800a-4fg676	**• Errors:**	No Errors
Product Version:	ISE 13.3	**• Warnings:**	19723 Warnings (19711 new)
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	X 1 Failing Constraint
Environment:	System Settings	**• Final Timing Score:**	665 (Timing Report)

Device Utilization Summary [-]

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	6,125	33,280	18%	
Number of 4 input LUTs	17,321	33,280	52%	
Number of occupied Slices	11,072	16,640	66%	
Number of Slices containing only related logic	11,072	11,072	100%	
Number of Slices containing unrelated logic	0	11,072	0%	

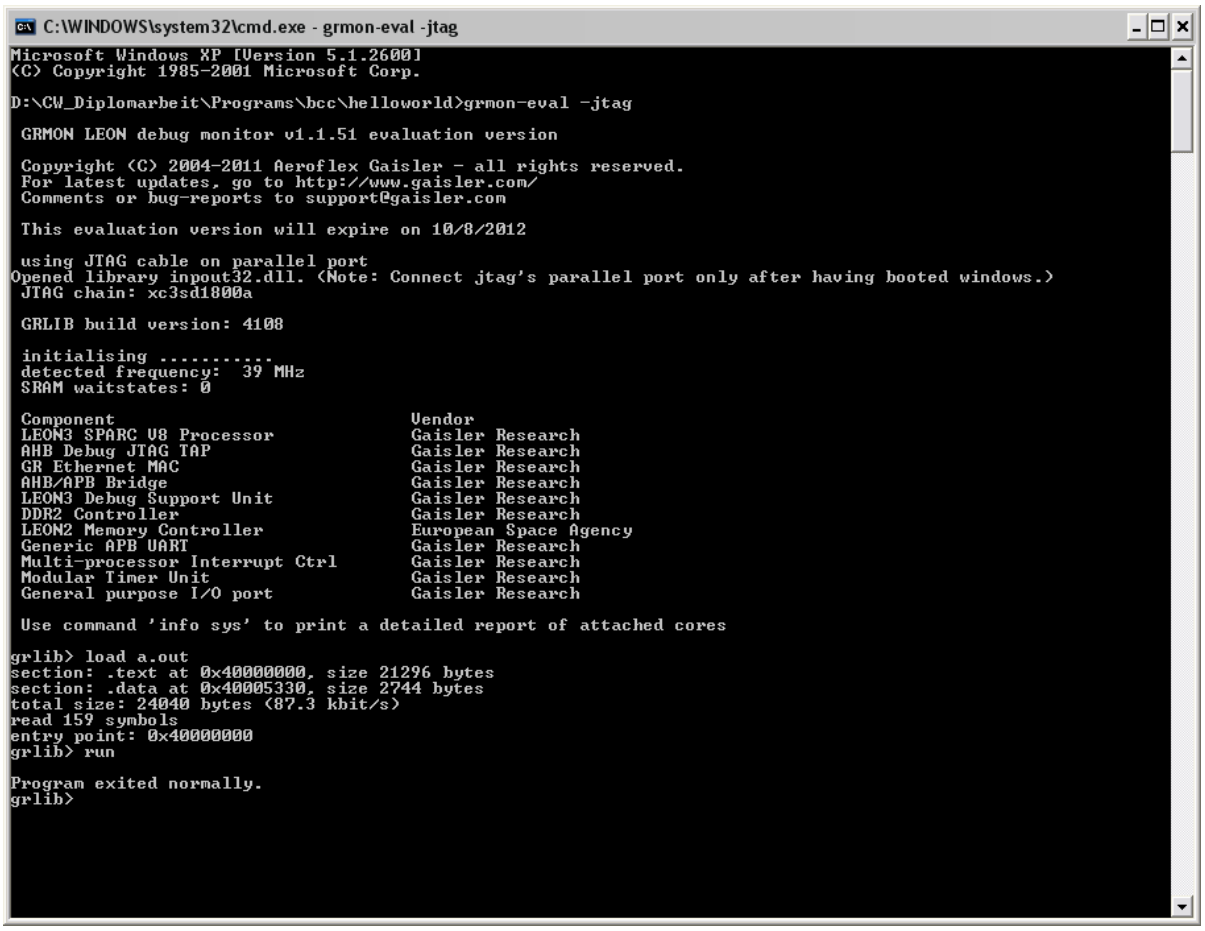
```
Command Line: bitgen -intstyle ise -f leon3mp.ut leon3mp.ncd

Process "Generate Programming File" completed successfully
```

Figure A.17
After the complete implementation process of the design ISE shows the fully populated Design Summary report.

Executing LEON3 Programs using GRMON

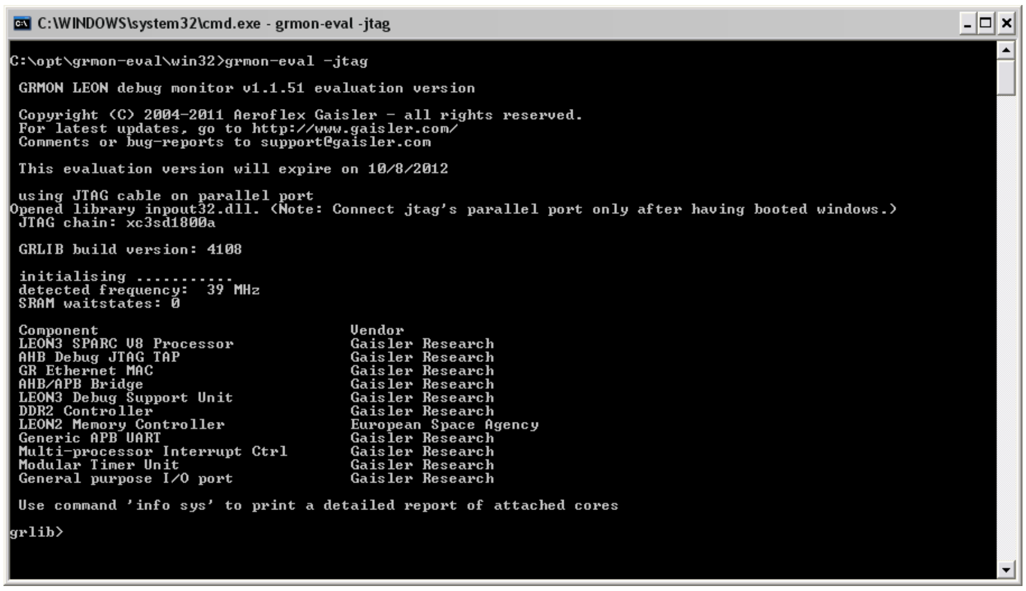
Follow these instructions to load a precompiled program into the LEON3 and execute it.

1. For this guide it is assumed that you have downloaded and extracted the evaluation version of GRMON and have added the folder grmon-eval\win32 to your systems path variable.
2. Open a command prompt and navigate to a folder where you have the compiled (presumably .exe) program you want to execute with LEON3.
3. To start GRMON type **grmon-eval -jtag**. The -jtag option tells GRMON to connect to the board using a parallel JTAG cable.
4. If the connection is successful the main screen of GRMON will be as shown in Figure A.19.
5. You can enter **help** to get a list of the commands available. Type **info sys** for now. This will display a list of all available units in your design. Since this guide uses the standard configuration, no Floating Point Unit is available. As shown in Figure A.20 GRMON the address ranges are also listed where the command and configuration registers of the given component are located.
6. To load a program type **load a.out** where a.out is the name of the compiled program.
7. GRMON now transmits the program to LEON3 and outputs information about the size.
8. The execution is started with the **run** command. GRMON is now blocked until execution is finished and then outputs a message indicating the termination state. A successful load/run procedure is shown in Figure A.18.
9. Type **exit** to leave GRMON.
   You're done.



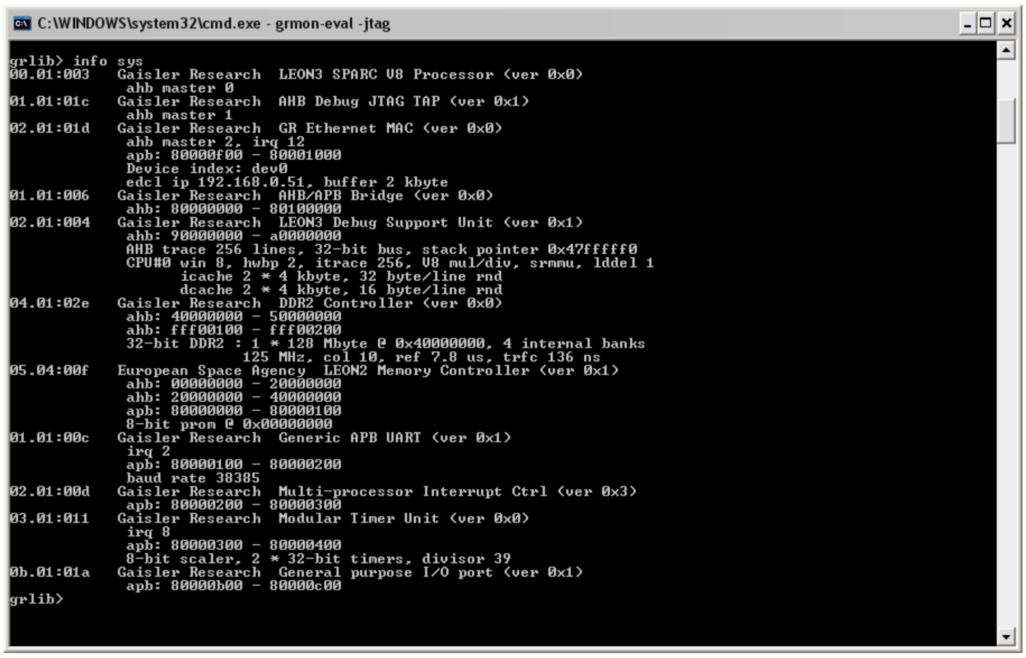
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

D:\CW_Diplomarbeit\Programs\bcc\helloworld>grmon-eval -jtag

 GRMON LEON debug monitor v1.1.51 evaluation version

 Copyright (C) 2004-2011 Aeroflex Gaisler - all rights reserved.
 For latest updates, go to http://www.gaisler.com/
 Comments or bug-reports to support@gaisler.com

 This evaluation version will expire on 10/8/2012

 using JTAG cable on parallel port
Opened library inpout32.dll. (Note: Connect jtag's parallel port only after having booted windows.)
 JTAG chain: xc3sd1800a

 GRLIB build version: 4108

 initialising ...........
 detected frequency:  39 MHz
 SRAM waitstates: 0

 Component                          Vendor
 LEON3 SPARC V8 Processor           Gaisler Research
 AHB Debug JTAG TAP                 Gaisler Research
 GR Ethernet MAC                    Gaisler Research
 AHB/APB Bridge                     Gaisler Research
 LEON3 Debug Support Unit           Gaisler Research
 DDR2 Controller                    Gaisler Research
 LEON2 Memory Controller            European Space Agency
 Generic APB UART                   Gaisler Research
 Multi-processor Interrupt Ctrl     Gaisler Research
 Modular Timer Unit                 Gaisler Research
 General purpose I/O port           Gaisler Research

 Use command 'info sys' to print a detailed report of attached cores

grlib> load a.out
section: .text at 0x40000000, size 21296 bytes
section: .data at 0x40005330, size 2744 bytes
total size: 24040 bytes (87.3 kbit/s)
read 159 symbols
entry point: 0x40000000
grlib> run

Program exited normally.
grlib>
```

Figure A.18
A program was loaded into LEON3 via **load a.out** and executed via **run**. After the execution has terminated (the return statement of the main-function was reached) GRMON indicates that the program terminated normally.



```
C:\opt\grmon-eval\win32>grmon-eval -jtag

 GRMON LEON debug monitor v1.1.51 evaluation version

 Copyright (C) 2004-2011 Aeroflex Gaisler - all rights reserved.
 For latest updates, go to http://www.gaisler.com/
 Comments or bug-reports to support@gaisler.com

 This evaluation version will expire on 10/8/2012

 using JTAG cable on parallel port
Opened library inpout32.dll. (Note: Connect jtag's parallel port only after having booted windows.)
 JTAG chain: xc3sd1800a

 GRLIB build version: 4108

 initialising ...........
 detected frequency:  39 MHz
 SRAM waitstates: 0

 Component                            Vendor
 LEON3 SPARC V8 Processor             Gaisler Research
 AHB Debug JTAG TAP                   Gaisler Research
 GR Ethernet MAC                      Gaisler Research
 AHB/APB Bridge                       Gaisler Research
 LEON3 Debug Support Unit             Gaisler Research
 DDR2 Controller                      Gaisler Research
 LEON2 Memory Controller              European Space Agency
 Generic APB UART                     Gaisler Research
 Multi-processor Interrupt Ctrl       Gaisler Research
 Modular Timer Unit                   Gaisler Research
 General purpose I/O port             Gaisler Research

 Use command 'info sys' to print a detailed report of attached cores

grlib>
```

Figure A.19
GRMON's main screen after launching it with the **grmon-eval -jtag** command.



```
grlib> info sys
00.01:003   Gaisler Research  LEON3 SPARC V8 Processor (ver 0x0)
             ahb master 0
01.01:01c   Gaisler Research  AHB Debug JTAG TAP (ver 0x1)
             ahb master 1
02.01:01d   Gaisler Research  GR Ethernet MAC (ver 0x0)
             ahb master 2, irq 12
             apb: 80000f00 - 80001000
             Device index: dev0
             edcl ip 192.168.0.51, buffer 2 kbyte
01.01:006   Gaisler Research  AHB/APB Bridge (ver 0x0)
             ahb: 80000000 - 80100000
02.01:004   Gaisler Research  LEON3 Debug Support Unit (ver 0x1)
             ahb: 90000000 - a0000000
             AHB trace 256 lines, 32-bit bus, stack pointer 0x47fffff0
             CPU#0 win 8, hwbp 2, itrace 256, V8 mul/div, srmmu, lddel 1
                   icache 2 * 4 kbyte, 32 byte/line rnd
                   dcache 2 * 4 kbyte, 16 byte/line rnd
04.01:02e   Gaisler Research  DDR2 Controller (ver 0x0)
             ahb: 40000000 - 50000000
             ahb: fff00100 - fff00200
             32-bit DDR2 : 1 * 128 Mbyte @ 0x40000000, 4 internal banks
                          125 MHz, col 10, ref 7.8 us, trfc 136 ns
05.04:00f   European Space Agency  LEON2 Memory Controller (ver 0x1)
             ahb: 00000000 - 20000000
             ahb: 20000000 - 40000000
             apb: 80000000 - 80000100
             8-bit prom @ 0x00000000
01.01:00c   Gaisler Research  Generic APB UART (ver 0x1)
             irq 2
             apb: 80000100 - 80000200
             baud rate 38385
02.01:00d   Gaisler Research  Multi-processor Interrupt Ctrl (ver 0x3)
             apb: 80000200 - 80000300
03.01:011   Gaisler Research  Modular Timer Unit (ver 0x0)
             irq 8
             apb: 80000300 - 80000400
             8-bit scaler, 2 * 32-bit timers, divisor 39
0b.01:01a   Gaisler Research  General purpose I/O port (ver 0x1)
             apb: 80000b00 - 80000c00
grlib>
```

Figure A.20
Output of the **info sys** command. You can see the important address ranges of the on-chip components. e.g. The registers of the General Purpose I/O (GPIO) unit are located at 0x80000b00.

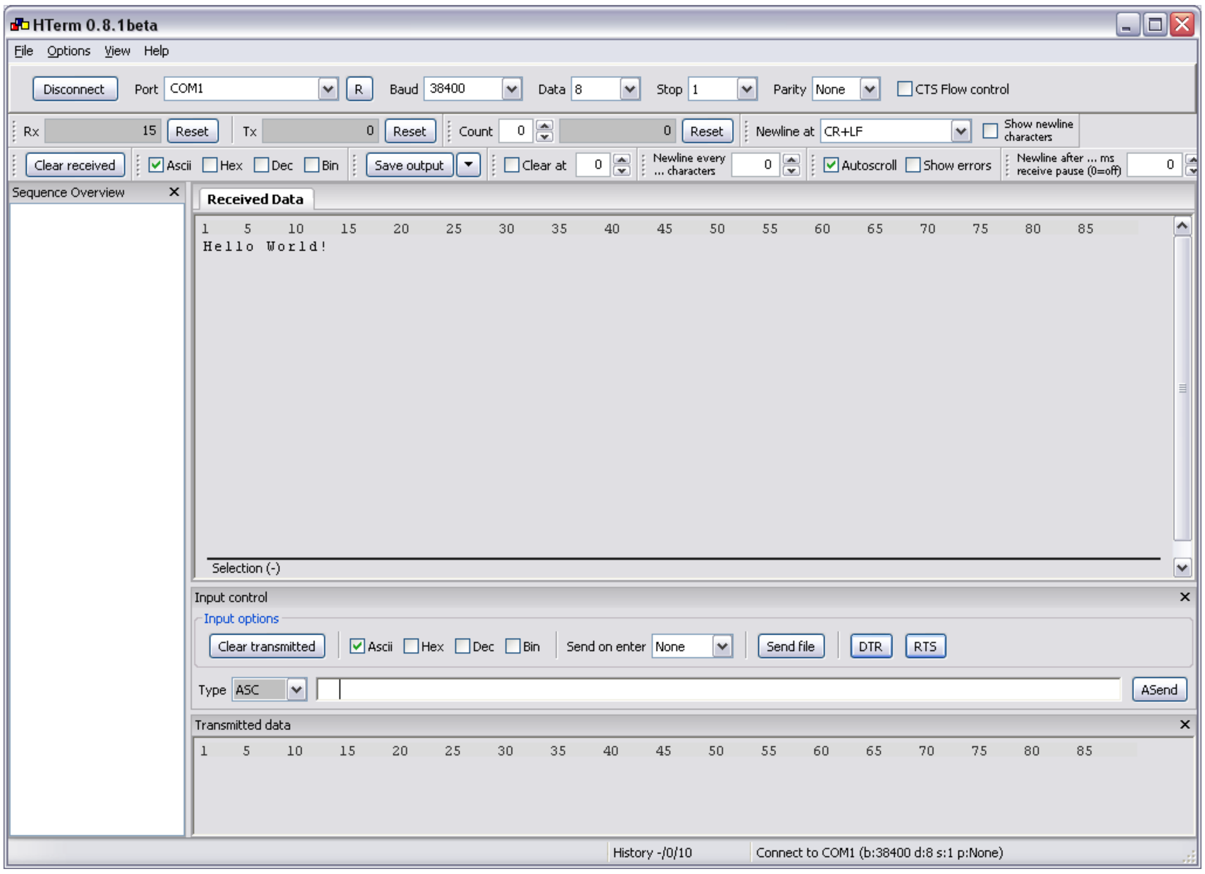
Configuring HTerm for Use with LEON3

Follow these instructions to set up HTerm in a way that it will work together smoothly with the LEON3.

1. You can download HTerm for free even for commercial use from the website[1] of the author Tobias Hammer.
2. HTerm does not need to be installed in any way. Just extract the zip-archive and start **HTerm.exe**
3. Depending on your PC's configuration and hardware you have to choose the appropriate **Port** where the serial cable coming from the LEON3 is attached to your system.
4. For **Baud** choose "38400". This is the baud rate the LEON3 UART uses in the default configuration.
5. For **Data** select "8".
6. For **Stop** select "1".
7. For **Parity** select "None".
8. Make sure the checkbox for **CTS Flow control** is unchecked.
9. Go to **Options** and **Modify Newline At** and select "CR+LF" then select "DEC" from the **Line-end-marker** combobox and add " 13" to the input line. Do not forget the space! Then click **OK**. By customizing the newline in this way you get a much clearer text output in the **Received Data** panel, since the LEON3 always sends a CR+LF+CR when using printf.
10. From the **Newline at** combobox select "CR+LF" (that is the profile you just modified).
11. Uncheck the checkbox **Show newline characters**
12. In the input control panel select "LF" from the **Send on enter** combobox.

[1] http://www.der-hammer.info/terminal/

13. Click **Connect** to establish a connection. There is no feedback if the connection was successful. Due to the nature of the RS-232 protocol only by using the connection you can test it.
14. The final HTerm configuration including the output of the well known "Hello World!" program is shown in Figure A.21.
    You're done!



Port COM1 Baud 38400 Data 8 Stop 1 Parity None ☐ CTS Flow control

Rx 15 Tx 0 Count 0 0 Newline at CR+LF ☐ Show newline characters

☑ Ascii ☐ Hex ☐ Dec ☐ Bin ☐ Clear at 0 Newline every ... characters 0 ☑ Autoscroll ☐ Show errors Newline after ... ms receive pause (0=off) 0

```
1   5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85
Hello World!
```

Selection (-)

☑ Ascii ☐ Hex ☐ Dec ☐ Bin Send on enter None

Type ASC

```
1   5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85
```

History -/0/10 Connect to COM1 (b:38400 d:8 s:1 p:None)

Figure A.21
HTerm configured for use with LEON3.

Using the Terminal Application

Follow these instructions to perform an exemplary session with HTerm and the terminal application.

The sourcecode of this program is contained on the CD delivered with the thesis. You can compile it using the BCC Compiler with the following command:

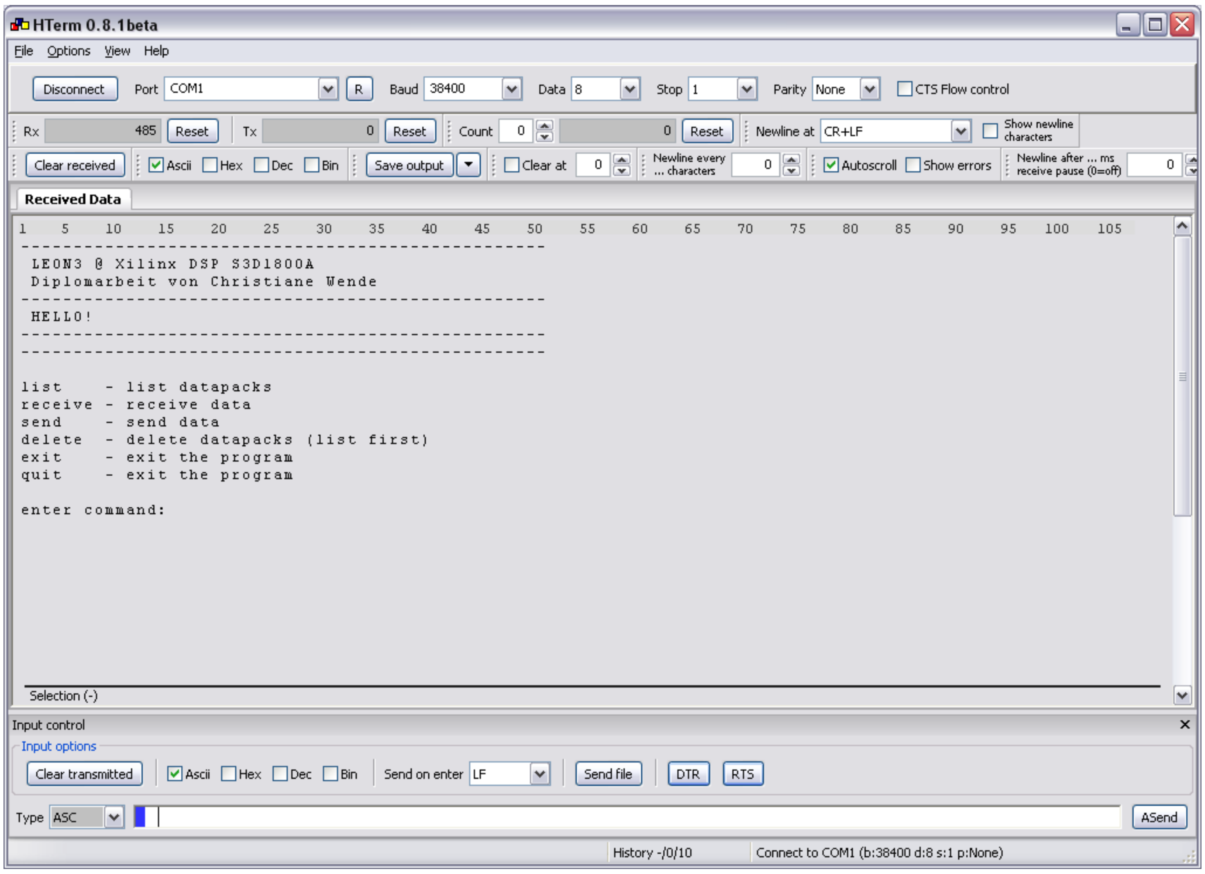
```
sparc-elf-gcc -msoft-float -O2 -Wall main.c
```

Refer to the guide Executing LEON3 Programs using GRMON if you need help executing the program. Refer to the guide Configuring HTerm for Use with LEON3 if you need help configuring HTerm for use with LEON3.

1. Figure A.22 shows HTerm directly after starting the execution of the program. The LEON3 sent a welcome message and a menu to the console. You can now enter the first command into HTerm. Since the LEON3 is waiting in a blocking call to the scanf-function, you have to end your command by sending a newline. To do so, select"LF" from **Send on enter**.
2. Send the command **receive**.
3. Send **5**.
4. Send **hello**.
5. These steps will send 5 bytes to the LEON3. They are not interpreted as characters, but simply sending a text is the easiest way to get some bytes. Figure A.23 shows HTerm after the receive-procedure.
6. Send the command **list**.
7. The LEON3 now sends a list of so called **data packs**. Each one consisting of a starting address, a size, information about its content and the content itself. The output is shown in Figure A.24.
8. Now order the LEON3 to receive another data pack. Send **receive**.
9. Send **6**.
10. Send **world!**.

11. Send **list**.
12. The LEON3 automatically increased its data pack array and added the new data as shown in Figure A.25.
13. You can try to delete data packs but this guide now closes with sending the **exit** command. While shutting down the LEON3 sends some information about the de-allocation of memory (shown in Figure A.26) and then program execution terminates.
    You're done!

It is very helpful to review the source code of the terminal application while performing some live test sessions. You can see how you can write your own code that can be called by a console-command. You can also see how memory allocation is done in a LEON3 program and how registers containing flags are used through binary manipulation to specify what actions the LEON3 should perform.



Port COM1 Baud 38400 Data 8 Stop 1 Parity None CTS Flow control

Rx 485 Tx 0 Count 0 0 Newline at CR+LF Show newline characters

[✓ Ascii] Hex Dec Bin Clear at 0 Newline every ... characters 0 [✓ Autoscroll] Show errors Newline after ... ms receive pause (0=off) 0

```
1    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95   100  105
--------------------------------------------------
 LEON3 @ Xilinx DSP S3D1800A
 Diplomarbeit von Christiane Wende
--------------------------------------------------
 HELLO!
--------------------------------------------------
--------------------------------------------------

list    - list datapacks
receive - receive data
send    - send data
delete  - delete datapacks (list first)
exit    - exit the program
quit    - exit the program

enter command:
```

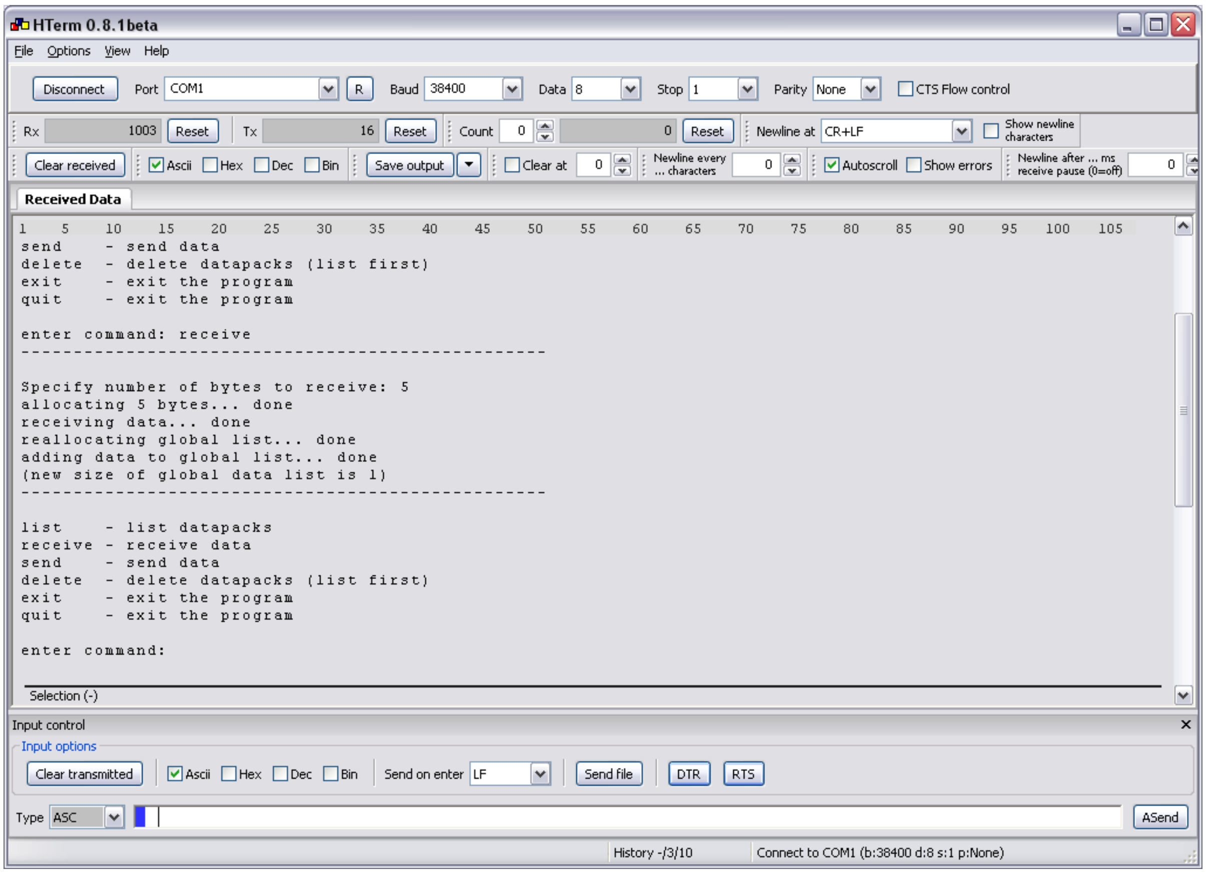
Selection (-)

[✓ Ascii] Hex Dec Bin Send on enter LF

Type ASC

History -/0/10 Connect to COM1 (b:38400 d:8 s:1 p:None)

Figure A.22
The welcome message after starting the execution of the demo program.



Port COM1 Baud 38400 Data 8 Stop 1 Parity None

Rx 1003 Tx 16 Count 0 0 Newline at CR+LF

[✓ Ascii] Hex Dec Bin Clear at 0 Newline every ... characters 0 [✓ Autoscroll] Newline after ... ms receive pause (0=off) 0

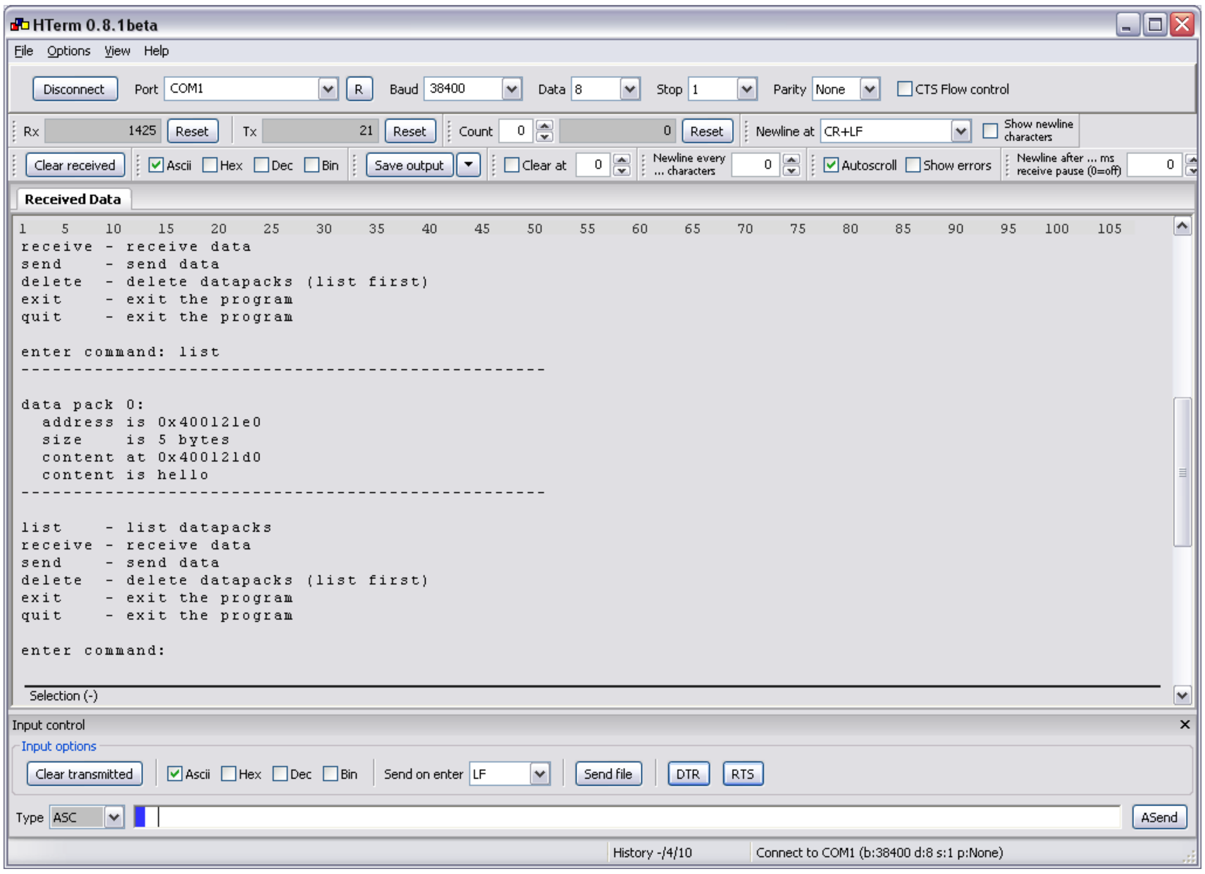
```
send    - send data
delete  - delete datapacks (list first)
exit    - exit the program
quit    - exit the program

enter command: receive
-------------------------------------------------

Specify number of bytes to receive: 5
allocating 5 bytes... done
receiving data... done
reallocating global list... done
adding data to global list... done
(new size of global data list is 1)
-------------------------------------------------

list    - list datapacks
receive - receive data
send    - send data
delete  - delete datapacks (list first)
exit    - exit the program
quit    - exit the program

enter command:
```

[✓ Ascii] Hex Dec Bin Send on enter LF

Type ASC

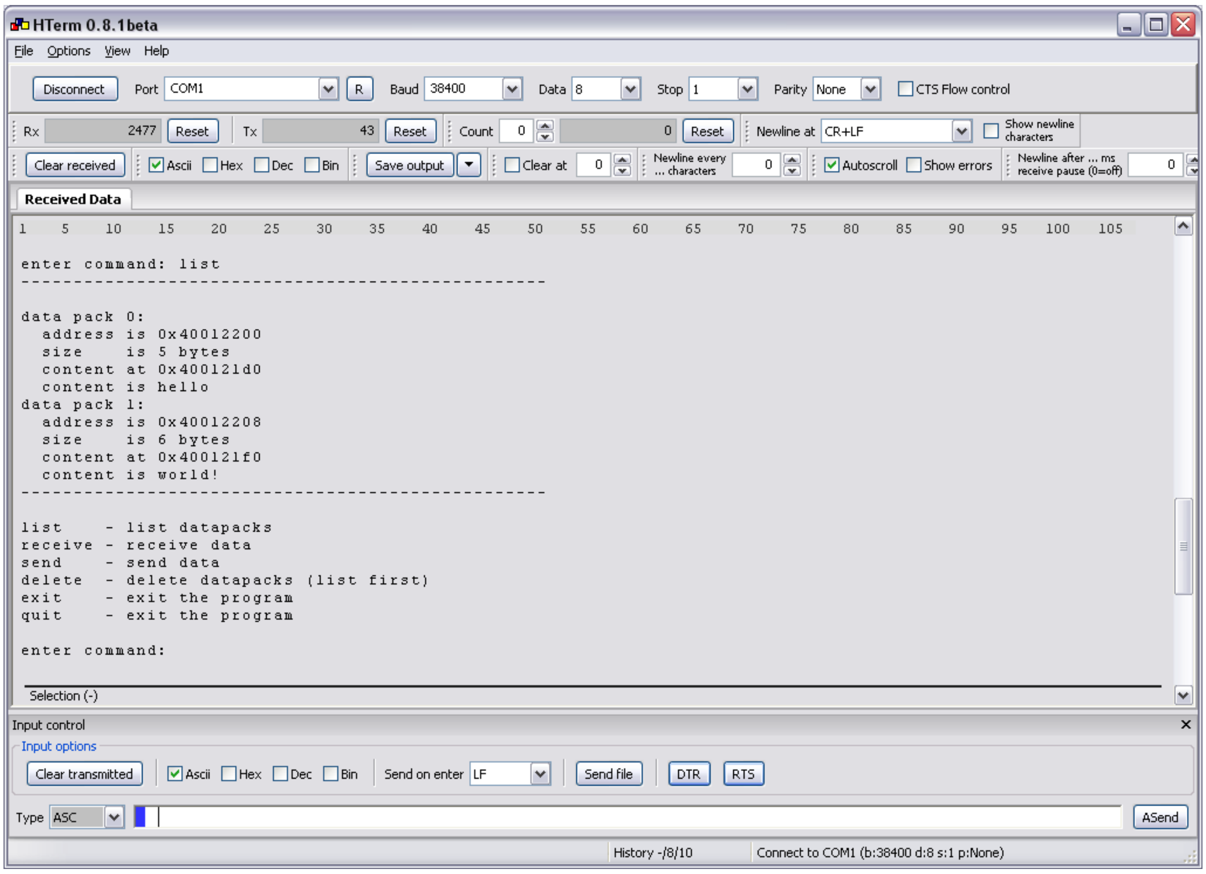
History -/3/10 Connect to COM1 (b:38400 d:8 s:1 p:None)

Figure A.23
The **receive** command followed by a number of 'bytes to receive', and the data itself lead to the shown output.



```
receive - receive data
send    - send data
delete  - delete datapacks (list first)
exit    - exit the program
quit    - exit the program

enter command: list
--------------------------------------------------

data pack 0:
  address is 0x400121e0
  size    is 5 bytes
  content at 0x400121d0
  content is hello
--------------------------------------------------

list    - list datapacks
receive - receive data
send    - send data
delete  - delete datapacks (list first)
exit    - exit the program
quit    - exit the program

enter command:
```

Figure A.24
The LEON3 lists its collection of data packs (only one at this point) and gives additional information about the content.



Port COM1 Baud 38400 Data 8 Stop 1 Parity None CTS Flow control

Rx 2477 Tx 43 Count 0 Newline at CR+LF

Ascii Hex Dec Bin Clear at 0 Newline every ... characters 0 Autoscroll Show errors Newline after ... ms receive pause (0=off) 0

```
enter command: list
--------------------------------------------------

data pack 0:
  address is 0x40012200
  size    is 5 bytes
  content at 0x400121d0
  content is hello
data pack 1:
  address is 0x40012208
  size    is 6 bytes
  content at 0x400121f0
  content is world!
--------------------------------------------------

list    - list datapacks
receive - receive data
send    - send data
delete  - delete datapacks (list first)
exit    - exit the program
quit    - exit the program

enter command:
```

Selection (-)

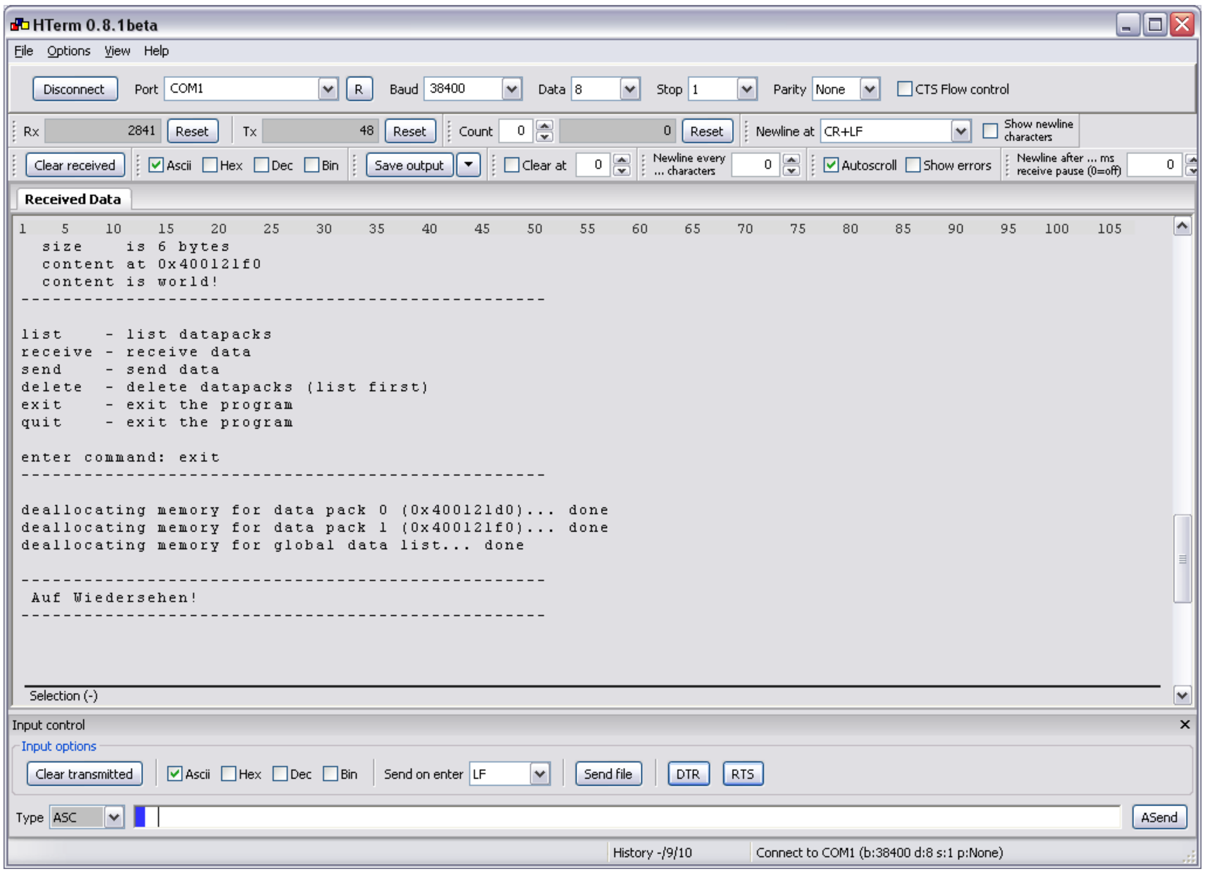
Input control

Input options: Ascii Hex Dec Bin Send on enter LF

Type ASC

History -/8/10 Connect to COM1 (b:38400 d:8 s:1 p:None)

Figure A.25
The data pack list after receiving additional data.



Port COM1 Baud 38400 Data 8 Stop 1 Parity None CTS Flow control
Rx 2841 Tx 48 Count 0 0 Newline at CR+LF Show newline characters
[✓ Ascii] Hex Dec Bin Clear at 0 Newline every ... characters 0 [✓ Autoscroll] Show errors Newline after ... ms receive pause (0=off) 0
[✓ Ascii] Hex Dec Bin Send on enter LF
Type ASC

```
  size    is 6 bytes
  content at 0x400121f0
  content is world!
--------------------------------------------------

list    - list datapacks
receive - receive data
send    - send data
delete  - delete datapacks (list first)
exit    - exit the program
quit    - exit the program

enter command: exit
--------------------------------------------------

deallocating memory for data pack 0 (0x400121d0)... done
deallocating memory for data pack 1 (0x400121f0)... done
deallocating memory for global data list... done

--------------------------------------------------
 Auf Wiedersehen!
--------------------------------------------------
```

History -/9/10 Connect to COM1 (b:38400 d:8 s:1 p:None)

Figure A.26
During shutdown the LEON3 de-allocates used memory and sends some information about this procedure to the console.

Bibliography

Aeroflex Gaisler (2012). *GRLIB IP Library User's Manual.*
– (2013). *GR712RC Dual-Core LEON3FT SPARC V8 Processor.*

Ashden, Peter J. (1990). *The VHDL Cookbook.* Dept. Computer Science, University of Adelaide.

Bagnasco, G. et al. (1999). *The Scientific Instruments On-board XMM.* Review. European Space Agency.

Barret, D. et al. (2013). *The X-ray Integral Field Unit (X-IFU) for Athena+.* http://athena2.irap.omp.eu/spip.php?rubrique2. Technical Report.

Bavdaz, Marcos et al. (2012). *Next-Generation Spacecraft Command & Data Handling System Based on the RAD750 Processor.* Conference Proceedings. Space Telescopes and Instrumentation 2012: Ultraviolet to Gamma Ray: International Society for Optical Engineering.

Bignami, Giovanni et al. (2005). *Cosmic Vision Space Science for Europe 2015-2025.* ESA Publications Division. ISBN: 9789290924890.

Boella, G. et al. (1997). "BeppoSAX, the wide band mission for X-ray astronomy". In: *Astronomy and Astrophysics Supplement* Volume 122 (2), pp. 299–307.

Caramia, Maurizio, Luca Bolognino, and Gianluca Furano (2013). *CAN bus solutions for Data Handling Space Systems.* Noordwijk, The Netherlands: ESTEC.

ESA (2011a). *EJSM-Laplace Assessment Study Report (Yellow Book).* European Space Agency.
– (2011b). *IXO Assessment Study Report (Yellow Book).* European Space Agency.

ESA (2011c). *LISA Assessment Study Report (Yellow Book).* European Space Agency.
– (2013). *LOFT Assessment Study Report (Yellow Book).* European Space Agency.

Fenimore, E. E. and T. M. Cannon (1978). "Coded aperture imaging with uniformly redundant arrays". In: *Applied Optics* Volume 17 (Issue 3), pp. 337–347.

Feroci, M. et al. (2007). "SuperAGILE: the hard X-ray Imager for the AGILE space mission". In: *Nuclear Instruments and Methods in Physics Research Section A* Volume 581 (Issue 3), pp. 728–754.

Fiethe, B. et al. (2003). *Miniaturized Data Processing Unit for Space Instruments.* White Paper. 54th Int. Astronautical Congress.

Fiethe, B. et al. (2007). *Reconfigurable System-on-Chip Data Processing Units for Space Imaging Instruments.* White Paper. IDA TU Braunschweig.

Forman, W. et al. (1978). "The fourth Uhuru catalog of X-ray sources". In: *Astronomy and Astrophysics Supplement* Volume 38, pp. 357–412.

Gaisler, Jiri (1996). *A structured VHDL design method.* White Paper. Gaisler Research.

Garmire, G. P. et al. (2003). "Advanced CCD imaging spectrometer (ACIS) instrument on the Chandra X-ray Observatory". In: *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* Volume 4851, pp. 28–44.

Giacconi, Riccardo et al. (1962). "Evidence For X-rays From Sources Outside The Solar System". In: *Physical Review Letters* Volume 9 (Issue 11), pp. 439–443.

Gliem, F. and B. Gerlach (2001). *DPUs based on COTS.* On-Board Payload Data Processing Workshop. Noordwijk, The Netherlands: ESTEC.

Gruber, D. E. et al. (1996). "The high energy X-ray timing experiment on XTE". In: *Astronomy and Astrophysics Supplement* Volume 120, pp. 641–644.

Gschwender, Michael (2014). *Hardware Implementation and Testing of the Module Back-End Electronics for the LOFT Mission.* Diploma Thesis. Tübingen, Germany: Institute for Astronomy and Astrophysics (IAAT).

Habinc, Sandi (2002). *Suitability of reprogrammable FPGAs in space applications*. Feasibility Report. European Space Agency.

IAF GmbH (2006). *Virtex-4-SX35 Board (Rev. 1.0)*. Datasheet.

IEEE (1996). *1355-1995 - IEEE Standard for Heterogeneous Interconnect (HIC) (Low-Cost, Low-Latency Scalable Serial Interconnect for Parallel System Construction)*. Standard. IEEE Computer Society.

ILC (2003). *MIL-STD-1553 Designer's Guide*. ILC Data Device Corporation.

Kayali, Sammy (2007). *Space Qualification for Semiconductor Devices*. Presentation. California Institute of Technology: Jet Propulsion Laboratory.

Kraft, R. P. et al. (2007). "A Radio through X-Ray Study of the Hot Spots, Active Nucleus, and Environment of the Nearby FR II Radio Galaxy 3C 33". In: *The Astrophysical Journal* Volume 659 (2), pp. 1008–1021.

Krishnaswamy, K. S. (1996). *Astrophysics - A Mordern Perspective*. New Age International. ISBN: 9788122406603.

Lechner, P. et al. (2010). "The Silicon Drift Detector for the IXO High Time Resolution Spectrometer". In: *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* Volume 7742.

Longdon, Norman, ed. (1984). *European Space Science: Horizon 2000*. Vol. 1070. ESA Special Publication.

Lumb, D. H, N. Schartel, and F. A Jansen (2012). "XMM-Newton (X-Ray Mulit-Mirror Mission) Observatory". In: *ArXiv e-prints*.

Martin, Michael (2009). *Entwicklung von Instrumenten zur Untersuchung von hellen Röntgenquellen mit hoher Zeitauflösung*. Thesis. Tübingen, Germany: Institute for Astronomy and Astrophysics (IAAT).

Mazzali, P. A. et al. (2007). "A Common Explosion Mechanism for Type Ia Supernovae". In: *Science* Volume 315 (5813), 825ff.

Michalik, H. et al. (2006). “High-rate CCSDS formatter/encoder plus IDEA encryptor as a single chip solution”. In: *Acta Astronautica* Volume 58 (Issue 12).

Nandra, Kirpal (2014). *Athena: The Advanced Telescope for High-Energy Astrophysics*. Mission Proposal. Garching: MPE.

National Research Council (2006). *Principal-Investigator-Led Missions in the Space Sciences*. The National Academies Press. ISBN: 9780309100700.

Niculae, A. et al. (2006). “Optimized readout methods of silicon drift detectors for high-resolution X-ray spectroscopy”. In: *Nuclear Instruments and Methods in Physics Research A* Volume 568, pp. 336–342.

Parkes, S. and J. Rosello (2003). *SpaceWire ECSS-E50-12A International SpaceWire Seminar (ISWS 2003)*. White Paper. Noordwijk, The Netherlands: ESTEC.

Piro, L. (1997). “The Beppo-SAX expeiment (overview)”. In: *Joint European and National Astronomical Meeting*, p. 284.

Ramadevi, M. C. (2007). *X-ray Binaries*. Bangalore: ISRO Satellite Centre.

Rau, A. et al. (2013). *The Wide Field Imager (WFI) for Athena+*. http://athena2.irap.omp.eu/spip.php?rubrique2. Technical Report.

Rothschild, R. E. (1996). “RXTE Broad Band X-Ray Spectroscopy”. In: *American Astronomical Society Meeting Abstracts* Volume 28, p. 1342.

Santangelo, Aandrea and Rosalia Madonia (2014). “Fifty years of X-ray astronomy: A look back and into the (near) future”. In: *Astroparticle Physics* Volume 53, pp. 130–151.

Schwartz, D. A. (2004). “The Development and Scientific Impact of the Chandra X-Ray Observatory”. In: *International Journal of Modern Physics D* Volume 13, pp. 1239–1247.

Själander, Magnus, Sandi Habinc, and Jiri Gaisler (2009). *LEON4: Fourth Generation of the LEON Processor*. Aeroflex Gaisler.

Snowden, Steve (2011). *Diffuse X-ray Emission in the Milky Way*. Maryland: NASA Goddard Space Flight Center.

TIA (1996). *Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits*. Standard. Tele-communications Industry Association.

Taylor, B. G. et al. (1981). "The Exosat mission". In: *Space Science Reviews* Volume 30 (Issue 1-4), pp. 479–494.

Tenzer, Chris (2012). *LAD Digital Electronics*. Presentation. Tuebingen: IAAT.

Tomayko, James E. (1988). *Principal-Investigator-Led Missions in the Space Sciences*. NASA. ISBN: 9780309100700.

Trenz Electronic GmbH (2008). *Spartan-3 FPGA Micromodule*. User's Manual.

Trümper, J. (1984). "ROSAT". In: *Physica Scripta* Volume 1984 (Issue T7), pp. 209–215.

Underwood, J. H. et al. (1977). "S056 x-ray telescope experiment on the Skylab Apollo Telescope Mount". In: *Applied Optics* Volume 16 (5), pp. 858–869.

Uter, Pascal (2013). *Development of the Module Back End Electronics for the Large Observatory For X-ray Timing*. Diploma Thesis. Tübingen, Germany: Institute for Astronomy and Astrophysics (IAAT).

Voges, W. et al. (1999). "The ROSAT all-sky survey bright source catalogue". In: *Astronomy and Astrophysics* Volume 349, pp. 389–405.

Weisskopf, Martin C. (2011). *The Chandra X-Ray Optics*. Alabama: NASA Marshall Space Flight Center.

Weisskopf, Martin C. et al. (2000). "Chandra X-ray Observatory (CXO): overview". In: *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* Volume 4012, pp. 2–16.

White, N. E. and A. Peacock (1988). “The EXOSAT observatory”. In: *Società Astronomica Italiana, Memorie* Volume 59 (Issue 1-2), pp. 7–31.

White, Nicholas E., Arvind N. Parmar, and Hideyo Kunieda (2009). *The International X-ray Observatory (IXO) Mission Configuration*. Maryland: NASA Goddard Space Flight Center.

Xilinx (2009). *Spartan-3A DSP Starter Platform User Guide*. UG454.

Yra, Pablito B. et al. (2010). *Next-Generation Spacecraft Command & Data Handling System Based on the RAD750 Processor*. Conference Proceedings. 28th AIAA International Communications Satellite Systems Conference: American Institute of Aeronautics and Astronautics.

in ’t Zand, Jean (1996). *Coded aperture camera imaging concept*. Review. SRON Netherlands Institute for Space Research.

*"Interfaces are the prime location to improve a design.
They're also the prime location to screw it up."*

Akin's Law of Spacecraft Design #15