EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATHEMATISCH-
NATURWISSENSCHAFTLICHE
FAKULTÄT

Advanced Practical Course

Digital Electronics for X-ray and Gamma Detectors

Christoph Tenzer

Kepler Center for Astro and Particle Physics
Institute for Astronomy and Astrophysics
Section High Energy Astrophysics
Sand 1
72076 Tübingen

Published: May 8, 2019

http://www.uni-tuebingen.de/de/4203

Contents

The goal of the experiment "Digital Electronics for X-ray and Gamma Detectors" is to become familiar with modern electronics development in the field of X-ray and gamma astronomy.

In this experiment, electronics for operating and reading out a phoswich detector are to be designed using the hardware description language VHDL. Typical steps of a hardware synthesis are: sketching the necessary components and modules, programming and integration of the individual control and analysis processes, simulation of the design on the computer, synthesis process for a XILINX Spartan-3 FPGA and commissioning of the hardware as interface between the detector and a data acquisition system (PC). The detector is then calibrated and its energy resolution determined. Finally, a collimator is used to turn the detector into a gamma-ray telescope and the achievable angular resolution is measured.

In the introductory part of this guide, we have included short tasks to help you understand the instructions. Tasks 1-7 should be prepared or solved by you before the day of the experiment. If you have problems solving the tasks, please bring at least your solution attempts along, we will then discuss the tasks together.

Have a great time experimenting:
Jörg Bayer, Henry Gebhardt, Thomas Schanz, Christoph Tenzer

Translation to English by Inga Saathoff

1 Introduction

X-ray astronomy deals with a part of the electromagnetic spectrum at photon energies from 0.1 keV to about 100 keV, gamma astronomy with energies above it up to the GeV range. Compared to observational astronomy at lower energies, this area of astronomy is characterized by other types of telescopes, detectors and a slightly different type of data analysis.

Radiation in this energy range is almost completely absorbed in the Earth's atmosphere, which is why instruments used for observations have to be brought to very high altitudes (Figure 1).

This necessity makes us understand why X-rays of the sun could not be detected until 1948 with a rocket experiment in New Mexico. Until the discovery of the source Sco X-1 in 1962, the detection of further cosmic objects in the X-ray light was considered highly unlikely due to the low measured X-ray luminosity of the sun. With the discovery of this source, whose emission in the X-ray range exceeds that in the optical range by a factor of 10^4, a window was opened to a new field of research that revolutionized the basic understanding of astrophysics about the structure of stars and the physical processes inside them.

Today, very high-energy gamma radiation ($E > 1$ GeV) can be detected indirectly via the formation of extensive particle cascades (so-called air showers) in the atmosphere using ground-based telescopes.

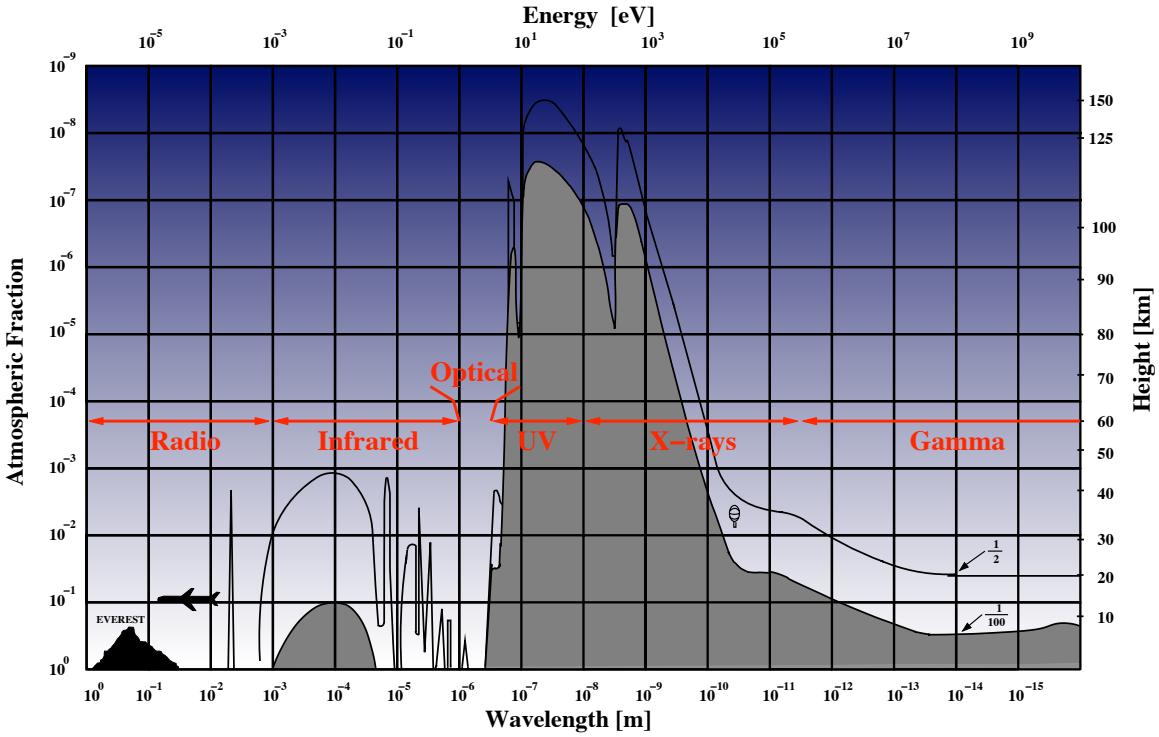

Figure 1: Attenuation of electromagnetic radiation of different wavelengths in the Earth's atmosphere (reproduction of a graphic from Giacconi et al. (1968)).

1.1 Astronomical sources in X-ray and gamma astronomy

Which physical processes exist that generate photon energies above 0.1 keV? According to Wien's law of displacement, a body of temperature T emits the maximum of its radiation at a photon energy of $E_{\max} = 2.8\ \mathrm{kT}$.

With $k \approx 8.6 \times 10^{-8}$ keV/K follows:

$$E_{\max} \quad = \quad 2.4 \times 10^{-7}\ \frac{\mathrm{keV}}{\mathrm{K}} \times T \tag{1}$$

Photons with energies in the range of 50 to 100 keV are emitted by bodies with temperatures in the range of about 100 million degrees! High energy astrophysics therefore deals with the explanation of observations of material under very extreme conditions that normally cannot be produced in the laboratory.

In addition to thermal radiation, there are also non-thermal radiation processes in which photons are also emitted in the keV and MeV range. Some typical examples of thermal and non-thermal astrophysical processes in gamma astronomy are:

- Bremsstrahlung
- Radioactive decay
- Synchrotron radiation
- Inverse Compton effect
- Mass accretion

Typical objects that can be seen in the sky in the X-ray and gamma range are:

- Stars (mostly their hot coronae)
- X-ray binary stars with compact objects
- Supernova remnants
- Gamma-ray bursts
- Galaxy clusters
- Active galaxies (AGN)

Some of the processes mentioned above occur in these objects in combination.

> **Task 1:** Look up the terms (unknown to you) mentioned in the two lists on the Internet and make yourself sufficiently familiar with them so that you can answer the question "*What is...*" reliably at the beginning of the experiment!

Material that falls from a normal star onto a compact companion, i.e. a neutron star or a black hole, in a double star system converts a large part of the potential energy of the material into radiation in the X-ray and gamma range during "accretion". The observed photon spectrum can be used to draw conclusions about the physical mechanisms involved. Accretion is by far the most efficient process in the universe for the conversion to radiation energy (per converted mass).

Accreting black holes often show jets - collimated matter ejections in which the outflowing gas can have velocities close to the speed of light. Such phenomena occur equally in stellar black holes (objects with masses in the stellar mass range) and in supermassive black holes in the centers of active galaxies, which can reach masses of 10^9 solar masses.

In neutron stars with strong magnetic fields, a large part of the accreted material will fall along the magnetic field lines onto the magnetic field poles of the neutron star. The magnetic fields prevailing there are so strong that the movement of electrons perpendicular to the magnetic field lines is quantized (Landau effect). This enables the absorption of high-energy radiation by electrons that transition from one Landau level to the next. This is noticeable as absorption lines in the observed X-ray spectrum. For the transition from the ground state, the line energy is $E \simeq 12\ \mathrm{keV} \cdot B_{12}$, where B_{12} indicates the magnetic field strength in units of 10^{12} Gauss[1]. The observation of such lines is the only direct method known to date for measuring the pole field strength of neutron stars.

In supermassive black holes in the centers of active galaxies, which can reach masses of 10^9 solar masses, the accretion of material is also considered responsible for the high luminosity of these objects. It is estimated that active galaxies have an accretion rate of 1-3 solar masses per year. With the exception of supernova explosions and gamma-ray bursts, these objects are the most luminous in the entire universe. Furthermore, jets are frequently observed in active galaxies: collimated matter outflows in which the outflowing gas can have velocities near the speed of light. These jets emit synchrotron radiation in the radio range, but are also very bright in the X-ray and gamma range. The reason for the high luminosity is that the electrons emitting lower energy synchrotron radiation also scatter this same radiation again to very high energies via the inverse Compton effect (Synchrotron Self-Comptonization).

Task 2: (*Also in the "X-ray CCD" experiment*) Calculate the maximum luminosity that a spherically symmetrical body can have through accretion. This is achieved when the radiation pressure on the accreted matter is greater than the gravitational force with which the accreted matter is attracted. To calculate this so-called Eddington luminosity, you can assume that the accreted matter consists only of hydrogen (why is that a good assumption?) and falls on an object of mass M. The gravitational force on each accreted proton is

$$F_{\mathrm{grav}} = \frac{GMm_p}{r^2} \tag{2}$$

(G: gravitational constant, M: mass of the accreting object, m_p: rest mass of the proton, r: distance of the proton from the accreting object). This is opposed to the force exerted by the radiation pressure. It is mainly exerted on the accreted electrons and is given by (cgs-system!)

$$F_{\mathrm{rad}} = \frac{\sigma_T\ S}{c} \tag{3}$$

[1] As you may have noticed during this practical course, astronomers are quite conservative and still use the units of the cgs system (cm, g, second...) and not the units of the SI you are probably more familiar with. In all likelihood, this will continue to be the case in the future. So please familiarize yourself with such units!

where $\sigma_T = 8\pi e^4/(3m_e^2c^4) = 6.652 \times 10^{-25}$ cm^2 is the Thomson cross section and

$$S = \frac{L}{4\pi r^2} \tag{4}$$

(S: energy flux, L: luminosity).

Why can you equate the two forces here, although F_{grav} acts on protons and F_{rad} on electrons? Specify a formula for the Eddington luminosity and calculate it for an object with $M = 1M_\odot$. The value obtained is in the typical order of magnitude of the luminosity of stellar objects observed in X-ray astronomy. Compare this luminosity with that of the sun ($M_\odot = 2 \times 10^{33}$g, $L_\odot = 3.8 \times 10^{33}$ erg s^{-1}).

1.2 Requirements for astronomical detectors and electronics in the X-ray and gamma range

As Task 2 has shown, the sources of interest to us here can have a very high luminosity. Nevertheless, considerable efforts are needed to obtain information about such sources. The following task should clarify why this is the case:

Task 3: According to Task 2, an astronomical X-ray source has a typical luminosity of 10^{38} erg s^{-1}. For simplicity's sake, assume that one tenth of this luminosity is emitted in the energy range of 10 to 100 keV and that the average photon energy in this range is 20 keV. How many photons per second and square centimeter can you expect in this energy range? Suppose a source distance of 4 kpc and assume that the luminosity is emitted isotropically in all spatial directions.

Consequently, the signal we have to measure is anything but strong. Apart from these low counting rates, another major obstacle is that X-ray and gamma radiation is very well absorbed by the Earth's atmosphere. In order to be able to conduct X-ray and gamma astronomy, one must therefore be above a large part of the Earth's atmosphere, depending on the energy of the photons to be observed. The instruments can be on research balloons (flight altitudes about 40 km), research rockets (flight altitudes up to 100 km) or satellites (above 300 km). Due to these technical difficulties, X-ray and gamma astronomy are very young research areas. The first balloon and rocket experiments were conducted by Herbert Friedman of the Navy Research Laboratory in the 1950s and were dedicated to solar astrophysics. At the beginning of the 1960s, research programs on extrasolar objects were increasingly carried out, first in the USA and later also in other countries such as Germany.

In Germany at the time, balloon-based X-ray and gamma astronomy was funded by a priority program of the German Research Foundation in the 1970s with the Tübingen Astronomical Institute and the Max Planck Institute for Extraterrestrial Physics in Garching near Munich (Figure 2). Since the 1980s, X-ray astronomy has been almost completely dominated by satellites and balloons are only used for testing new technologies and for rapid observation of special events (e.g. supernovae) with particularly suitable instruments.

The low signal of the astronomical sources to be observed and the fact that observations in space are made in the presence of a high detector background generate the following requirements, which are also reflected in the most important parameters of a detector.

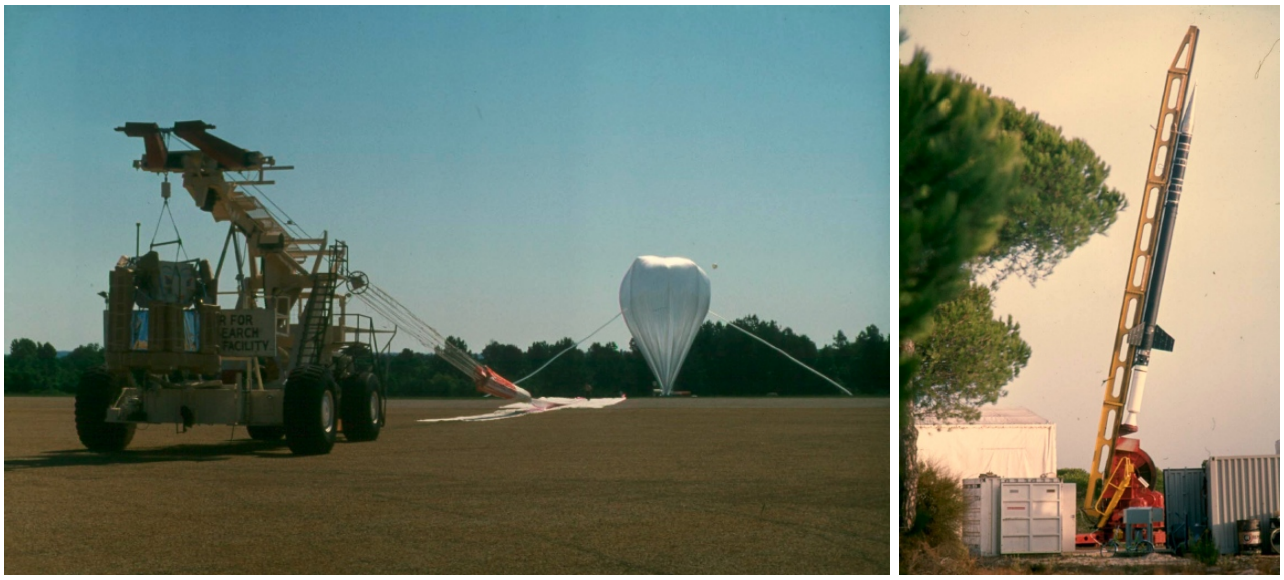

Figure 2: Two experiments in the X-ray range, which were accomplished 1977 with strong participation of the IAAT. Left: Launch of the High Energy X-ray Experiment (HEXE) in Palestine, Texas; Right: Skylark rocket on a mobile launch pad in Huelva, Spain, to observe a moon occultation of the Crab Nebula (Photos: R. Staubert).

The detectors must...

- detect photons with a high probability (*quantum efficiency*),
- provide a large collection area, yet be light enough,
- have a good energy resolution,
- have a good suppression of the sometimes strong background (*sensitivity*),
- achieve a high *spatial resolution* together with the imaging part of the telescope.

Only a few detector types fulfill these properties sufficiently well. The scintillation detectors have proven to be a particularly good "workhorse" in the energy range from > 10 keV up to several 100 keV, which will therefore play an important role in this experiment. Detectors that are mainly used at energies < 10 keV are examined in the "X-ray CCD" experiment.

2 Detection of X-ray / gamma radiation above 10 keV

With the telescopes commonly used in X-ray and gamma astronomy today, *1) the time of impact*, *2) the direction from which the photon arrived* and *3) the energy deposited in the detector* can be measured for each photon detected in the detector. Special detectors also allow the measurement of another property: *4) the polarization plane*. Today's knowledge of the X-ray and gamma sky is based on measurements of these properties of countless photons.

2.1 Scintillation detectors

Scintillators can be organic and inorganic. Organic scintillators are characterized by a very fast response to a signal, but have lower light output than inorganic scintillators and a very poor stopping power for the incident photons which are measured (proportional to the atomic number Z of the material). Therefore, inorganic scintillators are normally used in gamma astronomy.

2.1.1 General principles

The detection of ionizing radiation using scintillation detectors is one of the oldest methods[2]. The available literature is correspondingly large. We particularly recommend chapter 8 of Knoll (1999), the English standard work on detectors. The very well-made slides of a lecture by Christian Joram on particle detectors for summer students at CERN can also be recommended (Joram, 2001).

Scintillation is the emission of light through a body that has been excited by radiation. In our case, this radiation will be gamma radiation, but scintillators are also used to detect electrons, protons or neutrons. The generic term "scintillation" sums up many physical mechanisms, important in the following are in particular

Fluorescence: the emission of visible radiation directly after excitation of the material.

Phosphorescence: the emission of longer wavelength radiation over a period of time longer than the typical period at which the fluorescent light is emitted.

Delayed fluorescence: triggered by the same physical mechanism as fluorescence, but also with a slower time constant; it occurs in organic scintillators.

A good scintillator material should radiate as much of the absorbed energy as possible in the form of fluorescent light so that the photons to be detected can be detected as quickly as possible and so that it is also possible to distinguish photons arriving shortly after each other in time (short dead time). In other words, we want to generate the sharpest possible impulse of optical light from the incident photon.

In a scintillation detector, the weak pulse of visible fluorescence light is amplified by a photomultiplier and converted into an electrical signal (Figure 3). This signal is then further processed by downstream electronics.

[2]According to Grupen (1993), the first scintillation detectors were zinc sulfide screens on which charged particles triggered scintillation flashes, which were then registered with the eye. This method is relatively sensitive: in the green spectral range, even the smallest photon numbers of a few photons/second can be detected with the eye. The sensitivity can be increased even further with strong coffee or strychnine (Grupen, 1993).

In the following, we will first take a closer look at the scintillation mechanisms described above and the materials used in the experiment. Then we will describe the mechanism of signal amplification in the photomultiplier and the post-processing of the signal.

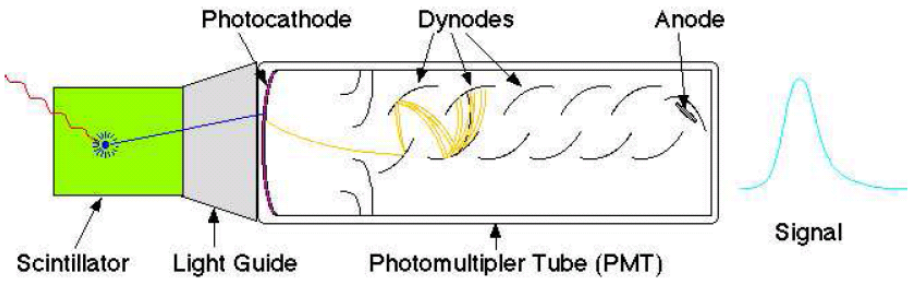

Figure 3: Schematic structure of a scintillation detector (Figure: M. Dahlbohm).

2.1.2 Inorganic scintillators

The most important inorganic scintillators are the halide crystals NaI(Tl) and CsI(Na), i.e. sodium iodide or cesium iodide, to which thallium or sodium is added as an activator. Other scintillator materials you may encounter in physics are CsI(Tl), BGO (bismuth germanate) with the chemical formula $Bi_4Ge_3O_{12}$, the above mentioned zinc sulfide, ZnS(Ag), and barium fluoride (BaF_2). A complete list can be found in Knoll (1999, p. 235).

The *band model*[3] of these crystals is suitable for understanding the scintillation mechanism in inorganic scintillators. Inorganic scintillators are insulators or semiconductors, i.e. the valence band and the conduction band are separated from each other. If a photon is absorbed by an insulator, then its energy can be sufficient to transport electrons from the valence band into the conduction band so that electron-hole pairs are formed (Figure 4).

In a pure crystal, the electrons and the holes would diffuse independently through the crystal and recombine at some point. This recombination produces light of an energy corresponding to the band gap of the crystal. The mechanism just described is comparatively inefficient. On the one hand, a long time passes before recombination takes place, on the other hand, the crystal is not transparent for the recombination light - after all, the energy of the resulting photon is sufficient to transport an electron from the valence band to the conduction band!

Therefore, in scintillators impurities are produced by the admixture of so-called activators. These are selected so that the crystal lattice is disturbed by the activators and energy states arise within the band gap. The (positive) holes that are formed when a gamma quantum hits the scintillator ionize the activator ions because the ionization energy of the activator is lower than that of a typical crystal ion. The electrons that are also formed during the absorption of the photon can then recombine through these luminescence or recombination centers. If the activator material is cleverly selected, the resulting scintillation light is in the visible range and can easily be further processed with a photomultiplier. Furthermore, the energy of the scintillation light is smaller than the band gap, i.e. the crystal is transparent to the scintillation light.

[3]If you really don't know the band model yet, you should definitely change this by at least reading up on the basics.

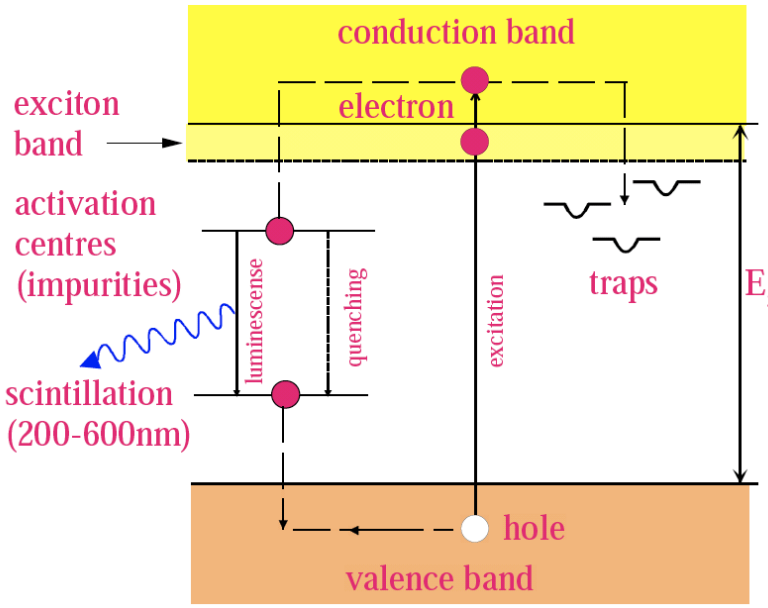

Figure 4: The generation of scintillation light in an inorganic scintillator (Joram, 2001).

The typical recombination time (half-life) is between 50 and 1000 ns, depending on the material. Since the drift time of the electrons through the crystal is negligible, this half-life determines the resulting light signal: It increases quasi instantaneously and then becomes exponentially weaker with the above-mentioned half-life. In some inorganic scintillators, the resulting pulse is more complicated and can for example be represented by the sum of several exponentially decreasing pulses.

This simple behavior can be disturbed by various mechanisms. The most important one is phosphorescent light. In this case, the electron is captured by the activator, but into an excited state, from which the transition to the ground state is quantum-mechanically forbidden. In order to de-excite such a state, the electron must first be excited again, for example by thermal shocks, into a higher state, from which a transition to the ground state is then possible.

This takes a long time compared to the typical length of the scintillation pulse, which is why we are talking about phosphorescence. Furthermore, it is possible that electron and hole do not diffuse independently through the crystal, but as an electron-hole pair, as a so-called *exciton*. Excitons are also de-excited by scattering on activators. The relevant time scale is in the order of magnitude of the time scales for the de-excitation of single electrons or holes. Finally, the excited electrons can also return to the valence band through nonradiative transitions ("signal quenching"), e.g. by releasing their energy through collisions with the crystal lattice.

Due to the complicated band structure of the scintillator crystal, a broadband scintillation spectrum is produced during the excitation of the activators, which has its maximum in visible light for most materials (Figure 5).

The mechanisms just described are quite complicated, but scintillation detectors are very efficient detectors: As the following task shows, about one optical photon is produced in NaI(Tl) per electron-hole pair.

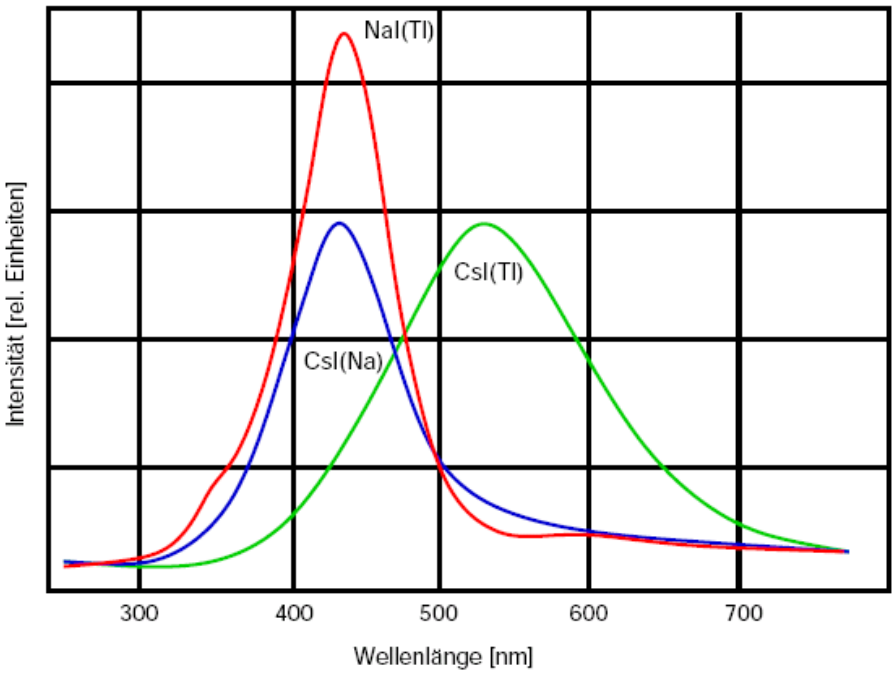

Figure 5: Spectra of typical inorganic scintillators (according to Bicron Technical Note "NaI").

Task 4: As a rule of thumb, it can be assumed that the average energy used to generate an electron-hole pair is three times the band gap of the material. For NaI(Tl) the band gap is about 8 eV, which means that 25 eV are needed to generate an electron-hole pair. Measurements show that the light yield of NaI(Tl) is 12 %, i.e. 12 % of the total energy of the incoming photons is converted into light. For simplicity's sake, assume that all the light is emitted at the maximum of the spectrum (Figure 5).

For a 50 keV photon, calculate the number of electron-hole pairs formed in the crystal and compare these with the number of optical photons generated. Is it true that in NaI(Tl) about one optical photon is produced per electron-hole pair?

In inorganic scintillators the light yield is in good approximation proportional to the number of electron-hole pairs formed. Thus the total number of measured scintillation photons is proportional to the energy of the gamma quantum. This means that scintillation counters can be used for energy measurements. Since the number of scintillation photons increases linearly with the gamma energy (neglecting quenching), scintillation counters are called "linear detectors".

Due to its excellent light yield, NaI(Tl) is the standard detector in gamma spectroscopy. The molar ratio of the thallium admixture is 10^{-3}, the refractive index of NaI(Tl) is 1.85. NaI(Tl) can also be produced in larger volumes (crystal diameter in the meter range).

Due to the high atomic numbers of NaI, NaI(Tl) efficiently absorbs the incident gamma radiation - even in a smaller volume the gamma radiation can be detected well, which is an advantage for satellite experiments. A disadvantage in the handling is that NaI(Tl) is strongly hygroscopic and must therefore be packed airtight.

2.2 Phoswich detectors

As we have seen in Section 2.1.2, NaI(Tl) and CsI(Na) are very similar in their principal properties, such as the spectral form of the scintillation spectrum and the refractive index. The most important distinguishing feature of these scintillators is the different duration of the scintillation pulse. In astrophysical applications, very weak signals must be detected in an environment containing a large number of high-energy particles. Strategies must therefore be developed to reduce the radiation background as much as possible. In a phoswich detector, the just mentioned different durations of the scintillation pulses of NaI(Tl) and CsI(Na) are used for background reduction to identify the origin of the measured scintillation pulse.

Phoswich detectors (from "Phosphor Sandwich") consist of a thin NaI(Tl) crystal that is optically coupled to a thick CsI(Na) crystal. The NaI(Tl) crystal is the actual detector, while the CsI(Na) crystal is used for background reduction: Particles entering the detector "from below or from the side" provide a signal in the CsI(Na). Electronics connected downstream of the photomultiplier analyze the pulse shape and can thus eliminate these events (pulse shape discrimination or rise time discrimination). You will develop such electronics in the first part of this experiment.

Figure 6 shows schematically what a phoswich detector that was actually flown looks like, Figure 7 shows the housing of an array of the detectors used in the experiment. In the High Energy X-ray Timing Experiment (HEXTE) on the Rossi X-ray Timing Explorer (RXTE) shown in Figure 6, the sky section visible by the instrument is limited to just under one square degree (four full moons) by means of a collimator. Furthermore, an ^{241}Am source is mounted in front of the detector[4]. This can be used to monitor the energy calibration of the phoswich detector.

^{241}Am radiation sources are also used in this experiment. This isotope decays into ^{239}Np while emitting an α particle with a half-life of 432.7 years. At the same time as the α particle, a γ photon is emitted at 59.5 keV.

Outside astronomy, phoswich detectors are used for example in medical radiation diagnostics. For medical reasons, it is essential to be able to measure even the lowest possible amount of radioactive materials.

A photomultiplier tube (PMT) is connected downstream of the phoswich detector to detect the scintillation light. It consists of a photocathode from which electrons exit upon irradiation with the scintillation light and several dynodes, between each of which a voltage is applied (Figure 3). The electrons (photoelectrons, pe) primarily generated by the photocathode are focused on the first dynode in the PMT and accelerated by an applied electric field. When hitting the dynode, further electrons are knocked out of there, which are then accelerated to the next dynode, etc. Since the acceleration voltage is high, the number of these secondary electrons is considerably higher than the number of primary electrons, i.e. the initial signal is strongly amplified. If we designate the gain factor of a dynode with Q and the photomultiplier has n dynodes, then the average number of electrons tapped at the last dynode is given by

$$N_{\text{signal}} = N_{\text{pe}} Q^n \tag{5}$$

[4]This americium isotope is not only of interest in science: Just under 0.1 mg ^{241}Am are also contained in standard household smoke detectors.

Typical dynode gain factors are $Q = 6 \ldots 8$, altogether gain factors of 10^6 to 10^8 are achieved.

As we had seen, the number of emitted scintillator photons is proportional to the energy of the gamma quantum:

$$N_{\mathrm{ph}} \quad \propto \quad E \tag{6}$$

These primary photons are converted into photoelectrons (pe, N_{pe}) by the photomultiplier with a certain (quantum) efficiency, and the photoelectrons are "counted" in the detector electronics. Counted means that a certain (mean) number of charge carriers per photoelectron is generated and measured at the anode. This means in particular that

$$N_{\mathrm{pe}} \quad \propto \quad N_{\mathrm{ph}} \propto E \tag{7}$$

Due to Poisson statistics, the measurement of a counting rate is subject to a certain uncertainty. The measurement with the smallest number of events, i.e. the number of photoelectrons, is relevant for the error. If ΔN is the Full Width at Half Maximum (FWHM) of the measured signal, then it is

$$\Delta N_{\mathrm{pe}} \quad = \quad 2.36\sqrt{N_{\mathrm{pe}}} \propto \Delta E \tag{8}$$

The pre-factor 2.36 is explained by the conversion of the Poisson variance ($= N^{1/2}$) into FWHM. Because of Equation 7 and with ΔE as FWHM of the energy measurement it is

$$\Delta E \quad \propto \quad \Delta N_{\mathrm{pe}}. \tag{9}$$

If the error from the photoelectron count statistics were the only error in the energy measurement, the result would be

$$\frac{\Delta E}{E} = \frac{\Delta N_{\mathrm{pe}}}{N_{\mathrm{pe}}} \quad \text{or more generally} \quad \frac{\Delta E}{E} \geq \frac{\Delta N_{\mathrm{pe}}}{N_{\mathrm{pe}}} \tag{10}$$

Thus the relative energy resolution of a scintillation detector is given by

$$\frac{\Delta E}{E} \geq \frac{\Delta N_{\mathrm{pe}}}{N_{\mathrm{pe}}} = 2.36\frac{\sqrt{N_{\mathrm{pe}}}}{N_{\mathrm{pe}}} = \frac{2.36}{\sqrt{N_{\mathrm{pe}}}} \propto \frac{1}{\sqrt{E}} \tag{11}$$

P*lease note that the energy resolution of the detectors is energy-dependent, i.e. must be specified for a given energy!* The statistics described here normally dominates the energy resolution. However, other systematic effects, such as the energy gain in the photomultiplier, variations in voltage, quenching, inhomogeneities in the crystal as well as deviations from the linearity of the detector, etc. can worsen the energy resolution.

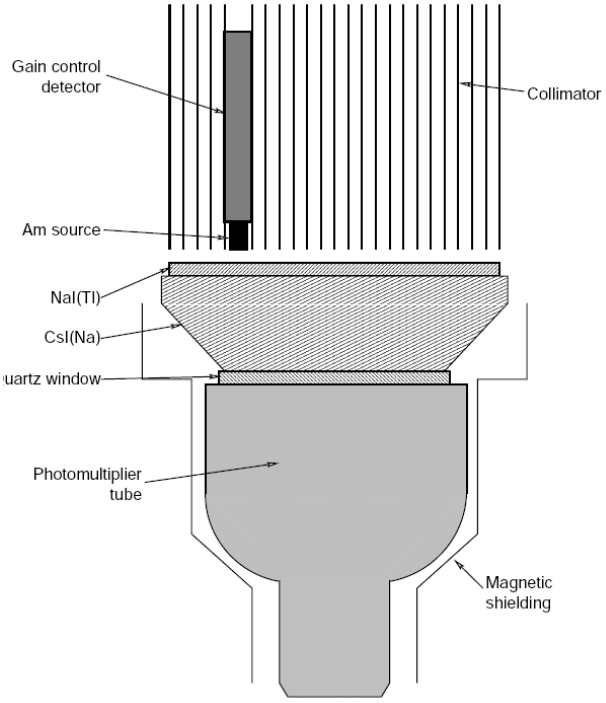

Figure 6: Schematic structure of the HEXTE instrument on the American Rossi X-ray Timing Explorer - the typical structure of an astronomical phoswich detector.

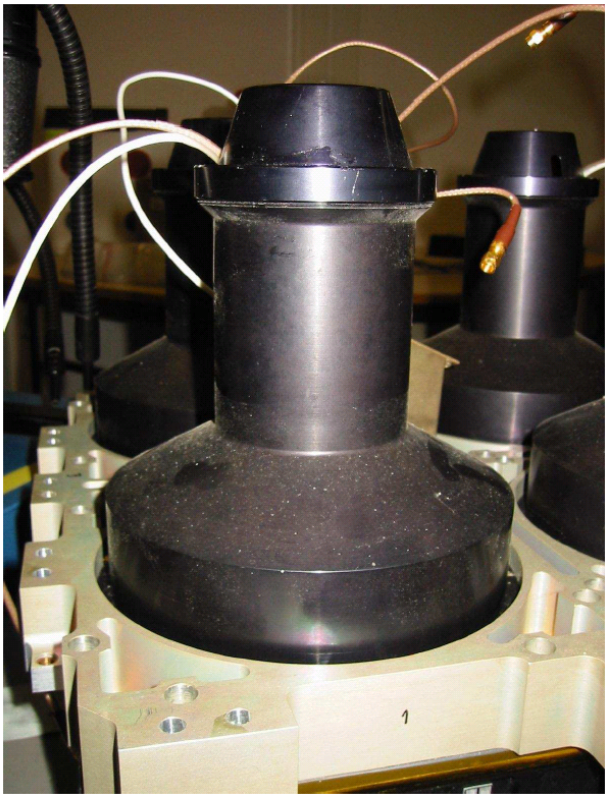

Figure 7: Housing of one of the phoswich detectors used in the experiment.

2.3 Other detectors for radiation above 10 keV

In addition to scintillation detectors, semiconductor detectors have increasingly been used in recent years for the detection of X-ray and gamma photons. Semiconductor detectors have a much smaller band gap, so that a significantly larger number of electron-hole pairs are generated for the same energy of the incident photon. This leads with the above reasoning to an improved energy resolution. A disadvantage, however, is that the detectors often have to be cooled in order to suppress thermal noise.

Semiconductor materials with a high atomic number (Z) are particularly suitable for photon energies above 10 keV (absorption $\propto Z^5$) in order to efficiently detect high-energy photons with relatively thin detectors. In addition to silicon (Si, $Z = 14$), germanium (Ge, $Z = 32$) and cadmium telluride (CdTe, $Z = 48 - 52$) have proven their value. Since CdTe has a relatively large band gap of 1.47 eV (for comparison, germanium has a band gap of 0.74 eV), it can also be used at room temperatures.

2.4 Imaging methods above 10 keV

As you have already learned or will yet learn in the "X-ray CCD" experiment, X-ray radiation at energies ≤ 15 keV can easily be focused on mirrors by grazing incidence. Therefore, an image can still be achieved in this energy range with focusing optics and position-resolved detectors. Above about 15 keV this is no longer so simple. Up to now, other imaging methods have been used for the energy range from ~ 10 keV to some MeV. By "imaging" we mean all methods by which radiation is detected only from a certain region of the sky. This may well be larger than the full moon. Nevertheless, such an "image" is important because it allows to separate the source signal from the

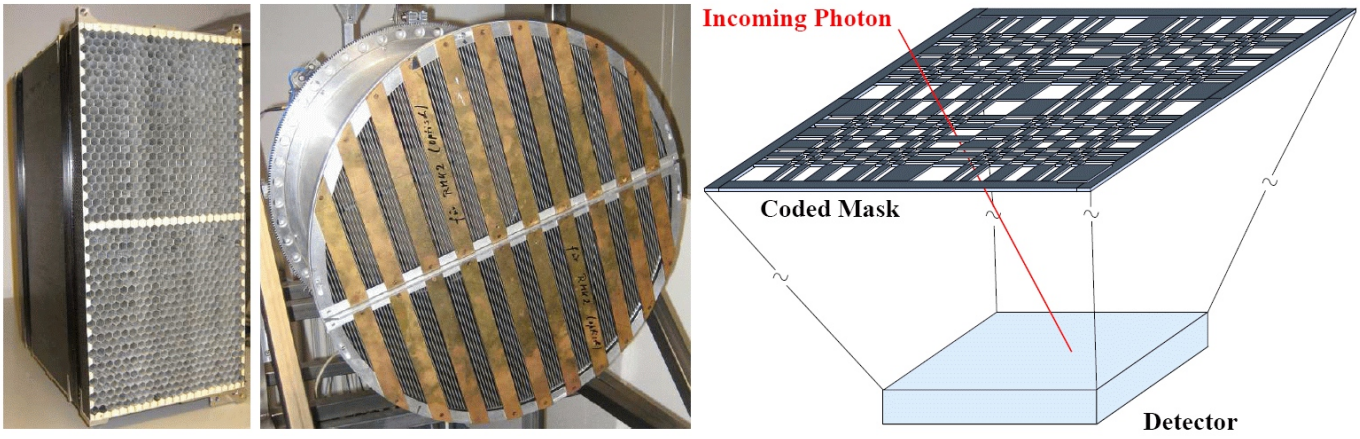

Figure 8: Various collimators that are used in X-ray and gamma astronomy. Left: Static honeycomb collimator; Center: Rotation modulation collimator; Right: Coded mask imaging principle.

(isotropic) background and to represent a spatial distribution of the radiation intensity. Since the particle background is usually very strong, this is absolutely necessary in order to obtain information about the observed sources (source spectrum, temporal variability, etc.).

The methods used for this can be roughly divided into three classes:

1. The field of view of the detectors can be limited with the help of so-called **static collimators**, i.e. only X-rays from a small region of the sky can fall on the detector, radiation from other regions is absorbed in the material of the collimator. This greatly reduces the radiation background in comparison to the source radiation. The simplest collimator consists of an X-ray absorbing material, which is attached as a tube around the detector. The imaging property of such a collimator has a triangular shape: Sources located directly on the optical axis are detected with full flux, the further away the source is from the axis, the fewer photons are detected by it. The angle at which a source with half its flux is detected (parallel incidence of light) is given by

$$\tan \vartheta = \frac{d}{2h} \tag{12}$$

where d is the width of the detector and h is the height of the collimator.

> **Task 5:** A collimated instrument should have a field of view ($= 2\vartheta$) of less than 2° for bright X-ray sources. Take a square detector with 4000 cm^2 collection area. How long does the collimator have to be?

To limit the field of view to a reasonable level with such a simple collimator, very long collimators would be necessary, thus a simple tube collimator is impractical. Since the field of view is proportional to d/h according to equation 12, it is better to place many small collimators in front of the detector. Such a layout is shown in Figure 8 on the left. Here, many small collimators cover the detector, which have the same aspect ratio as a "long" collimator.

The telescope therefore has the same field of view, but is much more compact. Therefore, such a design is better suited to be implemented e.g. on a satellite, because instruments there have to be as space-saving and as light as possible. In practice, collimators are either honeycomb-shaped

because this provides great stability, or they are manufactured in one piece from microchannel plates. In both cases, it must of course be ensured that the wall thickness of the individual collimator cells is as small as possible in order to keep the amount of non-sensitive detector surface small. An example of an instrument using collimators is the aforementioned HEXTE experiment on the Rossi X-ray Timing Explorer. Although collimators can limit the field of view, it is not possible to separate sources that lie within the field of view of the collimator.

2. Real imaging is possible by temporal aperture modulation. A temporal modulation of the source signal can be achieved by tilting the collimator over the source. A better solution are so-called modulation collimators, in which the image is also achieved with a scanning movement of the satellite, or the rotation-modulation collimators (Figure 8, center), in which the source signal is modulated in time by rotating the collimator. The latter were mainly used in balloon experiments in the 1980s. Here, the intensity of sources located at different points in the field of view is uniquely and periodically modulated depending on their position: If the two grids are rotated against each other, the sources - from the collimator's point of view - are covered by the bars with different frequencies and phases. The intensity of the detector signal is then a superposition of the modulated intensities of all sources in the field of view. The position of each source can be determined later on the computer from the period and the phase shift of the light curve.

3. As an alternative to time modulation of the signal, spatial aperture modulation is also an option. The image with coded masks (Figure 8, right) is based on the principle of shadow casting from sources: using a mask made of absorbent material, the shadows of all sources in the field of view are recorded in the image plane with a spatially resolved detector. If more than one source is visible, the position and strength of the sources responsible for the modulation of the signal in the image plane must be recalculated with the aid of complex algorithms. This method offers in comparison to temporally modulated collimators the advantage that even time-varying sources can be reliably reconstructed. Such a mask made of opaque tungsten is used on the INTEGRAL satellite, for example.

3 Modern electronics design for astronomy

The trend in the design of electronic components for scientific and industrial applications has been clearly moving towards increasingly miniaturized components for years. Due to better reproducibility in production, higher reliability and compatibility with IT systems, digital technology has long replaced analog electronics in control and measurement applications. Therefore, electronic design today is almost exclusively concerned with the design of digital circuits.

Another advantage of digital electronics hardware is that complex circuit functions can be described with the aid of mathematical formulas and can be built up from simple, inexpensive elements. Some of these basic components of digital circuits will be briefly introduced in this section.

3.1 Components

3.1.1 IC and ASIC

An IC (INTEGRATED CIRCUIT) is a chip component only a few millimeters in size, which, depending on its function, is made up of up to millions of semiconductor elements - mostly transistors - but also passive components such as capacitors. ICs are characterized by their small dimensions, reliability, short switching times and suitability for mass production. Which function a certain IC executes later is already defined during design and can no longer be changed. In the course of time, a wide range of different ICs has been developed for all conceivable requirements - from simple gates (digital basic circuits such as AND, OR, NAND, Inverter or FlipFlops) to the microprocessors used in mobile phones and computers today. The best-known and most widely used series is the 74-TTL[5] family, which contains many basic digital circuits that can be wired by the developer to complex applications on PCBs[6]. Analog and mixed signal circuits (analog *and* digital) are also available as ICs.

An ASIC (APPLICATION-SPECIFIC INTEGRATED CIRCUIT) is an IC that - in contrast to other ICs that can be used for many purposes - has been designed and manufactured specifically for a particular application. For smaller applications or the production of small quantities, ASICs have already been replaced by FPGAs (see below), which offer more and more space and are becoming faster.

3.1.2 FPGA and CPLD

A FIELD-PROGRAMMABLE GATE ARRAY (FPGA) is a generic form of ASIC in which the function is defined by the user only after the manufacturing process. Depending on the technology (Antifuse[7], EPROM[8], FLASH[9]) it can be (re)programmed one to several thousand times. FPGAs are less powerful than their ASIC counterparts in terms of switching times and power consumption, but have the advantage of shorter development times and lower manufacturing costs.

The basic structure of the FPGA is an array of standard cells, each with a simple programmable lookup table (LUT) and a 1-bit register (flip-flop), see Figure 9. Depending on the number of available inputs, the LUTs can implement any n-digit binary function. The desired function is programmed

[5] Transistor-Transistor Logic: circuit technology for logic circuits.
[6] Printed Circuit Board.
[7] Irreversible programming using the reverse fuse principle.
[8] Erasable Programmable Read-Only Memory: with UV light erasable and then rewritable memory modules.
[9] Electronically erasable and writable memory technology.

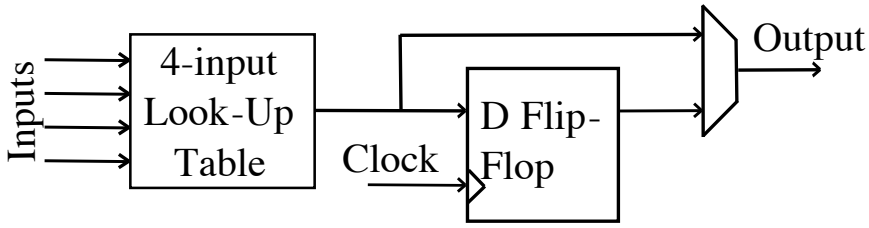

Figure 9: Example of a logic block with LUT and flip-flop.

by storing the defining truth table in the SRAM[10] cells of the LUT, the function is realized by reading out the memory address as determined by the inputs. For a long time LUT structures with 4 binary inputs were common. Newer FPGAs are switching to LUTs with up to 6 inputs to reduce the need for LUT-to-LUT connections to implement functions with more inputs. In addition to the LUTs, the interconnection of the components can also be configured with large degrees of freedom on the FPGA. Multiplexer structures in the standard cells often allow very fast local signal paths, for integrating or bypassing the flip-flop, for feedback from its output, for connecting neighboring blocks and the like. For the more distant connections, there is a grid of immense bus structures between the standard cells to which inputs and outputs can be connected. Other programmable switching components at the grid intersections allow signal distribution over the entire chip.

The hardware design tools described in more detail in the next section are used to define the behavior of an FPGA. This allows the creation of a network list of connections between the standard cells and their transfer to the FPGA. The term programming is therefore to be understood differently in this context than in the creation of software for a processor: In an FPGA, circuit structures are created using hardware description languages or in the form of circuit diagrams and these data are then transferred to the device for *configuration* purposes. This activates or deactivates certain switch positions in the FPGA, which results in a specifically implemented digital circuit.

Even complex structures such as processor architectures can be loaded into a modern FPGA, making FPGAs ideal for hardware development and testing. The idea of changing programming during runtime and adapting it to the task at hand in order to achieve optimum performance for time-critical requirements, such as digital video and audio signal processing, has recently opened up new areas of application for FPGAs in this area.

In contrast to the FPGA built from standard cells, a CPLD (COMPLEX PROGRAMMABLE LOGIC DEVICE) consists of so-called MACRO CELLS, each of which is capable of performing a Boolean operation[11] on its binary input values and outputting the result to the output port, or storing it in a flip-flop until the next CLK cycle. However, due to their internal structure it is often not possible to link arbitrary macrocells, so that hardware designs often cannot be fitted into a given CPLD, even if many macrocells are not yet occupied. Therefore, CPLDs have lost some of their importance lately.

[10] Static Random Access Memory: electronic memory type. Its content is volatile, i.e. the stored information is lost when the operating voltage is switched off.

[11] AND, OR, NAND, etc.

3.1.3 Digital Signal Processors

A digital signal processor (DSP) is a microprocessor specialized in real-time signal processing with an instruction set optimized for this task. DSPs have been available as single chips since 1979 and have become increasingly important since then. Their enormous performance in signal processing has recently also influenced the processor development of CPUs for commercially available PCs, e.g. the MMX expansion, where image and video processing are increasingly coming into the foreground. DSPs are also used to process signals in X-ray astronomy. Since DSPs are basically just microprocessors, programs can be written in programming languages such as C++ or machine language.

3.2 Hardware design tools

With the evolution of the hardware elements, the design tools used for their development were also adapted to the new technical challenges and possibilities. Electronic circuits that used to be designed on the drawing board are now planned on the computer using CAD software. Design editors and hardware description languages embedded in development environments offer the possibility to unravel circuits, create circuit board layouts and check the finished designs for functionality and correctness by simulation before they are implemented as hardware.

Modern design editors (*schematics editors*) provide extensive libraries with graphical representations of electronic components, which can be dragged onto a drawing surface with a mouse click and arranged there to form complex circuit designs. These libraries, which contain both standard elements and high-end components, can be continuously expanded and updated via the Internet. The finished circuits can then be analyzed and their behavior simulated with test signals. Some editors offer useful functions, e.g. to process the design to a board layout, or to create (if possible) network lists for programming FPGAs or CPLDs.

An alternative to graphic design are hardware description languages (HDLs[12]), which can be used to describe entire circuits or even individual hardware components in the form of a model. An advantage over schematics is that not only existing modules can be connected to each other to create new applications, but also the function of one's own modules can be defined in a program by logical, possibly clock-controlled linkage of input and output signals. Similar to the procedure for programming languages, components are instantiated from libraries, functions accessed and values assigned to variables in an ordinary text editor with specially developed syntax. The finished design is synthesized to the network list with a hardware compiler. There are also development environments for HDLs that allow the analysis and simulation of the design before the hardware implementation.

[12]Hardware Description Language.

4 Hardware synthesis with VHDL

In this experiment, electronics for operating and reading out a phoswich detector will be designed using the hardware description language VHDL, which is described in more detail in the next paragraph. First we would like to give you a short introduction to the hardware description language VHDL, which will show you the similarities and differences to programming languages you may know.

4.1 VHDL

The abbreviation VHDL stands for VHSIC[13] Hardware Description Language. VHDL contains many elements of the common, procedurally working programming languages, extended by the parallelism of the PROCESSES and other typical constructs for the hardware circuit design. Also the syntax is quite comparable with that of higher programming languages, so that developers with well-founded programming knowledge have an easier entry into modern hardware design with VHDL than developers with classical electronic know-how. VHDL has been developed by Intermetrics, IBM and Texas Instruments since 1983 and standardized by IEEE[14] in 1987.

4.2 Language concept

VHDL offers the possibility of a complete description of digital electronic circuits. Both their *structure* (structure of individual components and their wiring to each other) and their *function* (description with the help of logical operators) can be mapped in a design. A VHDL design is independent of the hardware and software components used; it can be easily exchanged between developers with different system platforms and target hardware.

Each VHDL design is divided into an ENTITY part and at least one ARCHITECTURE. If there are several architectures for a circuit in the design, a CONFIGURATION statement is required, which allows the selection between the architectures. This basic structure of a VHDL design is shown schematically in Figure 10.

The ENTITY part describes the external interface of the design. All inputs and outputs of the future hardware are specified with their respective signal types, regardless of whether it concerns the design of an FPGA, an IC or an entire circuit board.

The CONFIGURATION allows the developer to choose from several architectures. If only one architecture is available, the configuration part becomes optional (this is the case with most applications in everyday life - configuration is only mentioned here for the sake of completeness).

The ARCHITECTURE contains the actual hardware description of the VHDL design. Two main forms of description are distinguished:

- functional description
- structural description

[13] Very High Speed Integrated Circuit.
[14] Institute of Electrical and Electronics Engineers.

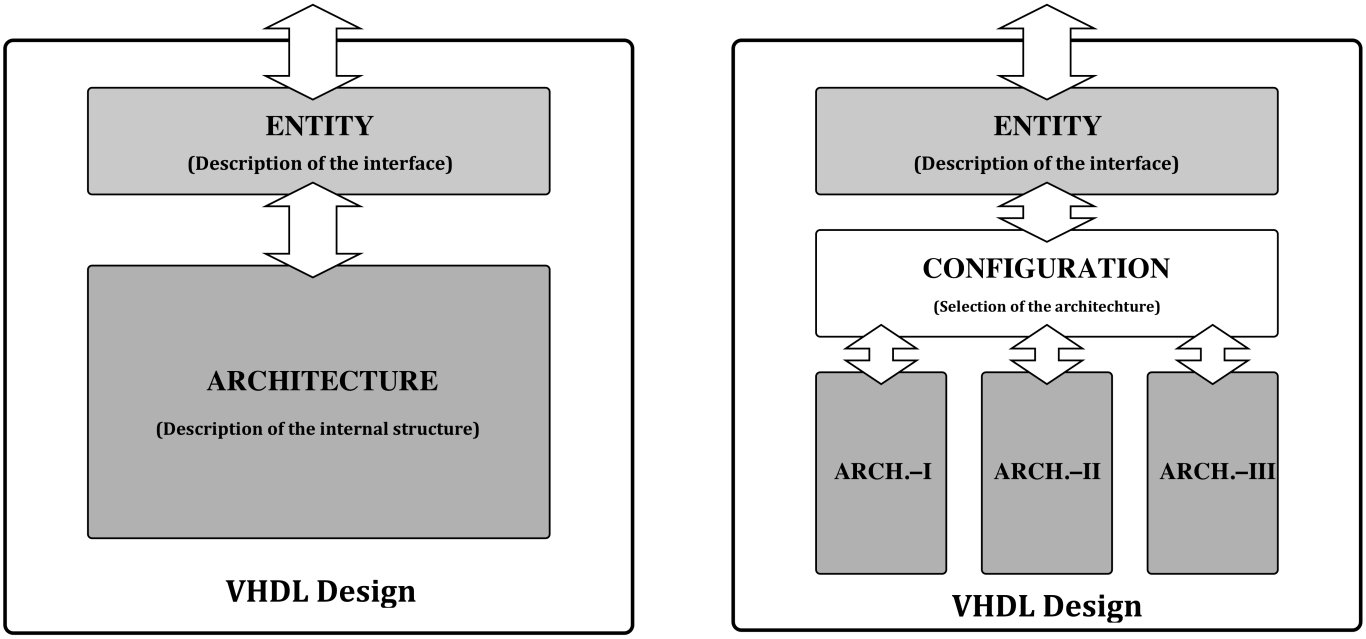

Figure 10: Structure of a VHDL design with only one ARCHITECTURE (left) and with CONFIGURATION for several architectures (right). Even if there are multiple architectures in a design, all use the same ENTITY.

Electronic circuits generally consist of components mounted on a printed circuit board (PCB). The *functional* description is used to model these components. The ENTITY corresponds to the interface of the component (the input and output pins of the IC), the ARCHITECTURE describes the function (therefore *functional* description) of the component instead. With a *structural* description, however, it is possible to model an entire circuit board. The ENTITY then corresponds to the PCB connector and the ARCHITECTURE describes the wiring of the components.

4.3 Architectures

Functional modeling describes the behavior of components. Figure 11 shows such a description. The ENTITY describes the interface of the part, the ARCHITECTURE the function.

Functional modeling enables the tasks of the circuit to be divided into separate functional groups. Like electronic components, these can be developed and tested in separate processes or in different teams before they are finally assembled into a description. Due to this modular principle, complex circuits can easily be assembled from simple groups.

The most important structure for the functional description is the PROCESS. Processes work sequentially like computer programs and are started by signals. Another very important structure are sequence controls, so-called FSMs[15], which will be discussed in more detail later.

4.4 VHDL syntax

The following section gives an overview of the typical constructs used in a VHDL description.

[15]Finite State Machine.

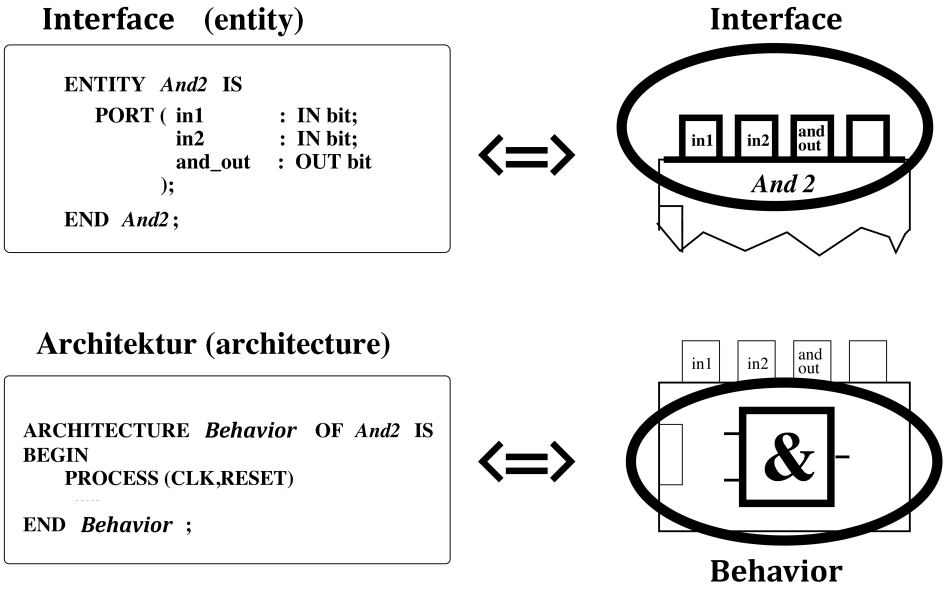

Figure 11: Behavioral modeling of VHDL in architecture: This modeling generates the function of components. Several components can then be connected to form complex systems using their entities.

4.4.1 Comments and identifiers

Comments in the program code are introduced by two simple hyphens/minus signs:

– – Comment

When selecting identifiers for variables, types, process names or functions, the usual restrictions for programming languages apply. Upper and lower case is not important in VHDL and can therefore be used arbitrarily.

4.4.2 Data types, declarations and instantiations

VHDL is "strictly typed" - each variable and each signal must first be declared at the beginning of the architecture with a corresponding type, which is not dynamically/automatically adapted to its use by the compiler. The most commonly used scalar types in VHDL are *integer*, *boolean*, *std_logic* and so-called *enumeration types*. Their definition and use is discussed below:

The type *integer* is predefined in VHDL and comprises all numbers from $-2,147,483,647$ to $+2,147,483,647$. Instantiations (i.e. the declaration of constants, variables and signals) of this type look for example as follows:

```
constant number_of_bytes: integer := 4;
constant number_of_bits: integer := 8 * number_of_bytes;
variable a := 0;
signal counter: integer range 0 to 255;
```

One of the predefined *enumeration types* is *boolean*. The following line of code represents the actual (already implemented) type definition:

```
type boolean is (false, true);
```

You can define your own types very easily by enumerating the possible states. A usage (instantiation) of the type then looks for example as follows. To the right of := the start value (instantiation value) of the signal or variable:

```
signal switch_open: boolean:=false;
```

Since VHDL was designed to describe digital hardware, a central type representing digital values is necessary. To take into account the electrical characteristics of our signals, the IEEE has standardized and predefined a type *std_logic* in the package std_logic_1164, which is used in almost all designs:

```
type std_logic  is ('U',  -- Uninitialized
                    'X',  -- Forcing Unknown
                    '0',  -- Forcing zero
                    '1',  -- Forcing one
                    'Z',  -- High Impedance
                    'W',  -- Weak Unknown
                    'L',  -- Weak zero
                    'H',  -- Weak one
                    '-'); -- Don't care
signal b, c, d: std_logic:='0';
```

This is the type you will work with the most in the experiment. Only use the states '1' and '0', which represent the values 'ON' and 'OFF' respectively.

To obtain multidimensional types, scalar types can be combined in arrays:

```
type point is array (2 downto 0) of integer range 0 to 255;
constant origin : point := (0, 0, 0);
type matrix is array (2 downto 0, 2 downto 0) of integer range 0 to 255;
```

However, the std_logic_1164 package mentioned above already contains an unlimited, one-dimensional array type ('vector') named std_logic_vector from several standard logic values, which can be instantiated as follows:

```
signal my_vec : std_logic_vector(7 downto 0) := "11100011";
```

A vector is an ordered set of individual std_logic signals, but does not yet represent a binary number, even if it already looks like one. One type that can be used for vectors that represent numbers is *unsigned*. It is very similar to *std_logic_vector*, with the difference that there are predefined calculation and comparison operators, so that the vector can always be interpreted as a non-negative binary number. It is defined in the *numeric_std* package, which should be loaded as the very first library in the second line of your code if you want to use it:

```
use ieee.numeric_std.all;
```

The most important operators are:

- `+` addition
- `-` subtraction
- `*` multiplication
- `<` less than
- `>` greater than
- `>=` greater equal
- `=` equal
- `/=` not equal

4.4.3 Data type conversion

VHDL is, as mentioned above, a very strictly typed language. In this case, this means that signals of different types must first be explicitly converted to the correct type ('type-casting') before they can be assigned to each other or compared with each other. The following examples introduce the most important conversions.

- *unsigned* to *std_logic_vector*:

```
my_vec <= std_logic_vector(my_unsign);
```

- *std_logic_vector* to *unsigned*:

```
my_unsign <= unsigned(my_vec);
```

- *integer* to *unsigned*:

```
my_unsign <= to_unsigned(23, 8);
```

  Note that in this case the number of bits of the unsigned vector must be specified as e.g. 8.

- *unsigned* to *integer*:

```
my_integer <= to_integer(my_unsigned);
```

The functions *to_unsigned()* and *to_integer()* are defined in the package *numeric_std*.

4.4.4 Assignments

Signals can be assigned values as follows, e.g.

```
c <= '1';
d <= b AND (NOT c);
b <= Switch1;
my_vec <= "00000000";
my_vec2 <= (1 => '1', 3 => '1', others => '0');
my_vec 3<= "010" & '1' & "0010";
my_unsigned <= x"1234" + x"0001";
```

In the second example, bits 1 and 3 of *my_vec* were assigned a logical '1', and all other bits a '0'. The logical operators **and**, **or**, **nand**, **nor**, **xor**, **xnor** and **not** accept operands of type *boolean* or *std_logic* and also yield a result of each type.

The mapped assignments are translated by the compiler into direct line connections outside of processes (see below). The code is thus not executed sequentially, but creates permanent connections.

> **Task 6:** Consider (purely in terms of formulas, that is, without adhering to the VHDL syntax), how you could realize the following 'voting system' with logical operators (maybe with the help of a truth table). The aim is to switch a lamp on when at least half of four input switches are set to 'on'.

4.4.5 Processes

Processes are a central component of most functional descriptions. They react to signals and process a list of commands. They modify other signals that serve as output of the process. A VHDL description can contain any number of them. Processes can start simultaneously if they are triggered by the same signal. This maps the parallel structure of the electronics.

The most important characteristics of processes are:

- Processes are started by signals in the SENSITIVITY LIST in the process header.
- Processes are parallel structures, i.e. any number of processes can start simultaneously.
- Processes manipulate signals (process output).
- Processes can declare local variables (in the declaration part).
- Process signals must be declared in the declaration part of the architecture.
- Variables of the process change at the moment of assignment.
- Signals of the process only change after the end of the process (delta cycle).

Figure 12 shows the basic structure of a process. The declaration part contains the process header with the process name and the sensitivity list. The sensitivity list is the content of the parenthesis after the word PROCESS. It contains the signals that start (trigger) the process. Start means that the commands of the instruction part are processed. Each change of a signal of the sensitivity list causes the start of the instruction part. In the example in Figure 12, this means that with each change of state of 'CLK' or 'RESET' the process is restarted, i.e. twice per 'CLK' cycle. In this case, the 'RESET' signal could also start the process outside of a 'CLK' cycle (asynchronous with 'CLK').

If 'RESET' is removed from the sensitivity list, the reset is performed synchronously with the 'CLK' signal edge. The state of the 'RESET' signal is then only queried for each 'CLK' edge.

Within processes, there are essentially two particularly frequently used conditionals - the *if-then-else* instruction and the *case* construct. An example illustrates the use of IF. The signals are defined as above.

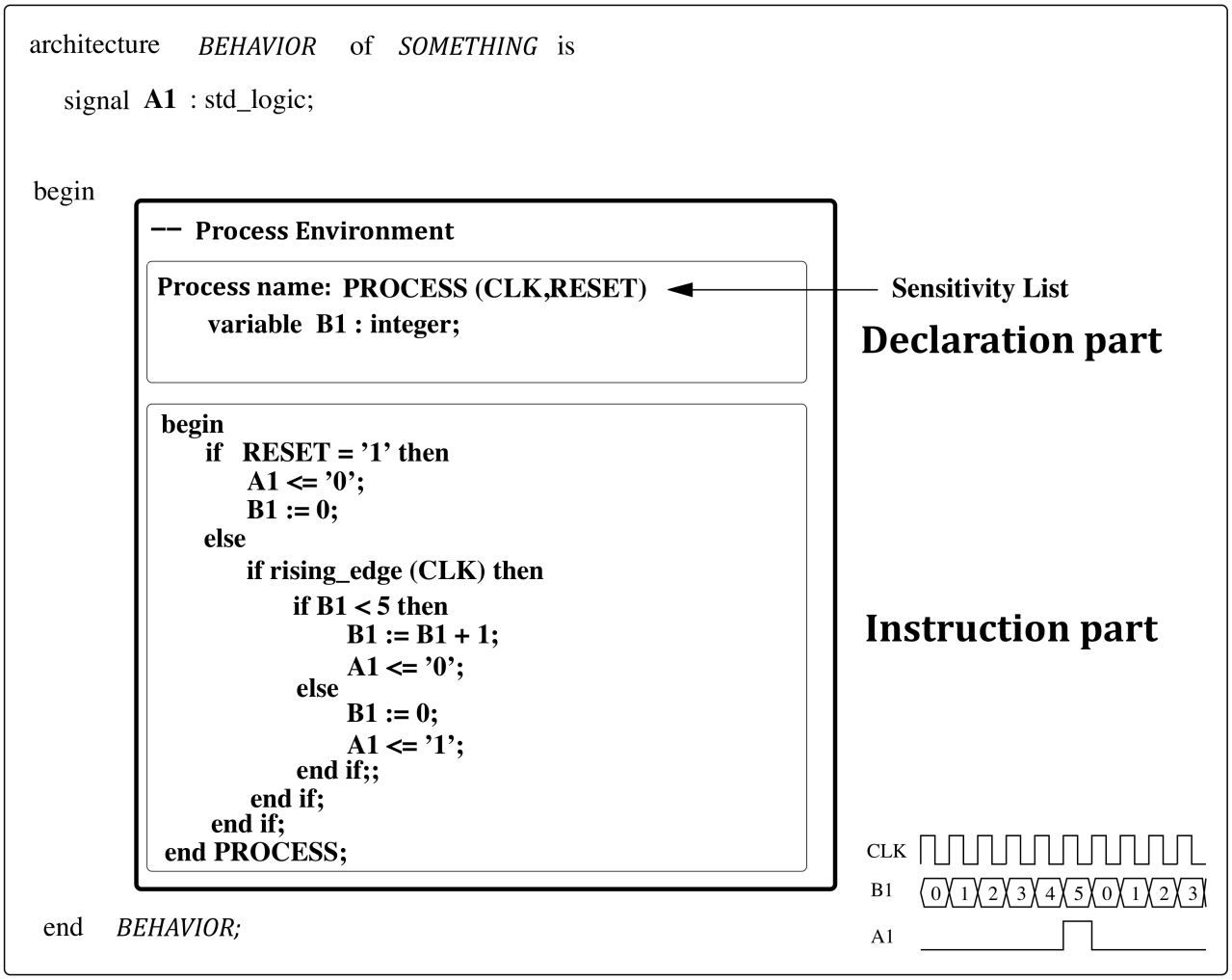

Figure 12: The example shows the structure of the process environment. Processes also consist of a declaration and a instruction part. The process is triggered by signals in the sensitivity list of the process header, or by a wait instruction in the instruction part.

```
if unsigned(my_vec) < my_unsigned(15 downto 8) then
    my_vec <= std_logic_vector(my_unsigned(15 downto 8));
elsif b = '1' and my_vec(7) = '0' then
    Lamp <= '1';
else
    Lamp <= '0';
end if;
```

Sometimes it is necessary to repeat several almost identical assignments. For this purpose, *for* loops can be used as in the following example. Note that in this case the *for* loop is not run sequentially, but the compiler simply creates eight parallel connections in the FPGA! To really do things one after the other, you need a flow control (see below).

The two signals *PHY_DATAOUT* and *crc16_buf* are vectors like *my_vec*.

```
for i in 0 to 7 loop
    PHY_DATAOUT(i) <= not crc16_buf(7-i);
end loop;
```

In contrast to signals, variables use ':=' instead of '<=' for assignment. The main difference to signals is that variables change immediately. Signals are only assigned on the clockedge at the end of the process or at the start of the process. Signals are also declared in the declaration part of the architecture, variables in the declaration part of the process. In addition, variables do not retain their value until the next start of the process, while signals used in the process are stored in registers.

Task 7: Suppose the following assignments concerned signals and were in a process that had just been started for the first time. Previously, all signals were initialized with ones. What is the value of the signals after a clock cycle? What would change if they were variables rather than signals?

```
b <= '0';
my_vec <= "010" & b & "0010";
my_unsigned <= x"1234" + x"0001";
my_unsigned(15 downto 8) <= unsigned(my_vec);
```

The number of operations in a process has a critical influence on the maximum achievable operating frequency of a design. The synthesis program defines these so that all commands of a CLK-synchronous process can also be executed within a CLK cycle. For this reason it is advisable to distribute complex operations to several processes and to coordinate them, e.g. by a flow control or STATE MACHINES.

4.4.6 State machines

Another important structure in functional modeling are so-called finite state machines or FSMs. An FSM is a logic circuit that runs through a sequence of states controlled by pulsed or often periodic external signals. Their theory forms the basis of all forms of automation. Thus they also represent a central concept for the development of digital applications with VHDL. In VHDL, FSMs are represented by processes with a case structure.
Components of an FSM are a signal in which the current state is recorded, a TRANSITIONAL COMBINATORIAL CIRCUIT that controls the change between these states and an OUTPUT COMBINATORIAL CIRCUIT that assigns values to signals depending on the current state.

Types that describe states in an FSM, for example, can be defined and instantiated as *enumeration type* in the following way:

```
type alu_function is (disable, pass, add, subtract, multiply, divide);
signal my_alu : alu_function;
```

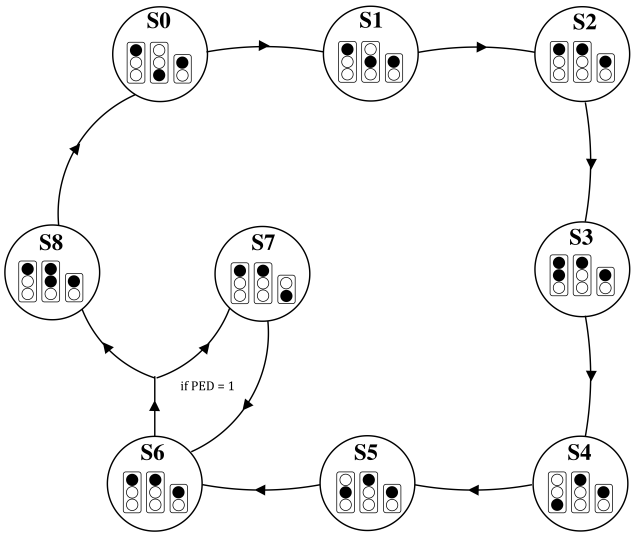

Figure 13: Sequence of the states S0 to S8 of a traffic light circuit, each of which merges into the other after several seconds, the phases in which a traffic light shows 'green' being extendable by any factor 'k'. After state S6, the transitional switching network branches once to S7 and back again to S6 when the pedestrian button is pressed.

Case statements are often used (as here) to implement FSMs (see below). All values that the referenced signal can take must always be specified in the *case* construct. The keyword *others*, which is listed last, helps here:

```
case my_alu is
    when disable =>    -- do nothing
    when add =>
        my_unsigned <= my_unsigned + my_unsigned2;
    when subtract =>
        my_unsigned <= my_unsigned - my_unsigned2;
    when others =>
        my_alu <= pass;
end case;
```

The simplest example of an FSM application in electronics is a traffic light circuit. The four traffic lights at a traffic intersection should alternately allow one street to pass through and show 'red' for the other street (see Figures 13 and 14).

4.4.7 Subprograms, packages and libraries

For the sake of completeness, *functions*, *procedures*, *packages* and *libraries* should be mentioned here, which serve to make the code clearer and more modular. However, this is usually already achieved through components and processes. Functions can be passed multiple arguments similar to mathematical functions and they return exactly one value. Procedures are very similar to functions, but stand for a complete VHDL statement and do not return a value.

4.5 Components

A *component* is a complete VHDL design from a previous development phase or a prefabricated library. Components can also be structural descriptions and contain components themselves. This creates a hierarchy of designs in large-scale projects whose degree of abstraction increases upwards (see Figure 15). The lowest level is always functionally modelled, the upper level is often purely structural. In the intermediate levels it is possible to mix functional and structural design.

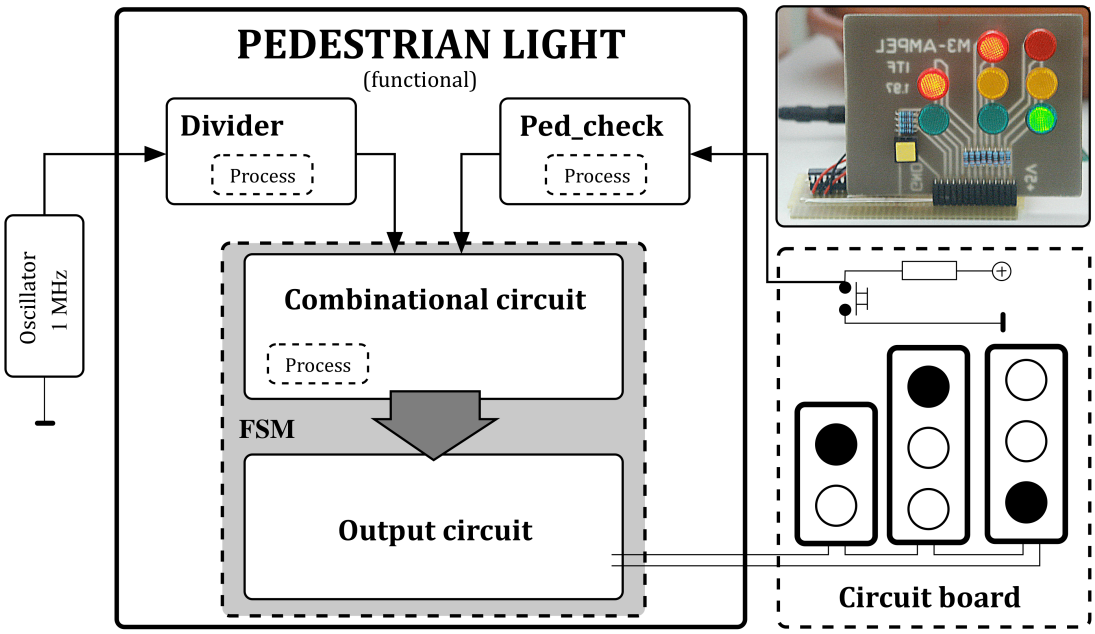

Figure 14: Block diagram of a traffic light circuit with pedestrian crossing - realized as an FSM.

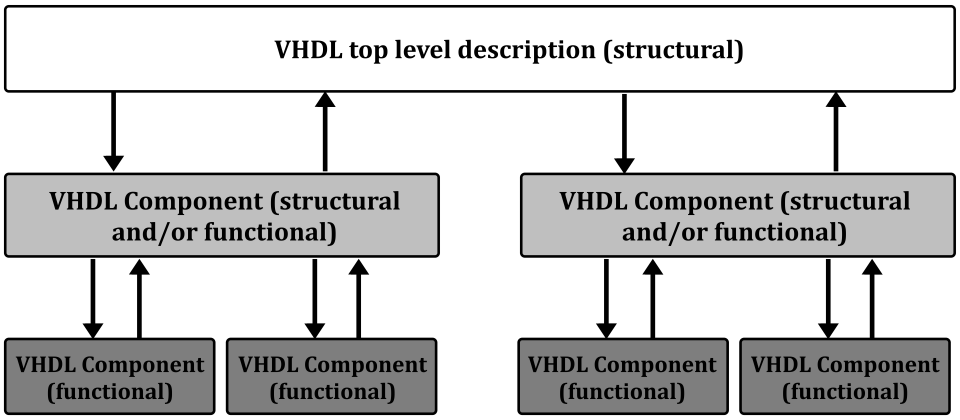

Figure 15: Hierarchy of a VHDL project.

Components are declared in the declaration part of the architecture with their complete entity and then instantiated in the instruction part. With a PORT MAP the signals of the components are linked to each other and also to the entity of the design.

If a component is to be instantiated several times, such as a series of FlipFlops for a counter, this work can be automated with the GENERATE instruction. Within a loop, the desired number of components is generated, whereby inputs and outputs can be linked to each other via the port maps. In many cases, however, a functional description is simpler and more space-saving than the numerous instantiation of components.

The use of both modeling techniques (functional and structural) allows the tasks of a circuit to be developed and tested in separate work processes or in different teams. The result is a kind of modular principle with which complex circuits can be assembled from simpler modules.

5 Experimental setup

This experiment is intended to provide an overview of the steps necessary for the development of modern operating and readout electronics for an X-ray/gamma detector.

Starting from the analog detector signal of a photomultiplier, its output is first digitized using an ADC (analog-to-digital converter). The control of this device as well as the digital processing of the signals will be realized in an FPGA[16] with the help of the hardware description language VHDL. This task includes designing a "sequencer" to operate the ADC, analyzing the waveform (determining the rise time and the maximum value of the detector signal) necessary to suppress background events, and then formatting and transmitting the data as "event packets" to a data acquisition system. You will get to know the typical elements of a "hardware synthesis": Sketch of the necessary components and modules, programming of the individual control and analysis processes (ISE Webpack), simulation of the design on the computer with ModelSim or iSim, synthesis process for a XILINX Spartan-3 FPGA and commissioning of the hardware as interface between detector and data acquisition system (see Figures 16 and 17).

The measurement and control of the analog detector signals as well as the testing of the characteristics of the developed digital electronics is carried out with a "mixed signal" (analog + digital) oscilloscope. Finally, the radiation background and the properties of various radioactive sources are measured with the resulting experimental setup. The data obtained in this way is analyzed with the aid of software; you will determine the properties of the detector (energy/time resolution) and of the radioactive sources used.

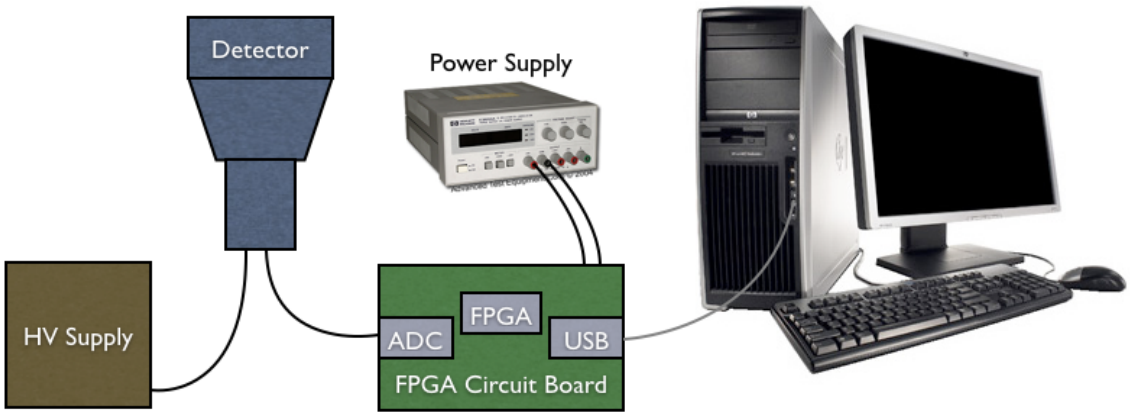

Figure 16: The individual components of the experimental setup.

5.1 Commissioning of the detector

Set the phoswich detector into operation (by connecting it to the high voltage source) and take a look at the output signal on the oscilloscope.

Task 8: Measure the rise times and the maximum amplitude in NaI and CsI when irradiating the detector with ^{241}Am.

Task 9: Determine the field of view of the collimator on the basis of its dimensions.

[16]Field Programmable Gate Array.

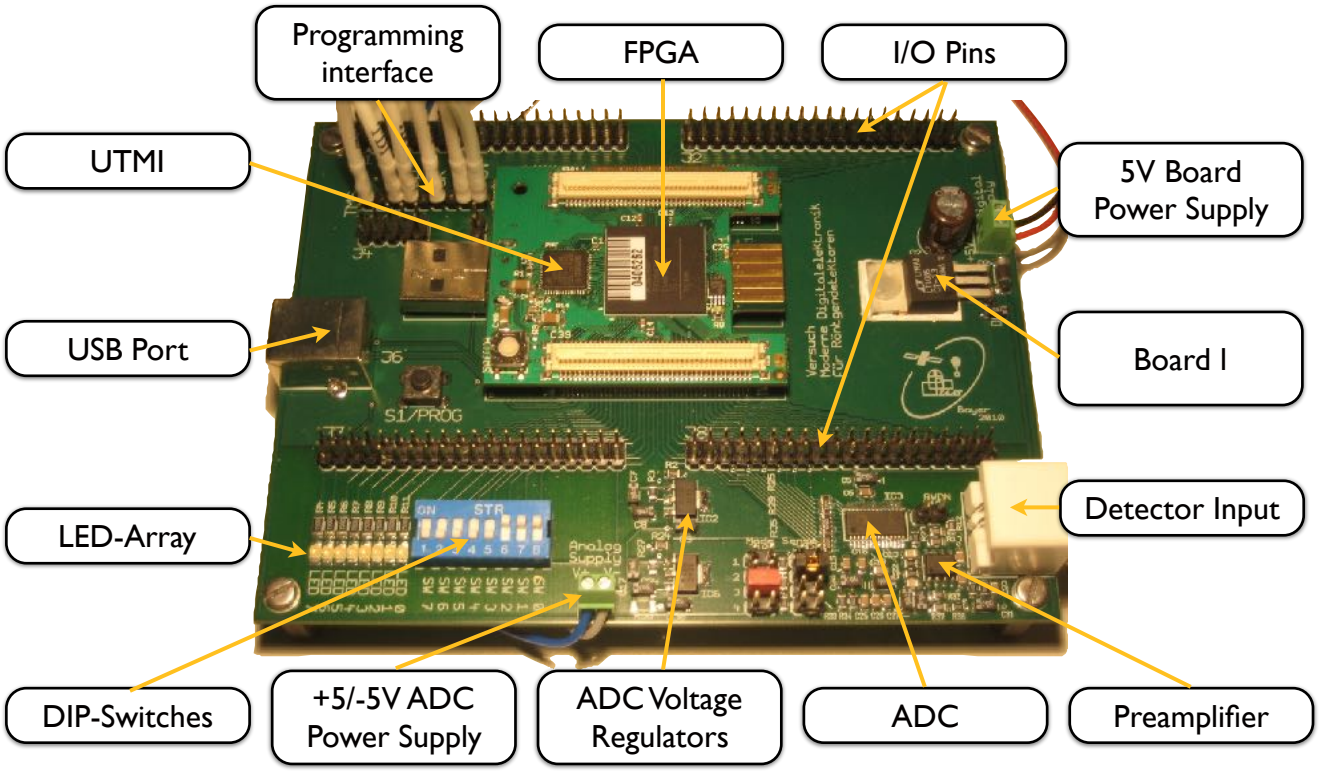

Figure 17: The circuit board shown is the heart of the experiment. It also carries the FPGA board and the ADC and acts as the interface between the detector and the data acquisition system.

5.2 VHDL programming

In the VHDL part of the experiment you will write and test some designs. The first ones should give you the opportunity to use simple VHDL constructs for the first time and to understand their function better. In the central design you will integrate the component "usb_serial" to send and view the ADC data directly to the computer. The last part is the actual objective: It is your task to write a VHDL process that processes the ADC values already in the FPGA, so that only the amplitude and the rise time of a γ event have to be transferred to the computer via the USB. Then you will analyze the properties of the phoswich detector with the resulting data acquisition system.

Task 10: In your first line of code, write the instruction that the LED0 should be permanently on. Add a comment to the line above, e.g. - - *Task 8, Lamp permanently on.*

Task 11: Write in your next line of code the instruction that the LED1 should assume the state of SWITCH0 (i.e.: if the switch is set to '1', the lamp should also be on; if the switch is set to '0', the lamp should be off). Try to stay as simple as possible - can you possibly do without the IF statement?

Task 12: Add to your design so that you make the state of LED2 dependent on the switches SW1 to SW4. It makes sense to implement your design from Task 6. Can you solve the task with a single assignment?

Now add an external clock signal to your design to realize time-controlled behavior: The FPGA in the practical course experiment has access to a configurable clock line that provides the 60 MHz clock frequency if you follow the appropriate steps:

1) Expand your entity with the following signal:

```
CLK:                    in STD_LOGIC;
```

2) Insert the corresponding pin for the CLK signal into the UCF constraint-file:

```
NET "CLK"               LOC = D9;
```

Now you can use the clock signal in the next task!

Task 13: Switch LED3 on and off with a frequency of 1 Hz. To do this, write your first process 'frequency divider' in VHDL, which is best implemented with a counter that counts rising edges of the CLK signal and is reset to zero at a certain value. Then switch the LED on or off depending on the counter reading.

Task 14: Now read in the values digitized by the ADC. The ADC (just like the LEDs) must be clocked with a *std_logic* signal to digitize the voltages on each rising edge. Note that you clock the ADC at 30 MHz, which is the fastest frequency the ADC can operate at. Simply copy the process of the previous task and adjust the values and the signal names. For test purposes, output the lower bits of the data read in on the LEDs that are still free.

Task 15: You can use a pre-built component from the Internet to transfer the data to your computer. This is done very often in practice. You can find it in the "templates" directory. Add the files to the project as a copy using the "add source" button. You must also consider how to send the 10 bit of an ADC value via the USB interface, which only allows 8 bits at a time. If you follow the instructions below, you can use the unpacking programs that have already been written: These expect the top two bits to indicate which byte is currently being transmitted, "00" stands for the low byte and "0100" for the high byte.

Task 16: Write a component that continuously checks the ADC values and, in case of a γ event in the detector, determines the amplitude and rise time and sends it to the computer. Just like the amplitude, the rise time is to be transmitted as 10 bits (or 11 bits) unsigned. The program on the computer now always expects 4 bytes, with the first two bytes "00" and "0100" being the maximum amplitude, the two following bytes, marked with "10" and "1100" are the upper bits of the rise time. To determine a threshold criterion for an event (trigger level), use the results from Task 14.

5.3 Measurement of detector properties

Now that the data acquisition and transmission to the PC is working, you can perform some measurements on various detector properties. First, you should optimize the data acquisition on the PC so that only the pulses from the NaI scintillator are registered and the CsI pulses are suppressed as far as possible. This selection is made by defining a valid time window for the rise times.

> **Task 17:** Create histograms of the rise times when the detector is irradiated with the source from the front (NaI) and then from above (CsI). In this way you can identify the signals from the NaI. Determine the position of the maxima in the plots and convert them into times. Compare the results with those from Task 8 and set a lower and upper limit for the time window in the NaI.

The next step is to calibrate the energy scale. For this it is necessary to convert the ADC channels into photon energies. Afterwards the relative energy resolution $\Delta E/E$ of the detector should be determined.

> **Task 18:** Create a histogram of the maximum amplitudes when the detector is irradiated with the source from the front in the center at a distance of a few centimeters. Use the two peaks in the spectrum to determine the conversion to the energy scale and specify the relative energy resolution. Perform the same steps for irradiating the detector from the front in a corner. How will the energy resolution change? Why?

> **Task 19:** Determine the background count rate and record a spectrum of the background in the laboratory.

Now complete the structure of the gamma telescope with a tube collimator.

> **Task 20:** By what factor is the background reduced by the tube collimator?

> **Task 21:** By measuring the count rate of the detector as a function of the rotary table angle, you have another way to check the collimator's field of view. First, point the collimator exactly at the source and then turn it 4° to the left. From there, at every half degree, measure the number of events in 30 seconds until you reach 4° on the right and evaluate the curve when preparing the protocol. Compare the result with that of the geometric measurement of the collimator in Task 9.

6 Notes on the preparation of the protocol

In this last part of the guide we give a few tips for the preparation of the protocols. Although some of them may seem obvious and self-evident, we ask you to look at them again when writing and above all to heed the points in the last paragraph.

The protocol should begin with an introduction summarizing the contents and objectives of the experiment. *Always use your own words. Don't copy from anywhere else. It is considered fraud to copy text from someplace else for your protocol.* After this, a short theoretical part should be inserted in which the astrophysical connection and the basics of the detector principle are outlined. Here it is recommended to go along the tasks 1 - 7 and answer them. Then add a part of the experimental procedure which describes the steps performed in the experiment sequentially - each with a few sentences, what to do in this part and why, how the measurement/step was performed, what came out of it and what can be concluded and learned from it. Finally, a section "Conclusion" or "Summary" would be appropriate, which reiterates the most important results in relation to the objectives described above.

Before submitting the protocol, an electronic spell check should be carried out. This is presently available on all computer platforms. After that, ALL members of the practical course group should read the protocol and agree to the submission. This is a process that scientists should make a habit of. Before a document with your name on it leaves your desk or is placed on the Internet, you should make sure that it is not only qualitatively correct in terms of content but also in terms of external impressions - especially if you are not the author yourself!

A Description of the command line programs

The command line programs are described here in the order in which they are needed in the experiment. Programs starting with "`./`" are located in the directory "`/home/student/usb-utmi-cdc-acm/-programs`". With the command "`make`" all can be re-compiled if necessary. Not all programs in this directory are needed for the experiment.

First, a brief overview. With *./rdwr* the data is received from the hardware via USB or sent to the hardware. They must then be unpacked with a conversion program and shaped with a print program that can use the following tools.

./rdwr: This program reads the data from the USB and writes it unchanged to *stdout*. It waits for *stdin* to send data to the USB peripheral device.

pv: With *pv*, the amount of data transferred during operation can be displayed.

./vra_convert: This takes the data from *./rdwr* and unpacks it as it has been packed in the FPGA. It always expects two bytes per integer, whereby the first two bits are used as markers for whether it is the high byte or the low byte.

./vra_print: Prints the data from *./vra_convert* in ASCII, one integer per line.

./vrdbl_convert: Similar to *./vra_convert*, it unpacks the data from the FPGA. It always expects four bytes – two for the first integer, two for the second – each marked by the highest 2 bits.

./vrdbl_print: Similar to *./vra_print*, it prints the data of *./vrdbl_convert* in two integers per line.

./bin.py: Determines the bin to which a line belongs.

./copycol.py: Copies a column.

grep: The General Regular Expression Parser is used to select lines.

cut: Selects one or more columns of the input stream.

./histogram.py: Counts how often each input line occurs.

sort -g: Sorts the lines of the input stream by a general numeric sort.

./prepend_time.py: Inserts a time stamp at the beginning of each line.

tee <**filename**>**:** Writes the input to the file "`<filename>`" and to *stdout*.

gnuplot: A powerful plot program.

The tools can be called together in a UNIX pipe and their output can be redirected to a file, for example

```
$ ./rdwr | pv | ./vra_convert | ./vra_print > data.txt
```

B Pinout

Pin Board	Pin FPGA	Task
J8_03	K4	ADC_CLK
J8_05	K3	ADC_D0
J8_08	J1	ADC_D1
J8_10	J2	ADC_D2
J8_09	M4	ADC_D3
J8_12	K2	ADC_D4
J8_11	P2	ADC_D5
J8_14	K1	ADC_D6
J8_13	R1	ADC_D7
J8_16	L3	ADC_D8
J8_18	L2	ADC_D9
J8_01	K5	ADC_OR
J7_18	E3	LED_0
J7_16	D1	LED_1
J7_14	D2	LED_2
J7_12	D3	LED_3
J7_08	C1	LED_4
J7_06	C2	LED_5
J7_04	B1	LED_6
J7_02	C3	LED_7
J7_20	B3	S1/PROG
J7_29	F5	SWITCH_0
J7_30	G4	SWITCH_1
J7_27	F4	SWITCH_2
J7_28	F2	SWITCH_3
J7_25	E4	SWITCH_4
J7_26	F3	SWITCH_5
J7_24	E1	SWITCH_6
J7_22	E2	SWITCH_7
-	D9	CLK

Pin FPGA	Task	Dir FPGA	Default
B14	DATABUS16_8	OUT	0
B4	RESET	OUT	0
D6	XCVRSELECT	OUT	1
B6	TERMSELECT	OUT	1
T10	OPMODE<0>	OUT	0
C6	OPMODE<1>	OUT	0
B12	TXVALID	OUT	0
A3	LINESTATE<0>	IN	
A5	LINESTATE<1>	IN	
A14	TXREADY	IN	
B13	VALIDH	BIDIR	
A12	RXVALID	IN	
C11	RXACTIVE	IN	
C12	RXERROR	IN	
A13	D0	BIDIR	
B11	D1	BIDIR	
C10	D2	BIDIR	
B10	D3	BIDIR	
A10	D4	BIDIR	
C9	D5	BIDIR	
A9	D6	BIDIR	
B8	D7	BIDIR	
A8	D8	BIDIR	
C8	D9	BIDIR	
C7	D10	BIDIR	
A7	D11	BIDIR	
B7	D12	BIDIR	
B5	D13	BIDIR	
C5	D14	BIDIR	
A4	D15	BIDIR	

Literature

Birks, J. B. (1964). *The Theory and Practice of Scintillation Counting*. Pergamon, Oxford.

Grupen, C. (1993). *Teilchendetektoren*. BI Wissenschaftsverlag, Mannheim, Leipzig.

Joram, C. (2001). Particle detectors. Available under the URL
http://joram.web.cern.ch/Joram/lectures.htm.

Knoll, G. F. (1999). *Radiation detection and measurement*. Wiley, New York, 3rd edition.

Reichard, J., Schwarz, B., *VHDL Synthese*.

Lehmann, G., *Schaltungsdesign mit VHDL*.