UNIVERSITÀ DEGLI STUDI DI PARMA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

PARMA (I) - VIALE DELLE SCIENZE

TEL. 0521-905800 • FAX 0521-905758

Dottorato di Ricerca in Tecnologie dell'Informazione
XIX Ciclo

Giulia Papotti

ARCHITECTURAL STUDIES OF A RADIATION-HARD TRANSCEIVER ASIC IN 0.13 um CMOS FOR DIGITAL OPTICAL LINKS IN HIGH ENERGY PHYSICS APPLICATIONS

DISSERTAZIONE PRESENTATA PER IL CONSEGUIMENTO
DEL TITOLO DI DOTTORE DI RICERCA

GENNAIO 2007

CERN-THESIS-2007-111
//2007

εστω δε ο λογος υμων ναι ναι ου ου
το δε περισσον τουτων εκ του πονηρου εστιν

(Simply let your 'Yes' be 'Yes,' and your 'No,' 'No';
anything beyond this comes from the evil one.)

(Matthew, 5:37)

Acknowledgements

I have always been concise, so all you get is a list. Here it goes…

Paulo and Sandro for useful comments, critiques, discussions, explanations, feedback, questions, support… Apart from helping out in the completion of the PhD thesis in particular and PhD program in general, they have been role models who helped me in my personal professional growth.

At my University in Parma, Prof. Ciampolini for being there every time I needed him, despite the amount of other engagements. I will not forget that he was the one who suggested CERN to me in the first place.

Mike and all the Microelectronics group at CERN, for being such a stimulating working environment. In particular Bert, Ernest, Federico, Francois, Giovanni e Giovanni, Guido, Hugo, Kostas, Matthieu, and Rafael. Winnie needs to be specially thanked among them for reading and patiently correcting most of this thesis.

Francois, Jan and Karl cannot be forgotten either, for having initiated me to CERN and the world of experimental measurements. They were not thanked in my previous thesis, so here they are.

My family: Mamma and Ermanno, to whom this thesis is dedicated. And Alma, Fabio, Giampietro, Sandro, Silvana e Tina. I owe you the fact that I am what I am.

Anna, Marci, Vale, and Zsach, the ones I can always rely on.

Michele, Tommaso, Nuno for having been so important in my life.

Ale, Andrea, Dora, Elena, Enzo, Giovanni, Giulio, Michele, and Marcello. And in particular Ale, Gio, Marco e Matteo for their crucial smiles (and dinners) the days of the final rush. It would have been much harder without your friendship.

All the flatmates with whom I shared life in Geneva: Sarah and Ines, Margherita and Katrin, Cesar, Daniel, Pedro and Rui, Gianluca, Bernd and Nina.

Bubba, Carlo, Donio, Elisa, Ferro, Gianpi, Grego, Leo, Marco, Paolo, Siro, and Tom, for the good old days at university, and the bond that survived to the present.

Angela, Annemarie and Peder, the closest in Geneva. And also Agaath, Alice, Catalin, Cristiana, Matteo, Olivia, Osamu, Phu, Sabrina and Tom.

Many of the Portuguese, among which Frodo, Helder, Martins, Miguel, Parracho, Ricardo, Tony and Ze (my dealer).

Marco for sharing one of my deepest passions and buying the tickets, and for the time spent in Verona and Milano.

A special thought for Dimitri, from Minsk.

Patrice, for interesting updates on LHC and CMS, but also for often not being there, leaving me a whole office for myself.

Klaas for his “self-esteem boost” that came just at the right time.

Rick for having invented 64b/66b and for having replied to my email.

Google and Wikipedia, for making things so much faster, and Lindt for the sugar supply and endorphin releases in tape out times.

Finally, all the ones I met along the way, even though their names might be forgotten in these acknowledgements.

To all of you, grazie.

Table of Contents

Chapter 1

Introduction

Data transmission in future High Energy Physics (HEP) experiments will be crucially affected by high radiation levels in the detectors so that careful error handling techniques are required to protect the data and thus assure reliable transmission. Optical links are often chosen in HEP as the data transmission means due to the many advantages they have over electrical transmission, such as galvanic isolation between the two link ends, high bandwidth, low electromagnetic interference, low mass and low weight. While optical components have to be bought off-the-shelf and thoroughly tested for radiation effects, microelectronic components can be designed by making use of radiation-hardening techniques which guarantee them to last over the lifetime of the experiments and for the expected radiation levels. Line coding and Forward Error Correction (FEC) guarantee a channel-optimized line data stream and error handling, and are implemented in the electronic integrated circuits with a small increase in complexity.

The work presented in this thesis deals with the integration of line coding and FEC techniques for optical links to be used in radiation environments. It was carried out in the Microelectronics Group of CERN, the European Laboratory for Particle Physics.

A short introduction about CERN follows, after which optical links for HEP data transmission are introduced. Subsequently, the issues addressed by line coding and FEC are presented together with an overview of the solution proposed for the next generation high speed links at CERN. This introductory chapter ends with a presentation of the thesis structure.

1.1 LHC, the experiments and the upgrade

CERN is one of the world's largest scientific laboratories, and its aim is to study the basic constituents of matter and their interactions. The "probes" for sub-atomic observations (10^{-10}~10^{-16} m and beyond) are particle beams which are produced and subsequently accelerated to high energies in particle accelerators. Once the beam is accelerated to the desired energy, it can be used in an experiment: an experiment consists of colliding particles either onto a fixed target or with another particle beam, and observing and studying the outcomes of the collisions using particle detectors that surround the interaction point.

The Large Hadron Collider (LHC) is the world's next high energy particle accelerator, due to start at CERN in 2007. It is a 27 km-long proton-to-proton beam circular collider, with beam energies of 7+7 TeV and a design luminosity of 10^{34} $cm^{-2}s^{-1}$ for protons (where the luminosity is a measure of the density of particles in the beams).

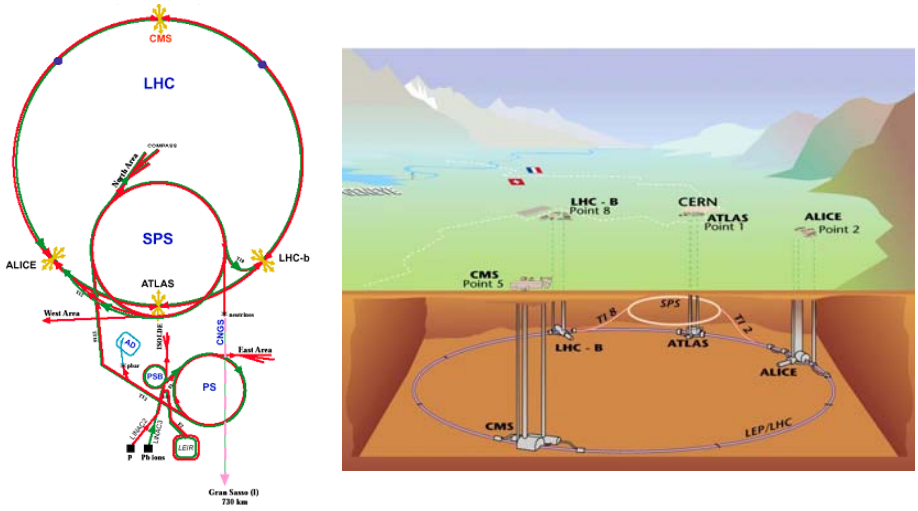

Figure 1: LHC ring, pre-accelerator stages (not to scale) and main experiments. Plan (left) and assonometric view (right).

The main experiments at the LHC are ATLAS, CMS, ALICE and LHCb. Their positions in the LHC ring, together with the main pre-accelerator stages, are depicted in . While ALICE and LHCb are aimed to study heavy ion collisions and bottom quark physics respectively, ATLAS and CMS are "general purpose" experiments, designed to be able to handle different

possible physics scenarios. The four LHC experiments will probably give evidence to new particles: in particular physicists around the world are impatiently waiting for the possible confirmation of the existence of the Higgs boson and of supersymmetric particles. The latter might be an explanation to the issues of dark matter and dark energy. The former is theorized as the explanation for the mechanism of electroweak symmetry breaking, and as a result gives masses to matter particles.

While the LHC is just about to start, the community has already started thinking of its luminosity upgrade. The motivation for this lies in the fact that the statistical error in a measurement decreases proportionally to the square root of the number of measurements, and so, in order to halve the error, four times as many measurements have to be performed. After 4~5 years of operation of the LHC at full luminosity, then, an upgrade of the machine is needed, and is planned, for improving the precision of the measurement within a reasonable amount of time. The upgraded machine is called SLHC, or super-LHC. Regardless of the upgrade physics details, the result is very likely to be an increase in luminosity of a factor of 10. This will impact the detectors, which will have to be upgraded as well.

1.2 Optical links for HEP

LHC experiments like ATLAS and CMS will generate high volumes of data (e.g. 18 Pbyte/year for the ATLAS detector). Even higher bandwidth will be required for the SLHC experiments, where an increase of a factor of 10 in the number of events is expected. Optical links seem to be the natural solution for data readout due especially to the high inherent bandwidth, but also due to other properties such as galvanic isolation, low electromagnetic interference, low atomic number of the materials involved, and low cabling weight.

The detectors at the LHC will be subject to harsh radiation levels due to the high number of colliding particles and the high frequency of occurrence of collisions. The optical links components which will sit inside or in the proximity of the detectors have to be radiation hard. Radiation though, is a concern for very few applications, like HEP, space missions and weaponry, and thus radiation-hard components cannot be easily bought off-the-shelf. The link's optical components are often chosen among commercially available components to minimize the customization required, and thus need to be thoroughly tested and qualified for understanding and overcoming

radiation effects. On the contrary, ASIC components are designed by making use of a whole set of layout, circuit and architectural solutions in order to guarantee functionality and reliability over the 10-years expected lifetime of the LHC experiments.

In the framework of future luminosity improvements of the LHC, a new optical transmission system is being developed in which the link is bidirectional and adaptable to different link configurations and functionalities. This new link is named Versatile Bi-Directional (VBD) link, while GigaBit Transceiver (GBT) is the name chosen for its transceiver ASIC. The VBD link upgrades previously designed systems, such as the Timing, Trigger and Control system (TTC, [Tay02]), and components, such as the Gigabit Optical Link ASIC (GOL, [Mor01]). The new link profits from a recent technology for GBT implementation and this allows an important speed improvement compared to the previous systems. While in the present TTC system two bits per 25 ns are delivered, more than 60 bits are available in the new version (corresponding to a data bandwidth increase from 80 Mb/s to ~2.4 Gb/s), allowing many improvements to the functionality of the system. Moreover, the new system will be subject to higher error rates compared to the old systems due to exposure to higher levels of radiation in SLHC [DeR06] and due to the higher speed of the link.

1.3 FEC and line coding for HEP data transmission

As optical links in HEP experiments are placed in a radiation environment, single event upsets on the photodiode are likely to be the main source of data transmission errors. In short, photodiodes respond to radiation particles in the same fashion as they respond to light emanating from the optical fibre. If the particle deposits enough energy, a false signal is detected which corresponds to a link error.

At the same time, line code is often required to map the digital information to be transmitted over a link to a signal that is optimally tuned for the specific properties of the physical channel and of the receiving equipment. In case of optical serial links, for example, a line code is required to provide a relatively high number of transitions on the serial bit stream in order to facilitate clock and data recovery and in particular to achieve low jitter. Moreover, the serial data stream has to be constituted of roughly the same number of zeros and ones in order to allow for ac-coupling in the receiver. Examples of line codes which are used in commercial

applications are 8b/10b (used in the Gigabit Ethernet Standard), 64b/66b (used in the 10 Gigabit Ethernet Standard), CIMT (used in Hewlett-Packard G-Link).

A line code for the VBD link thus needs to include an error correction scheme particularly targeted to the issue of SEUs on the photodiode. Commercial standards were studied for use in the GBT ASIC, but were found to lack error correction capability combined with high code efficiency and low latency as required by the VBD link.

A new scheme is thus devised in this work which combines traditional line coding properties with low latency error correction capability. The proposed code uses the concatenation of a scrambler for data randomization, a Reed-Solomon error correcting encoder/decoder for addressing errors on the photodiode and the addition of an error-tolerant header for frame synchronization. This thesis reports on the development of such a scheme, which is proposed in two options which achieve slightly different error correction capabilities.

A demonstrator ASIC for one of the two options was implemented in a 0.13μm CMOS technology and tested, so that implementation details and test results are also discussed in this thesis.

1.4 Thesis structure and contributions

A description of the contents of the thesis, chapter by chapter, follows.

- Chapter 2, Optical Links for radiation environments: In this introductory chapter the main blocks which constitute and optical link are introduced in light of the need for line coding, which is also defined and introduced. Radiation effects on the link components are also introduced.
- Chapter 3, The GBT ASIC and the Versatile Bi-Directional Link: The VBD link and the GBT ASIC are presented via a discussion on main blocks and functionalities. A study of the characteristics of radiation induced errors on the photodiode is also presented. The combination of existing line coding and error correction is studied, effectively motivating the need for the work carried out in this thesis.
- Chapter 4, Error-correction theory introduction: The basic concepts of FEC are presented here as a basis for understanding Reed-Solomon codes. Reed-Solomon encoding and decoding are presented. The technique of interleaving is also introduced here.

- Chapter 5, GBT line encoding scheme: The proposed line coding scheme is presented: a scrambler is used for its randomization properties, RS codes are used for error correction capability and a header is used for frame alignment. Two different coding options are presented.
- Chapter 6, Code performance and hardware requirements: The proposed code properties, such as error correction capability, DC-wander performance and run-length characteristic, are calculated. Encoding and decoding block complexity is also presented.
- Chapter 7, A demonstrator ASIC for the GBT line code: The demonstrator ASIC implemented in a 0.13 um technology is described, including its functionality, design procedure, test set-up and results.
- Chapter 8, Conclusions and future developments.

The contributions made during the course of this research have led to the following publications:

- G. Papotti, "An Error-Correcting Line Coding ASIC for a HEP Rad-Hard Multi-GigaBit Optical Link", Proc. 2nd Conference on Ph.D. Research in Microelectronics and Electronics (PRIME 2006), Otranto (Lecce), Italy, 12-15 June 2006, pp.225-8.
- G. Papotti, A. Marchioro, P. Moreira, "An Error-Correcting Line Code for a HEP Rad-Hard Multi-GigaBit Optical Link", to be published in Proc. 12th Workshop on Electronics for LHC and Future Experiments, Valencia, Spain, 25-29 September 2006.

Chapter 2

Optical Links for radiation environments

This chapter introduces optical links to be used in radiation environments, explaining the reasons why line coding and Error Correction (EC) are required on the data to be transmitted. First, in section 2.1.1, a motivation for the choice of optical links in a HEP environment is given, followed in section 2.1.2 by a short presentation on the link's building blocks. The need for line coding is introduced in section 2.1.3, where examples of solutions adopted in commercial links are as well introduced. Finally, in section 2.2 the effects of radiation on the different components of the link are discussed, explaining why single event upsets on the photodiode are the main issue to be addressed, thus requiring an EC scheme. The combination of line coding and EC techniques is the main contribution of this thesis, and the proposed solution is discussed in chapter 5.

2.1 Considerations about optical links

2.1.1 The choice of optical links for data transmission in HEP

The use of optical fibers as a transmission medium has two main advantages which drove its use in the telecommunication industry and made it such a widespread choice, especially for long haul communications: low losses and high bandwidth [Agr05]. Losses can in fact be as low as ~0.2 dB/km depending on the materials that are used, allowing long distances to be covered without the insertion of repeaters, making the system cheaper and easier to implement. High bandwidth is also an intrinsic property of the system: given that the bandwidth of the modulation can be

up to a few percent of the carrier frequency and with the optical carrier frequency at typically ~200 THz, optical communication systems have the potential of carrying information at bit rates of ~1 Tb/s.

The utilization of optical links has become more popular with the advancement of research and reduction of prices, making new technologies both available and affordable. Optical links are often preferable over copper cables as they have the potential for high bandwidth and low power losses, and also exhibit other characteristics which are important in a HEP environment. Many physical constraints come from the fact that parts of the link are placed inside a particle physics detector: the environment is subject to radiation, often only little space is available, and it is desirable to have minimum interaction with the particles to be detected.

The small size of optical fibers is thus valuable because of space constraints that are often a concern in the tightly packed detectors of the experiments and because cable volume creates “blind spots” in the same detectors. Additionally, optical systems are immune to Electro-Magnetic Interference (EMI) as the signal is constituted of light, and the front end and back end electronics are electrically isolated, which are big advantages in such dense and large systems where power and ground distribution is a major issue. Moreover, an optical fiber is constituted of low atomic number materials (silicon and oxygen in the glass, carbon, hydrogen and oxygen in the plastic coatings) leading it to have less interaction with the incoming particles than for example copper cables, so that the shadowing effect towards sensitive parts of outer detectors is lower.

2.1.2 System components

Any communication link can roughly be divided into three main parts, as sketched in Figure 2: a transmitter, a receiver and communication channel which connects them. The role of the communication channel is to transport the signal from transmitter to receiver with the least possible distortion. The transmitter transforms the information to be transferred into a signal which is suitable for the communication media. After having traveled through the channel, the signal arrives to the other side of the link, where the receiver converts back the information. Optical links are characterized by the use of optical fibers as the communication channel.

Optical fibers ([Sac05]) support and guide the transmission of the optical beam through a confinement of light based on the principle of total

reflection (see Figure 3). Fibers are in fact based on a two-layer cylindrical structure in which the inner layer, called the core, has a refractive index that is slightly smaller (1%) than the outer layer, the cladding. The difference in the refractive index and the propagation angle prevent the light from leaking out from the core/cladding interface when launched into the core. Other layers are added externally to the cladding for mechanical protection.

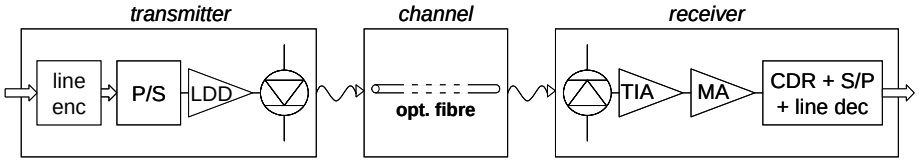

Figure 2: Main building blocks of an optical link. In the transmitter block, line encoding, parallel-to-serial conversion (P/S), laser diode driving (LDD) and electrical-optical conversion (by the laser) are performed. The optical fibre constitutes the communication channel. In the receiver the photodiode performs optical-electrical conversion, the signal is then amplified by the transimpedance amplifier (TIA) and the main amplifier (MA); finally clock and data recovery (CDR), serial-to-parallel conversion (S/P) and line decoding are performed.

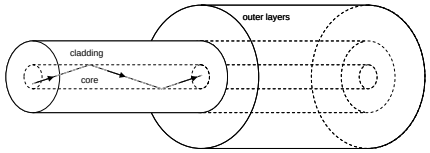

Figure 3: Optical fibre section (a) and light confinement principle operation (b).

The transmitter function is to convert the electrical signal at its input into an optical signal matched to the chosen medium conveying the same information as the original electrical signal. The transmitter ASIC has many different functions, including serialization (P/S, in order to allow the user to deal with slower signals) and optimization of the signal stream for the transmission medium (line coding).

The signal serial stream is often applied to the driving circuit of the laser diode (Laser Diode Driver, LDD), so that the laser output is directly modulated by the signal. Within the LDD, an Automatic Power Control (APC) mechanism is often implemented to stabilize the output power. This mechanism, though, suppresses the low frequency components of the transmitted signal and this can produce baseline drift when transmitting long strings of ones or zeros. This unwanted drift can be avoided by using DC-balanced data signals.

Semiconductor diode lasers ("Light Amplification by Stimulated Emission of Radiation") are statically described by the relationship between the optical output power and the drive current, the P/I curve, reported in Figure 4. Below the threshold current (I_{th}), the light output is small and incoherent. For currents higher than threshold the light output increases rapidly and is proportional to the drive current (the proportionality constant is called laser *efficiency*). The drive current values I_0 and I_1 correspond to the light outputs P_0 and P_1 (with $P_0 \sim 0$).

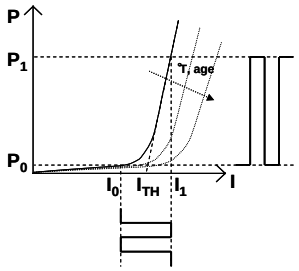

Figure 4: Laser diode characteristic: light output power plotted against drive current. The threshold current increases and the slope efficiency decreases with an increase in temperature [Agr02]: it is thus necessary to control the laser temperature through a built-in thermoelectric cooler in order to guarantee a constant power output.

The receiver block functionality is complementary to that of the transmitter, as it converts back the information from a serial optical signal to a parallel electrical one.

The optical signal is first converted to electrical by a photodetector, often a photodiode, which responds to the incident optical power with a proportional photocurrent (the proportionality constant is called *responsivity*). Two main types of noise are generated in the p-i-n photodiode along with the photocurrent: shot noise and dark current noise [Sac05].

As the incident optical power is of the order of microwatts, and the responsivity is typically in the order of 1 A/W, amplifying stages are needed directly after the photodiode in order to obtain electrical levels suitable for the CDR circuitry. A cascade of two amplifying stages is commonly used[Sac05]: a TransImpedance Amplifier (TIA) and a Main Amplifier (MA, an automatic gain control or limiting amplifier). The TIA needs to be designed carefully as its noise dominates all other noise sources in the receiver. The MA amplifies the small signal from the TIA to a level that is sufficient for the reliable operation of the clock and data recovery circuit. A

low-frequency cutoff in the receiver response is often observed due to AC coupling between the TIA and the MA, or due to offset-compensation circuits in the MA. The low-frequency cutoff results in a baseline wander when long strings of zeros or ones are received, and the baseline shift makes detection difficult and degrades the system noise margin.

The amplified voltage signal is then processed by the Clock and Data Recovery (CDR) circuitry, deserialized (S/P) and line decoded so that it is presented at the receiving end of the link in the same form as it had been transmitted.

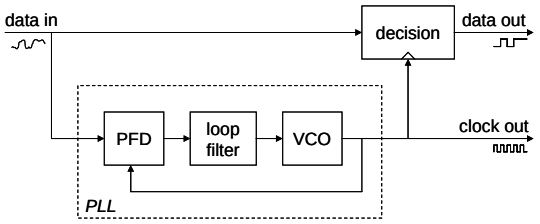

Figure 5: Clock and Data Recovery simplified circuit.

The Clock and Data Recovery circuitry (CDR, see Figure 5) extracts the clock frequency, that is the spectral component at $f = 1/T_{bit}$, and demodulates the digital data. A local oscillator (often a Voltage Controlled Oscillator, VCO) locally recreates the transmitter clock with the highest possible precision concerning phase and frequency in order to synchronize the data detection process. The VCO is regulated by the information received from the phase/frequency detector (PFD) block which is sensitive to phase/frequency differences between the local oscillator and the incoming data stream. In order for the PFD output to be meaningful, though, it is necessary for the transmitted encoded serial bit stream to have enough embedded clock information. For example in the case of Non-Return-to-Zero (NRZ) data format, the phase detector can perform a comparison between data and local clock phase only if data transitions (as from zero to one or from one to zero) are present.

2.1.3 Line encoding

Line coding is a form of coding used to match the data stream to the characteristics of the channel. Consequently, the line code facilitates proper signal reception by being tailored to the link under study [Bro83]. The data input is encoded in the transmitter through a *line code* which guarantees

vital properties on the coder output stream, that is the line data stream, for any possible data input. The code verifies the property of being "transparent", i.e. it does not impose any restrictions on the content of the transmitted message [Byl80].

A list of the required line code features in the case of fiber systems employing optical intensity modulation follows [Fai91], [Sac05]:

- Bit sequence independence: the line code must adequately encode any source bit sequence.
- Small low frequency content: the transmitted signal should be balanced (ideally contain an equal number of ones and zeros) to allow for AC coupling in the receiver and simplify laser bias circuitry.
- Transmission of adequate timing information: the encoded bit stream must produce a high density of level transitions to allow for proper operation of the clock recovery circuitry.
- Small number of equal consecutive bits: this reduces the low frequency content of the transmitted signal and limits the associated baseline wander when AC coupling is used.
- High efficiency: introduced redundancy should be kept to a minimum in order to keep noise bandwidth and rate of terminal circuitry as low as possible.
- Low error multiplication: an error which occurs during transmission should not result in many decoded errors.
- Low systematic jitter: the sequence of transmitted bits must ensure that pattern dependent jitter is kept low.
- In the case of a framed data stream, some method for guaranteeing frame alignment is required, so that the boundaries of a frame can be located at the receiver.

A few parameters are defined in order to evaluate the code performance by accessing the quality of the line data stream and for comparing different codes according to the properties previously enumerated. The *maximum run-length* is defined as the maximum number of consecutive binary "1"s or "0"s that can be found in the coded stream [Wid83]. The *average run-length* is the average length of strings made of the same symbols.

The *disparity* is the difference between the number of "1"s and "0"s in a block, while the *running disparity* is the same difference calculated on the whole sequence of transmitted bits (also *running digital sum*). The *digital sum variation* is the peak-to-peak difference between minimum and maximum running digital sum. When the digital sum variation is bounded, the code is free of any DC component [Wid83].

Moreover, the *redundancy r* of a code is the number of bits used to transmit a message *n* minus *k* bits of actual information in the message (n = k + r). Complementary to the code redundancy is another parameter, the code *efficiency*, defined as the ratio between the rate of information of the code and the line rate, or, especially for block codes, it is the number of information bits divided by the number of bits required to encode them. Both measure the bandwidth increase required by the use of a code.

In practice line coding is usually implemented as scrambling, block coding or a combination of the two [Sac05]. Scrambling (see later, section 5.1) introduces no redundancy to the source sequence but does not fully constrain the low frequency content or ensure transmission of adequate timing information, even though very long runs are highly unlikely.

In block coding a contiguous group of bits (a block) is replaced by another slightly larger group of bits such that the average mark density becomes 50% and DC balance is established. It requires an increase in the bit rate, but the maximum run length can be strictly limited by careful choice of the coding technique. Examples of commercial line codes based on block coding are CIMT (see section 2.1.3.1) and 8b/10b (see section 2.1.3.2). The 64b/66b is instead based on a scrambler (see section 2.1.3.3). The proposed line code scheme, based on scrambling and Reed-Solomon error correction (and the core work of this thesis), is presented in chapter 5.

It is important to note that each different standard has a different set of goals and requirements, and that the line code is designed to match the link and its communication channel. In the case of optical recording on compact disk support, for example, an ad-hoc code was proposed: the so-called "eight-to-fourteen" code [Imm81]. The requirement is that sequences of "0"s and "1"s should not be shorter than 3 or longer than 11. This is fulfilled by concatenating an eight-to-fourteen lookup table with an NRZI binary code. The lookup table ensures that every byte gets converted into a sequence of length 14 and maximum weight 4, in which binary "1"s are always separated by a minimum of two and a maximum of ten binary "0"s.

Then in the NRZI code a binary "1" is encoded as a change of value in the stream, while a "0"s is indicated by the absence of a change. The concatenation of the two guarantees a DC-free and run-length limited stream. The eight-to-fourteen code has an efficiency of 47%.

2.1.3.1 Examples of line encodings: CIMT

The Conditional Inversion Master Transition (CIMT) line coding scheme was introduced together with the G-Link chipset in the Hewlett-Packard Journal in 1992 [Yen92]. The chipset converts parallel data for transmission over gigabit serial links, and comprises both transmitter and receiver for the implementation of a bidirectional link for point-to-point communication.

The coding scheme is designed to transmit either 16 or 20 bit-wide data words; four control bits are added to each. The efficiency can be as high as 21/24 or 87.5% [Wal92].

Frames are conditionally inverted as necessary to maintain DC-balance, according to the accumulated disparity. If frame inversion was performed on the transmitter side, the inversion is performed again at the receiver to restore the information as originally sent. The conditional inversion guarantees bounded digital sum variation [Wal91].

One master transition is inserted in a fixed position in the frame within the four control bits. This transition is the reference for the clock recovery circuit to restore frequency and phase of the "low frequency" clock. This avoids the need for periodical interruption of the service in order to send synchronization characters. At the same time the reference transition operates as a frame alignment indication and guarantees a maximum run-length equal to the frame length. Bit clock is reconstructed by dividing the frame clock by 20 or 24.

The four control bits have the additional function of distinguishing among three different frame types: data frames, control frames (to be used when the information packet should be handled in a different way from pure data, i.e. to be used for packet headers) and fill frames.

Fill frames are sent automatically at link startup or when no data or control are to be sent by the user. At link startup, they consist of a pure square wave easily allowing for accurate frequency locking. At the same time, they are also used to implement an initial handshake protocol in order to notify acquired lock to the other side of the link. At first the two

transmitters send a first type of fill frame. As the link is bidirectional, once the chipset receiver has acquired lock, it notifies the other side of the link by sending a second type of fill frame through the transmitter. This second fill frame is identical to the first, apart from the position of the falling edge of the square wave. In this way, an indication of the link readiness for transmission is implemented. Finally, when the link is operational and frequency lock has already been acquired, fill frames essentially provide the receiver with the master transition in order to maintain phase and frequency lock even in the absence of information to be sent.

In Table 1 the contents of different frame types are exemplified for CIMT encoding of 20-bit data. The first column shows the data field contents (the actual bit values are given only for fill frames). The second column reports the contents of the control field: the master transition is between the second and third bit of the four control bits. The third column shows the frame type

Table 1: Contents of different frame types for CIMT encoding of 20-bit data [Yen92]. FLAG is an additional flag bit input. D0 to D19 are the parallel inputs.

Data Field	Control Field	Frame Type
111111111 10 000000000	00 11	Fill (FF0)
111111111 00 000000000	00 11	Fill (FF1L)
111111111 11 000000000	00 11	Fill (FF1H)
D0-D8 01 D9-D17	00 11	Control
D0-D8 10 D9-D17	11 00	Inverted Control
D0-D19	11 01	Data, FLAG low
D0-D19	00 10	Inverted Data, FLAG low
D0-D19	10 11	Data, FLAG high
D0-D19	01 00	Inverted Data, FLAG high

2.1.3.2 Examples of line encodings: 8b/10b

The 8b/10b is a very popular line code invented at IBM [Wid83]. It has been widely used, for example qas been adopted in the Fiber Channel standard and in the IEEE 802.3 Gigabit Ethernet standards (1000Base-SX, 1000Base-LX) for data communication systems.

The code maps every source byte into a constrained 10-bit binary sequence by subdividing it into two parts: 5 source bits are mapped to 6 line bits and the remaining 3 bits to 4. The coding tables (a few examples are reported in section 3.4.1) are designed so that a minimum number of bits must be changed when passing through the two encoders.

The encoded sub-blocks are chosen in such a way that the permitted disparity for the each of them can be only 0, +2 or -2. This, together with the fact that the polarity of nonzero disparity blocks alternates, guarantees maximum digital sum variation of 6 and, consequently, that the line stream is DC-free. The code guarantees also a maximum run length for the signal stream: maximum 5 consecutive identical symbols, obtained by a careful choice of mapping.

Special characters are defined for special functions in the link like establishing byte synchronization and marking start and end of packets. A comma is defined for the indication of byte boundaries and is used for acquisition or verification of byte synchronization. The comma signal has to be periodically sent over the link for verifying the synchronization, and this generates an interruption of the service.

The code redundancy can be used for error detection [Wid83] at the packet level. In general, error patterns that violate the alternating disparity rule or generate illegal characters can be detected.

Single errors in the encoded line digits generate bursts in the decoded stream (see Figure 6). These bursts are confined to the 6-bit or 4-bit subblock in which they occur and consequently are no longer than 5 or 3 bits respectively. This is due to the fact that each 6b or 4b subblock is uniquely decodable on the basis of just the bit values belonging to that sub-block. In the original paper [Wid83] Fire codes are suggested as an additional CRC protection for the 8b/10b code: packets as long as 142 bytes need 16 check bits and packets as long as 36862 bytes need 24 check bits for the detection of two bursts of total length 10 bits, this is a double error in the line digits.

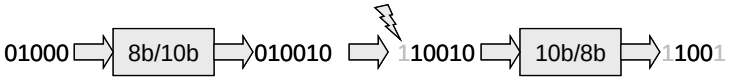

Figure 6: Example of error multiplication with 8b/10b line code.

2.1.3.3 Examples of line encodings: 64b/66b

The Open System Interconnect (OSI) model defines a networking framework for implementing protocols in seven layers. Control is passed from one layer to the next, starting at the highest and proceeding to the bottom in one station, over the channel to the next station and back up the hierarchy. Ethernet was standardized as IEEE 802.3 standard, and refers to

the two lowest layers of the OSI model. The lowest layer is called the *Physical Layer*.

The *Physical Coding Sublayer* (PCS) is part of the Physical Layer and is responsible for frame delineation, frame formatting and line coding. When working on the serial 10 Gb/s standard for Ethernet (IEEE Std 802.3ae-2002) in 1999, a discussion was opened on the choice of physical layer coding scheme. The default choice would have been maintaining the 8b/10b standard that was in use for the 1 Gb/s version. This choice would have made it possible to profit from the reuse of years of expertise in the 8b/10b standard. Opponents however argued that the 25% overhead was unacceptable to maintain backward compatibility, given that it would have meant an increment in bandwidth from 10 to 12.5 GHz.

At the same time, new codes were also proposed as alternative solutions to the line coding problem. M810 for example was introduced [Lee00]: a line code which is binary, run-length limited, dc-free and of minimum bandwidth. The Simple Link Protocol [Tru01] was also proposed, based on a scrambler and frame delimiters.

Finally, the 64b/66b encoding scheme was chosen [Iee00]. The scheme is implemented by adding a 2-bit header to a 64-bit scrambled data packet. A “01” synchronization preamble is used if the 64-bit packet is constituted of pure data, a “10” preamble is used for mixed data/control words or pure control words. Both “00” and “11” preambles are considered errors. This preamble is used in the receiver for frame synchronization. Synchronization is acquired after 64 contiguous frames are received with a valid “01” or “10” synch header. Frame synch is declared lost after 32 “11” or “00” synch patterns are found in any block of 64 frames. In fact, if the frame is not synchronized then the two bits belong to the scrambled packet, and this means that the four possible preamble values are all equally probable, so on average half are valid and the rest are not.

Concerning the choice of a scrambler, more details are given in section 5.1. The order of the scrambler should be less than 64 in order to minimize implementation complexity, and higher than 57 following jamming probability analysis [Iee00]. A scrambler of order 58 is chosen for the 64/66 line code. Moreover, the scrambler is implemented in a self-synchronized version which allows the receiver to resynchronize, autonomously avoiding the need for a synchronization command. At the same time though, this implementation choice has the drawback of causing error multiplication.

This is taken into account by the scrambler order, which is chosen to minimize the interaction with the Cyclic Redundancy Checks at high protocol levels.

2.2 Optical links in radiation environments

2.2.1 The experiments at the LHC as radiation environments

The luminosity is planned to be the increased by a factor of ten from the LHC to the SLHC upgrade [Der06]. In the case of proton collisions, that leads to an average production of 10^{10} inelastic proton-proton collisions per second, from 10^{9} collisions expected at the LHC. In addition to the primary collisions, induced radioactivity will also be a major concern at the LHC and its upgrade. In fact, the high beam intensity combined with the high luminosity results in numerous intense cascades, which give rise to numerous low energy particles (energy range around and below 1GeV).

This creates a very hostile radiation environment for all the machines and equipment at the LHC, and in particular requires that all ICs in the experiments be radiation resistant and the systems to be radiation tolerant. For example, ASIC designs in applications for HEP experiments require radiation hardness up to the Total Ionizing Dose of 100 Mrd level [Fac05].

2.2.2 Radiation-matter interaction and effects

The manner in which radiation interacts with solid materials depends on both the incident particle and the target material [Rod03, Sch94]. Charged particles like protons, electrons or heavy ions interact mainly through Coulomb attraction or repulsion with the electronic clouds of the target atoms, causing ionization or atomic excitation. Massive particles, like protons, neutrons and heavy ions can collide with the nuclei of the target material, causing excitation, displacement or nuclear reactions. Electrons can also generate X-rays (Bremsstrahlung) when decelerating into the target. Photons, being massless and chargeless, can interact with matter through photoelectric effect, Compton effect and creation of electron-positron pairs [Sch94].

The effects through which both charged and neutral particles lose energy passing through matter can be grouped in two classes: nuclear displacement and ionization effects. Neutrons, which are neutral and massive particles,

give origin mainly to nuclear displacement, whereas photons and electrons are mainly responsible for ionization effects.

When a particle passes through a semiconductor, it can collide with a nucleus and displace it from its normal position, thus altering the regular crystalline structure of the semiconductor material. This effect is called displacement damage because the disruption in the lattice structure causes variations in the behaviour of the device. The stable lattice defects cause energy levels to be introduced into the bandgap, which separates the valence and the conduction bands of the semiconductor. These can act as generation, recombination or trapping centers which, for example can decrease the minority carrier lifetime in electronic devices, and increase the thermal generation rate of electron-hole pairs and reduce the mobility of carriers. As a result, displacement damage is a concern primarily for minority carrier devices (i.e. bipolar transistors) and optoelectronic devices, while it is relatively unimportant for MOS transistors.

Ionization in a semiconductor or insulating material is the phenomenon in which a highly energetic particle (i.e. electrons and protons) can remove one electron from its parent atom producing a free electron and consequently a mobile hole (electron-hole pair creation). As long as the energies of the electrons and holes generated are higher than the minimum energy required to create an electron-hole pair, they can in turn generate additional electron-hole pairs. When the electronic component is biased, an electric field is applied where the holes move in the field direction while the electrons drift in the opposite direction. Holes are slower than electrons and may be trapped in the component oxide films. The deposition of energy in a material by means of ionization is conventionally termed “dose” that is the energy absorbed locally per unit mass as a result of irradiation by energetic particles. The number of electron-hole pairs created is proportional to the quantity of energy deposited in the material, which is expressed through the total absorbed dose. The ionization dose effects produced in a component can be divided in two types: long-term effects (which are cumulative) and transient effects (or single events).

From a general system level point of view, it can be said that if radiation effects are not properly taken into account they can cause malfunctioning (single events), system slow down (slew rate degradation due to total dose effects), or increased power consumption (i.e. increase in leakage currents due to total dose effects).

The impact of cumulative and transient effects on both optical and electronic components of an optical link, are presented next in sections 2.2.3 and 2.2.4. Section 2.2.5 carries examples of previously adopted system solutions.

2.2.3 Radiation effects on optical components

Concerning the optical components, it is widely recognized that particle induced displacement damage is able to affect the light output of optoelectronic sources as well as cause dark current increases and responsivity degradation in optoelectronic detectors [Kal04]. Passive devices such as fibers and connectors are more susceptible to TID, whereas active devices like lasers and photodiodes are most sensitive to DD. System implementations may leverage on Commercial-Off-The-Shelf (COTS) technologies, which minimize the cost of developing customized components and systems. Typically though, they have to undergo very careful design cycles where a detailed understanding of component and subsystem radiation performance is mandatory [Jen99]. For example extensive testing is required to verify radiation tolerance over the desired amount of time and radiation doses.

2.2.3.1 Laser diodes

Semiconductor lasers are in general more susceptible to displacement damage [Kal04] than to other radiation effects. Displacement damage introduces defect states into the bandgap and these can act as generation-recombination centers, increasing non-radiative recombination and decreasing carrier lifetime. This results in higher threshold currents (i.e. increase of 100~200%, [Gil98]), lower output power at a given current (that is lower efficiency, i.e. 10~25%, [Gil98]), and higher leakage current. The shift in threshold current can be large and is for example approximately linearly related to proton fluence [Kal04]. The behaviour is qualitatively similar to the one shows in Figure 4, even though in this case, it is due to an increase in the age or operating temperature of the device.

2.2.3.2 Optical fibers

Total ionizing dose is the dominant damage mechanism in optical fibers [Kal04]. Ionizing radiation leads to the creation of color centers and causes optical absorption at certain wavelengths, such as visible or ultraviolet, effectively increasing attenuation. Radiation damage can also cause a

change in the refractive indices of the core and the cladding, leading to a sub optimal confinement of the optical mode in the fiber and consequently optical loss. Polymers, used in the various external coatings, are also affected due to excitation or ionization leading to a mechanical degradation.

Published results measure the additional attenuation in dB/m (0.1dB/m in [Tro98]), suggesting that if optical networks are sufficiently short, the effects of increased loss in fibers can be tolerated, as such losses can be readily accommodated in optical power budgets.

2.2.3.3 Photodiode

Photodiodes are found to be susceptible to both TID and DD effects. Radiation induced defects in the bulk build up over time, and generation of charge at the defects causes an increase in dark current, while trapping/recombination centers for electrons and holes cause a reduction of the photocurrent. The overall effect is a reduction of the responsivity (of up to 90% [Gil98]) and an increase in the dark current (of orders of magnitude [Gil97]), so that the minimum power level of detectable optical signals and the noise increase. As the signal-to-noise ratio decreases, increased power margins are required for correct operation over time.

Additionally, it is widely established that the photodiode is the most sensitive part of the link to Single Event Effect (SEE) by virtue of its low input signal level (few microwatts at rates into the Gbps/s regime), which makes it sensitive to false signals created by direct ionization by incident protons. Basically a radiation particle makes the diode react similarly to a light impulse coming from the transmitter.

In Figure 7 a Non-Return to Zero (NRZ) data stream is depicted: the NRZ stream hops between the values I_0 and I_1, corresponding to logic “0” and “1”, and the decision threshold I_{TH} is automatically adjusted to be midway between I_0 and I_1. A received signal higher than threshold at the decision point is detected as a logic “1”, lower than threshold as a “0”.

Radiation particles can ionize the detector and create additional current pulses in the receiver circuit which decay with an RC time constant determined by the circuit bandwidth [Bri93]: three possible particle strikes are depicted in Figure 7. If the transmitted bit is a digital “1”, the induced current only increases the photocurrent, and no error occurs. If the transmitted bit is a “0”, but the induced current is below the decision threshold, again no corruption occurs. An error results only if the induced

current is sufficiently high at the time of sampling. The ion-induced photocurrents flowing when a "0" has been transmitted increase the probability of false detection [Mar94].

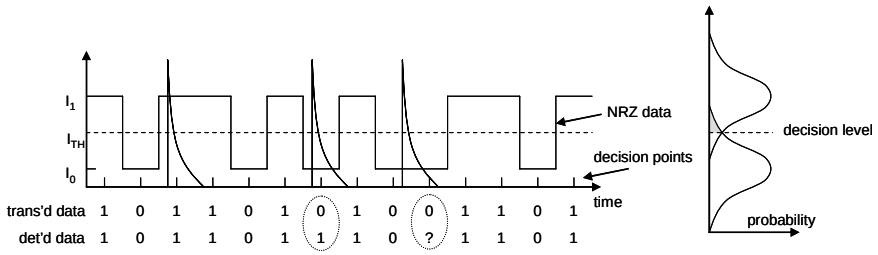

Figure 7: Proton ionization in receiver photodiodes induces photocurrents which may disrupt data. The first particle strike has no effect, the second probably corrupts the data and the third is far from the decision point, leaving doubts on the result. On the right probability distributions are depicted (not to scale). The shaded area represents the probability of erroneous detection from a transmitted "0" to a detected "1". The non-shaded triagular-shaped area is the probability of 0-to-1 erroneous detection.

The two probability functions on the right in Figure 7 are a distribution of the received signal levels for "1" and "0" logic signals. The distribution tail shaded area is the probability of detecting a "1" (signal above threshold) when a "0" is transmitted, and corresponds to making a decision error (similarly for a 0-detection after a 1-transmission). Any ion-induced photocurrent flowing when a "0" is transmitted increases the probability of erroneous detection. SEU errors on the photodiode are studied in more detail in section 3.3. As they cannot be fully accommodated by simply increasing the optical power budget as the other radiation effects, they are addressed through the use of forward error correction.

2.2.4 Radiation effects on ASICS and hardening techniques

Concerning ASIC components in the link, techniques exist to accommodate for radiation effects and make the devices become "radiation-tolerant". One option is to modify some of the process steps, but this is very costly due to the little demand for such techniques in the commercial world, compared to which the HEP community is too small to be able to modify general trends or to afford exotic processes. The other two possibilities are hardening by layout (section 2.2.4.1) or through system and circuit special architectures (section 2.2.4.2).

2.2.4.1 Hardening by layout

MOS transistors are almost entirely insensitive to displacement damage, since they are devices whose conduction is based on the flow of majority carriers below the SiO_2-Si interface, a region which does not extend deep in the silicon bulk.

Ionizing radiation instead has an effect on the MOS structure, and in particular in the silicon dioxide. When an ionizing particle goes through an MOS transistor, electron-hole pairs are generated, and while the electron-hole pairs quickly disappear in the gate and in the substrate, in the oxide the holes in particular may be trapped close to the interface giving origin to a fixed positive charge. Ionizing radiation also induces the creation of traps at the SiO_2-Si interface. These result in [Ane00]:

- The threshold voltage changes due to the holes trapped in the oxide.
- The subthreshold current increases due to the decrease of the threshold voltage.
- Leakage currents increase due to the generation of parasitic paths within single transistors and between different transistors.
- The mobility degrades, essentially due to the increase of the interface traps.

Scaling of technology helps in reducing TID damage. In fact reducing the gate oxide thickness t_{ox} improves the radiation tolerance of the gate oxide: for thinner oxides, the threshold voltage shift and the mobility degradation are lower than for thicker oxides (see Figure 8).

In particular for the 0.13 μm technology to be used, the gate oxide for core transistors seems to be extremely TID tolerant [Fac05] even after doses in the order of several tens of Mrd. For transistors with thicker gate oxide (such as I/O transistors that allow integration with systems requiring 2.5 V logic levels), wider variations of the threshold voltage and higher leakage currents are observed [Fac05].

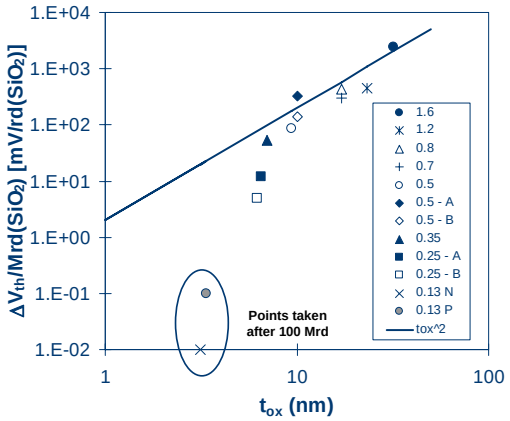

Figure 8: Variation of threshold voltage normalized to the Mrd dose. Measurements done in the Microelectronics Group of CERN (only 4 points are taken from the literature: the 1.6, 1.2, 0.8 and one 0.5 um) for different technology nodes.

In the previous technology node utilized in the microelectronics group at CERN (0.25 μm), hardness-by-design techniques were implemented [Ane99], which were namely enclosed layout transistors (to eliminate leakage paths for single transistor) and heavily doped P+ guard rings around each n-well or n+ diffusion (to prevent inter-device leakage currents). These measures seem to be unnecessary for digital circuits in the 0.13 μm studied technology [Fac05].

2.2.4.2 Hardening by system and circuit architecture

A Single Event Effect (SEE) results from the charge deposited by a single particle crossing a sensitive region in the device and then collected at a sensitive node [Duz03]. If the amount of deposited charge is sufficient, an immediate malfunctioning is generated.

The errors generated can be reversible (soft errors, i.e. non destructive) or non reversible (i.e. hard errors, destructive, which cannot be tolerated). Single Event Latch-up is an example of destructive SEE as it is a latch-up phenomenon initiated by an ionizing particle.

A Single Event Upset is the instantaneous reversible modification of the logic state of an elementary memory cell, due to, for example, charge collected in the circuit at a sensitive node. Concerning scaling influence on SEUs, a clear indication from the literature is not available, even though

SEU occurrence could worsen with scaling [Ane06]. Thus, in general, SEUs still need to be addressed in modern technologies such as 0.13 μm [Fac05].

Generally for digital circuits, the synchronous mode of operation limits the sensitivity to electrical parameter variation, which affects analog blocks more thoroughly. SEUs though can still affect the application if they modify some piece of information stored in crucial memory elements. For this reason, state machines are triplicated in critical radiation environment applications (triplication is the most common form of hardware redundancy [Nir96]).

2.2.5 Examples of links to be used in radiation environments

Two examples are reported which exemplify the use of optical links at the LHC: links for the CMS Tracker and an ASIC for data transmission, the GOL.

2.2.5.1 The CMS Tracker Readout and Control System

The Tracker is the innermost detector in the CMS experiment and is formed of approximately 10 million individual detector channels. About 40000 uni-directional analogue links perform the data readout and ~2500 digital bidirectional links constitute the control system [Tro03]. Both systems operate single-mode at 1310nm wavelength with edge-emitting InGaAsP laser diode transmitters and InGaAs p-i-n photodiode receivers of small active volume[Vas98].

Concerning the optical components of the systems, it was possible to meet the requirements with the extensive use of commercially available components with a minimum of customization. Thus, it was possible to benefit to a large extent from the routine quality testing carried out by the manufacturers even though an initial pre-production qualification had to be carried out [Pap03] to verify compliance, and less thorough batch testing had to be performed to verify process stability. Extensive studies were carried out to measure the radiation response of all optical components to be used in the radiation zone, proving that the components are sufficiently radiation resistant and allowing to parameterise their response to predict the degradation due to radiation exposure within CMS (i.e. [Tro98], [Gil98]).

Concerning the ASIC components of the link, a 3-way Laser Driver ASIC [Cer01] was designed and implemented in a 0.25 μm CMOS commercial technology according to the CMS Tracker performance and

radiation-tolerance requirements. The linearly amplified signals are superposed to a DC-current, programmable over a wide range (0-55 mA). The programming of the DC-current substitutes the automatic power control mechanism (introduced in 2.1.2) in compensating for the aging-caused or radiation-induced threshold increase.

A receiver ASIC was designed comprising three main blocks, a preamplifier, a limiting amplifier chain and an output driver [Fac01]. The Single-Event Upset (SEU) behaviour of the receiver coupled to a prototype photodiode was measured and at the typical in-system optical power level of -10dBm, a BER below 10^{-12} is predicted for operation under the Tracker particle flux of $10^{6}cm^{-2}s^{-1}$, indicating the digital control link to be SEU tolerant to the required level.

2.2.5.2 GOL

In the LHC detectors, massive amounts of data will be generated and will have to be transmitted out of the different sub-detectors for storage and off-line data analysis. Optical links operating in the Gbit/s range were chosen for these applications. Commercial components could be found meeting the needs existing in the HEP environment, apart from radiation resistance. In particular, as the transmitters of the aforementioned links are subject to radiation, only they needed to be developed and qualified for radiation tolerance; all other parts in the chain could adopt commercial components, thus reducing the development and maintenance costs.

Consequently, a transmitter ASIC was developed [Mor01] that is capable of operating with two of the most common data transmission protocols, so that commercial components can be used in the parts of the link that do not sit in the radiation environment. The transmitter ASIC is designed using radiation tolerant layout practices that guarantee tolerance to irradiation effects to the levels necessary for the LHC experiments.

The transmitter ASIC performs the function of a serializer and can operate in four different modes that are a combination of two common transmission protocols (8b/10b or CIMT) and two data rates (0.8 Gbit/s and 1.6 Gbit/s). The data input comes from a data bus operating either as a 16 or 32-bit bus synchronously with the LHC clock (running at 40 MHz), resulting in data bandwidths of 640 Mbit/s and 1.28 Gbit/s respectively, for serial data rates of 800 Mbit/s or 1.6 Gbit/s. Depending on the chosen line coding, either a G-Link or a Gbit Ethernet/Fiber Channel receiver can be

used at the other end of the link. Once serialized, the encoded data can be used to drive either a laser, or a 50 Ω line. In the case of the optical transmission, due to radiation effects, an increase in the threshold current of the laser diodes over the lifetime of the experiments is expected. To compensate for this, the laser-driver contains an internal bias current generator that can be programmed to sink currents between 0 and 55 mA.

2.3 Summary

This introductory chapter presents optical links to be used in radiation environments. The choice of optical links is initially motivated, followed by an introduction on link building blocks. The need for line coding is motivated and examples of commercial line codes are presented. The effects of radiation on link components are then introduced, and the fact that SEUs on the photodiode are the main issue to be addressed is motivated.

Thus, in optical links for HEP, line coding has to be combined with error correction capability to resolve SEUs on the photodiode. The development of a line code with error correction capability is in fact the main contribution of this thesis, which is introduced in the fifth chapter of this thesis.

Chapter 3

The VBD Link and the GBT ASIC

This chapter presents the Versatile Bi-Directional (VBD) Link, for which the proposed line code was developed. As the new link is designed to serve several systems with different data transmission requirements in HEP, these roles are first presented. Data acquisition from the detectors to the adjacent counting rooms was already introduced in section 2.2.5.2 during the discussion of the GOL ASIC. The Timing, Trigger and Control system employs an optical link, called the TTC link, which is described in section 3.1, together with an explanation of its requirements and the limitations which drove its upgrade.

The VBD link and its functionality are presented in section 3.2 together with possible link configurations. The transceiver ASIC is also presented, as the line coding developed for this thesis is a part of it. In section 3.3, the characteristics of the errors in the photodiode are discussed in order to understand the requirements on the error correction capability of the line code. Finally, in section 3.4, the combination of existing line codes and error correction codes are studied, motivating the need for the work carried out in this thesis.

3.1 TTC system and the TTCrx ASIC

The Timing, Trigger and Control (TTC) system is the distribution scheme for fast timing signals at the LHC [Tay02].

The timing signals generated by the LHC RF generators have to be distributed to all experiments and beam instrumentation. In the system the

timing signals are conveyed from the RF generators to the LHC central control room in the CERN Prevessin site via single mode optical fibers which are slightly less than 10 km long. The control room is then the star distribution point to the ATLAS, Alice and LHCb experiments: these connections are via single mode optical fibre, at 1310 nm, and the lengths differ depending on the destination (between 3 and 10 km long). For the CMS experiment, the fibers follow the shortest path along the tunnel from the RF generators, instead of passing through the control room [Bar06].

Once at the experiment sites, additional trigger and control information is joined to the timing signals and distributed through the TTC system. At the experiment level, the trigger acceptance is generated, reset commands for registers are sent and decisions are made regarding a sub-detector mode (i.e. test or calibration).

More details about the three main functionalities are explained in the following sections 3.1.1, 3.1.2, 3.1.3 respectively for timing, trigger and control. The line coding technique chosen for the TTC is described in section 3.1.4 and the limitations of the system which motivated an upgrade are listed in section 3.1.5.

3.1.1 Timing

The LHC beam is not continuous but constitutes a series of bunches, that is groups of particles which move together in the accelerator. The bunch spacing is 25 ns or about 7.5 m, which corresponds to an accelerator operation frequency of ~40.079 MHz. Moreover, the bunch filling is not continuous due to issues related with the accelerator chain, and only 2808 bunches out of 3564 are present [Met06]. The peak luminosity in the LHC machine for Atlas and CMS is designed to be 20 events per bunch crossing (i.e. 10^{34} cm^{-2} s^{-1}) and a factor of ten more for SLHC.

The TTC distributes the bunch clock and the orbit signal, which allow precise identification of the bunch event number. They are derived from the LHC RF generators and their frequencies vary slightly during acceleration, as they are synchronous to the circulating beams. The timing properties of these signals are critical as detector electronics, DAQ and beam instrumentation work synchronously to the machine and consequently rely on those signals for detector synchronization, trigger system alignment, assignment of bunch crossing to data, and pipeline synchronization. Thus,

the delays associated with the different signal paths and the jitter properties of the signal are to be carefully controlled.

The TTCrx ASIC [Chr96, Chr03] is the radiation-hard ASIC receiver which acts as the interface between the TTC system and the detector front end electronics. It is composed of a full custom part for the analog and timing critical functions, plus a standard cell implementation for digital logic and non-time critical functions. Within the full custom part, clock and data are recovered and a fine deskewing function is implemented. The digital part of the chip contains several internal registers used for control and monitoring.

The TTCrx receiver is equipped with all signals necessary to synchronize the detectors. The 40 MHz LHC clock is extracted from the serial data stream and fed into two independent high-resolution phase shifters which provide a fine programmable delay in steps of 104 ps between 0 and 25 ns. An additional coarse delay register allows a compensation range of up to 16 bunch-crossing intervals, which can be used to compensate for the propagation delays associated with the detectors and their electronics. The bunch counter and event counter registers keep track of bunch and event collision numbers.

The timing requirements of the TTC system are strict: an additional ASIC component has to be used as a complement of the TTC system in all the situations where the TTCrx clock jitter proves to be excessive. The quartz crystal based phase-locked loop (QPLL, [Mor03]) can reduce the jitter level to a cycle-to-cycle jitter of 22 ps RMS from 76 ps RMS at the output of the TTCrx.

3.1.2 Trigger

In Figure 9, proton bunches approaching the center of the detector for a collision are shown. It is also shown how the collision between two bunches is in fact an event of discrete type, where a collision of two protons is in fact a collision between the partons (quarks and gluons) which form it. Only a head-on collision of two partons can have enough energy to give rise to an interesting event. Proton bunches cross at the rate of 40 MHz. Despite the large number of protons per bunch (10^{11}) and due to the granular structure of the bunch, the proton collisions happen at a rate that is only 10^{9} Hz. Many of these collisions though are not interesting from the point of view of new physics: “new” particles are in fact produced at a rate of $\sim 10^{-5}$ Hz. At

the same time, not all the produced data can be either driven out of the detector (due to bandwidth limitations), or stored (due to storage limitations).

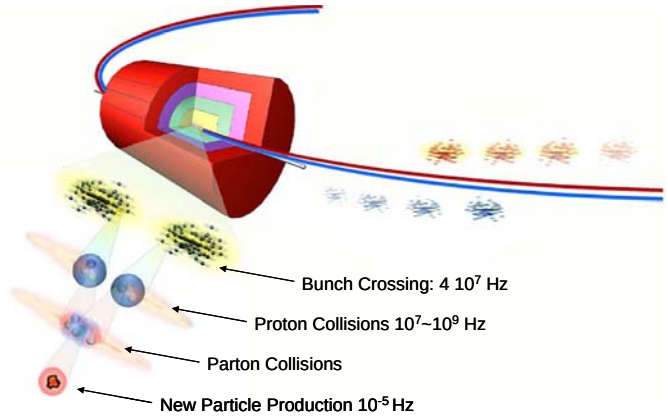

Figure 9: Collisions: bunch structure, particles interaction [Sph06]. The overall event selection is 1 in 10,000,000,000,000.

The data abundance problem is usually addressed in the experiments through a technique called "triggering" [Sph06]: the data is subsequently selected through various levels of decisions, called trigger levels (e.g. two levels in CMS and three in ATLAS). The first level trigger, or level-1 (L1) trigger reduces the rate from 40 MHz to 10^5 Hz.

The L1 trigger's accept signal has important latency issues as it has to be promptly decided if the data is to be processed further or to be dismissed right away. Data from the sub-detectors is collected, transferred to the processors outside the detectors and processed, and then the decision has to be distributed back into the detector. All these operations have to be performed within a 3 μs interval, as that is the time available before the information on the collision falls off a front end pipeline and is thus lost.

3.1.3 Control

Two types of commands can be delivered with the TTC system [Chr03]: broadcast commands and individually addressed data. Broadcast commands are used to distribute messages to all TTC receivers in the system. These messages are used for example to reset the event counter and the bunch counter.

Individually addressed commands are implemented in the TTC system to transmit user-defined data and commands over the network. Each TTCrx can be addressed independently as each one is identified in the distribution network by a unique 14-bit channel identification number. The individually addressed commands are either aimed at the TTC receivers themselves to control the receiver operation (i.e. regulating deskewing), or the data are intended for the external electronics.

Both the broadcast and the individually addressed commands are transmitted over the TTC network using the frame formats depicted in Figure 10. The information in the frames is protected through a 1-bit error correct 2-bit error detect Hamming scheme: 5 check bits protect the 8-bit data packet for broadcast commands, 7 check bits protect the 32-bit data packet for individually addressed commands.

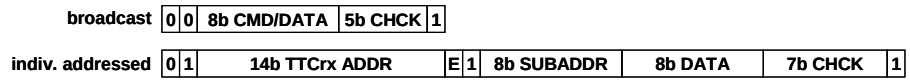

Figure 10: Control data frame formats [Mor03]: broadcast format (top) and individually addressed format (bottom). In both frames: the first bit is set to zero as a "start of packet", the second defines frame type ("0" for broadcast, "1" for individually addressed), the last one is set to "1" as "end of packet".

3.1.4 Line coding in the TTC system

Two channels are time multiplexed and transmitted in the TTC system [Chr03]. Channel A is reserved for the L1 trigger information only. Channel B is used for delivering the slow control information.

Only one bit of information is delivered per channel per bunch crossing for a total of 80 Mb/s. The L1A is a 1-bit information, either "pass on" or "reject" the data, and is not protected by any error control scheme. The information on channel B is instead formatted over multiple bunch crossings to create the frames shown in Figure 10.

The two time-division multiplexed channels are biphase-mark encoded before transmission over the network. This line code consists of representing a logical "1" as a pair of different bits ("10" or "01") and a logical "0" as two equal bits ("00" or "11"). It thus requires a line frequency that is twice the data bandwidth. As every logical level at the start of a cell is inverted with respect to the level at the end of the previous cell, this encoding scheme provides a very high number of transitions, at least one

every encoded bit that is sent. Moreover, the data stream is DC-balanced. The coding scheme is sketched in Figure 11.

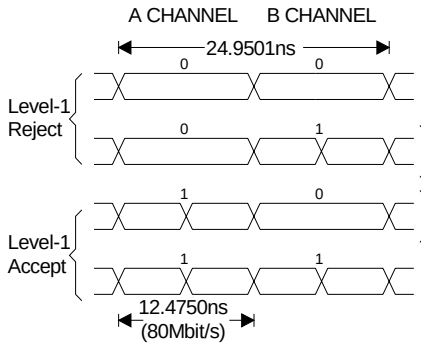

Figure 11: Time division multiplexed biphase mark encoding [Mor03].

3.1.5 TTC limitations

A TTC system upgrade is under study in view of the upgrade of the LHC machine to SLHC. In particular, the issues that need to be addressed in a new system are summarized below:

- The transmission speed needs to be increased, especially on Channel B where the transmission of control commands requires multiple LHC clock periods;
- Channel A, which carries trigger information needs to be protected by an error correction code while in the present TTC only channel B is protected;
- Extended trigger functionality needs to be provided, namely the possibility of transmitting different trigger masks, instead of a 1-bit only decision.

These issues are mostly due to technological limitations at the time of development and can be addressed by increasing the speed of the link so that more bits can be delivered with each LHC clock. If the ASIC implementation utilizes a recent technology, line speeds in the order of a few Gb/s are easily reachable, and this would guarantee several tens of bits per bunch crossing, enough to expand the trigger and control data spaces and to allow the addition of redundancy bit to be used for error correction capability.

3.2 The Versatile Bi-Directional Link

The VBD Link is the link for which the line code proposed in this thesis was designed. The idea of a "versatile" link is presented in section 3.2.1, possible link configurations and frame formats in sections 3.2.2 and 3.2.3, while the top-level block diagram of the GBT ASIC is reported in 3.2.4.

3.2.1 The concept

Data transmission is used in HEP for data acquisition (DAQ), for timing and synchronization of the detectors, for triggering and for experiment control.

The Versatile Bi-Directional Link project is based on the idea of developing one solution that can be adapted and used for the different functionalities listed previously. The objective is to build a system based on a single ASIC, the GigaBit Transceiver (GBT), a limited set of optoelectronics "flavours", the same multi-chip module (as sketched in Figure 12) so that development and test efforts can be optimized.

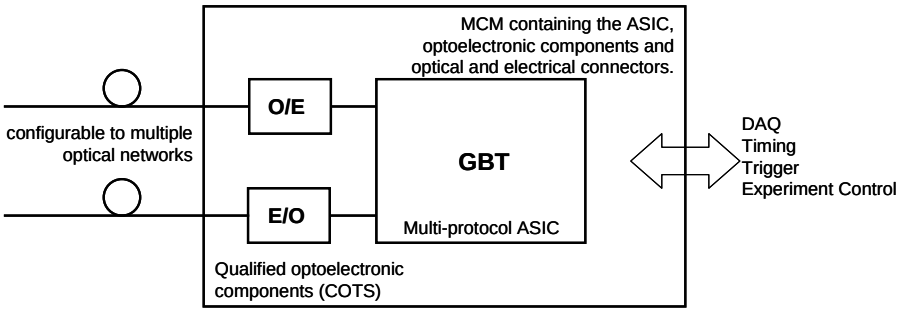

Figure 12: Transceiver module scheme.

The proposed link can implement different link topologies, have bidirectional capability and have higher bandwidth compared to the previously implemented LHC links due to the use of more recent technologies. The links would also be guaranteed to robustly handle irradiation effects, for both total dose and SEU, as the ASIC would be designed with radiation hardening techniques and the optoelectronic components would be qualified for use in a radiation environment.

3.2.2 Link configurations

For the link to be versatile, the same ASIC transceiver has to be able to handle different link topologies and different modes of operation: it is designed to operate as a general purpose link or as a TTC link. When operating as general purpose link, operation can be simplex or duplex and the full data bandwidth is available for data transmission without any restrictions on the data contents. On the contrary, when operating as a TTC Link, the data contents of the transmission are preallocated so that trigger functions and control commands can be implemented. In this mode, the down-link operates by transmitting a continuous stream of data (broadcast to many destination), while the up-links transmit bursts of data for talking to a single destination.

Consequently, the ASIC handles both continuous and burst (packet mode) data transmission. The case of continuous transmission consists of one transmitter which occupies the link 100% of the time, and the transmitter and receiver pairs are fully synchronous. On the contrary, burst transmission (or packet mode) is used, for example, when a common transmission medium is shared, and one transmitter can only send data upon the request of a master transmitter, so that several devices can communicate with the same destination, and collisions are avoided.

Four possible network connections are shown in Figure 13 through Figure 15: broadcast networks with and without repeaters (Figure 13), point-to-point connection (Figure 14), and bidirectional return link (one-to-N / N-to-one, Figure 15).

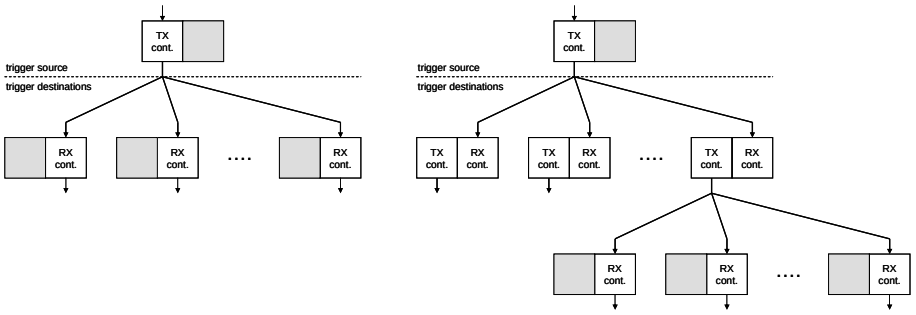

Figure 13: Broadcast network: simple (left) and with O/E/O Repeaters (right).

The broadcast network configuration (see Figure 13, left) is similar to the one currently employed for the TTC system architecture, and operates in

continuous mode. However, a high fan-out (i.e. 1-to-1024 as in the TTC) requires the use of high power optical sources, and this is outside the GBT laser driver capabilities (a fan out of 1-to-8 or at maximum of 1-to-16 is foreseen). The fan-out limitation can be overcome by using a mixed optical/electrical tree as represented on the right-hand side of Figure 13. In this case a master transmitter broadcasts optically to several destinations, and these in turn electrically regenerate the signal and retransmit the master's data to several other destinations further down the tree. Noted that the second topology is characterized by higher latency due to the regeneration steps.

The point-to-point configuration shown in Figure 14 is used for plain bidirectional data transmission without any restriction on the data contents. The link operates in continuous mode, simplex or duplex, and its main purpose is data transmission.

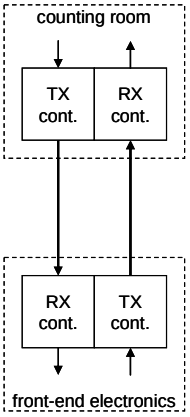

Figure 14: Point-to-point connection.

In the configuration shown in Figure 15, the link uses an optical tree network in both directions. The trigger-continuous down-link broadcasts data from one master transmitter to N slave receivers. The up-link uses a similar optical passive tree to collect data from several slave transmitters to a single destination (N-to-1), the master receiver. The slave transmitters operate in packet mode. The master transmitter arbitrates the access of the slave transmitters to the up-link optical tree.

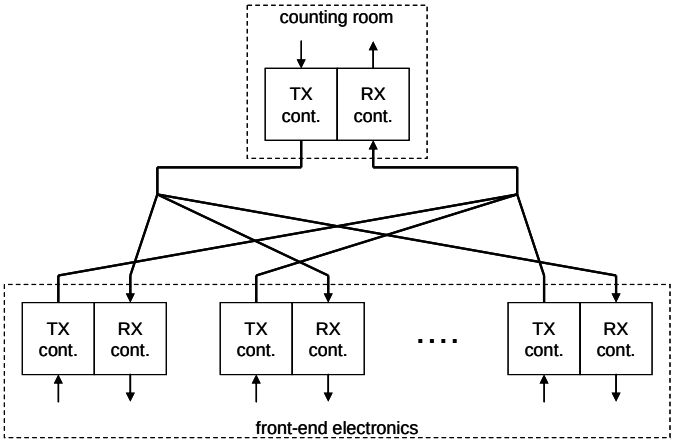

Figure 15: Bidirectional One-to-N / N-to-One link.

3.2.3 Frame structure

A few different frame formats are foreseen due to the fact that the transmitter and receiver work in two operation modes, trigger and general purpose, combined with two link modes, continuous and packet. For simplicity, this paragraph focuses on a 64-bit data packet, which is consistent with the first option for the proposed code (discussed in detail in the fifth chapter of this thesis). Slight changes have to be applied in case the second code option is chosen, as it is based on a 60-bit data packet.

For a 64-bit data packet to be delivered synchronously with the 40 MHz LHC clock, the data bandwidth is 2.56 Gbit/s. The proposed line code forms the frame by adding an 8-bit header and a 16-bit redundancy check field (see Figure 16), for a total 88-bit frame length resulting in 3.52 Gbit/s line data rate.

Figure 16: General frame format.

All frame formats then take the general form displayed in Figure 16 regardless of the link functionality, and only the packet field is differently interpreted according to the chosen transmission mode in order to address specific operation features (see Figure 17).

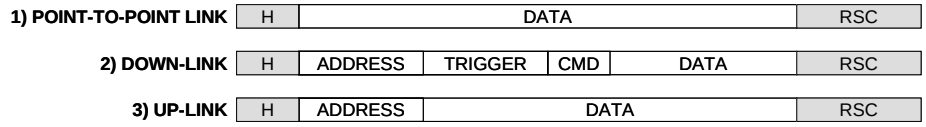

Figure 17: Frame formats for the different transmission modes to be used.

For a point-to-point link the full bandwidth is available to the user. Thus, 64 data bits are available for data transmission in both directions (up and down links) and no restrictions are made to the data contents (Figure 17, 1).

In the case of the down-link trigger-continuous mode, the packet field of the frame (Figure 17, 2) is subdivided into an address field, a trigger mask field, a command field and a data field. The 16-bit address field is reserved to identify a slave receiver in the network, and some of the possible addresses are reserved for global addressing (broadcasting) of the slave-receivers. The 16-bit trigger field is reserved for trigger distribution at constant latency, and is directed to all the slave receivers. The 8-bit command field is used to control the operation of the receiver itself and is validated by the address field; i.e. the command field is used by the master-transmitter to arbitrate the access of the slave transmitters to the up-link optical tree. The 24-bit data field is used to transmit data to a slave receiver or to some other piece of electronics that interfaces to the GBT ASIC.

For the up-link, operating in the link-packet mode (Figure 17, 3), the packet-field is subdivided into a 16-bit address and a 48-bit data field. The address field is used to identify the slave-transmitter accessing the return link while data field carries data from either the GBT ASIC itself or from the electronics interfacing to it.

3.2.4 The GBT ASIC

In Figure 18 a simplified block diagram of the GBT is reported. Some of the blocks are typical components of transceivers (i.e. CDR and parallel/serial converter), others are specific to the trigger functions (i.e. trigger logic and bunch emulator), perform control of error conditions that are common in HEP environments only (i.e. SEU monitor, watchdog), or help testability (i.e. self test logic). Many parallel and serial ports following well established industrial standards are available: 1–Wire, I2C, JTAG.

The blocks performing line encoding and decoding are those concerning the work carried out in this thesis; they sit next to serializing and deserializing blocks.

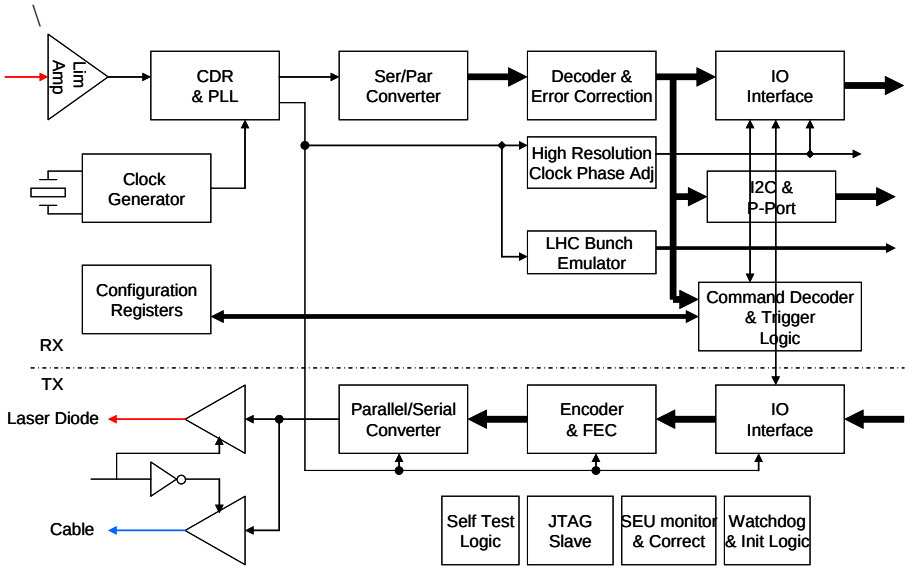

Figure 18: A simplified block diagram of the GBT13. Block without linking arrows represent functionally that it is common to both the transmitter (TX) and receiver (RX) functions.

As the GBT ASIC should be able to operate even if neither the receiver nor the transmitters are locked to the LHC reference clock, it incorporates a precision quartz crystal oscillator that serves as a local reference in the absence of the LHC clock. The 40 MHz clock phase adjustment takes advantage of the short clock period of the serial bit clock to be implemented: as both phases of the clock are used, a phase adjustment resolution of 142 ps is achieved.

3.3 Radiation-induced errors characteristics

This section is aimed to study the characteristics of SEU-induced errors on the photodiode. In section 3.3.1 the calculation of a link bit error rate is presented for the case without radiation, while in section 3.3.2 the effects of the SEU-induced current in the photodiode are taken into account. As experimental data was unavailable at the time of writing, an estimation of the number of errors to be expected on the link due to radiation is performed in section 3.3.3 based on data reported in literature.

3.3.1 Estimation of error probability in an optical link

The calculation of the bit error rate of the system can be performed by studying the time-varying current generated at the receiver.

The receiver can be represented as the series of blocks reported in Figure 19 [Car86]. The photo-current signal x(t), with added channel and receiver noise, is first amplified by a factor A (here A = 1 for simplicity) and then low-pass filtered to reduce the noise bandwidth. The filtered signal y(t) is then sampled at bit frequency ($t = t_k$, k integer) with a "sample & hold" circuit and the obtained sampled values $y(t_k)$ are compared to a decision threshold. These values $y(t_k)$ fluctuate around an average value of I_0 or I_1 depending on whether the transmitted bit corresponds to a logical 0 or 1.

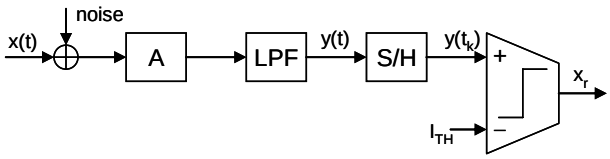

Figure 19: Receiver block diagram. The transmitted signal is summed to noise (from channel and receiver, reported to the input) before being received at the other side of the link. In the receiver the signal is amplified and filtered, sampled and compared to a threshold value for quantization.

The comparator examines the difference between the sampled data $y(t_k)$ and a threshold value I_{TH}. If the difference is positive, the received bit x_r is considered a logical "1", if negative, a "0". An error occurs if, due to noise, $y(t_k) > I_{TH}$ for a transmitted logical "0", or if $y(t_k) < I_{TH}$ for a transmitted logical "1".

The error probability is defined as $BER = P_1 P(0/1) + P_0 P(1/0)$, where P_0 and P_1 are the probabilities of transmission of a logical "0" or "1", and P(0/1) and P(1/0) are the conditional probabilities of deciding zero when "1" is transmitted, and "1" when "0" is transmitted, respectively. If zeros and ones are transmitted with equiprobable statistics, i.e. $P_0 = P_1$, the BER is: $BER = (P(0/1) + P(1/0))/2$.

P(0/1) and P(1/0) depend on the statistics of the noise sources responsible for the current fluctuations. Receiver noise is dominated by thermal and shot noise (in the case of p-i-n detectors [Agr05]), this noise is well described by Gaussian statistics with zero mean and variance σ^2. The probability density functions then take the form $p_N(n)$, which has even symmetry:

$$p_N(n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{n^2}{2\sigma^2}}; \quad Q(k) \doteq \frac{1}{\sqrt{2\pi}} \int_k^{\infty} e^{-\frac{\lambda^2}{2}} d\lambda$$

In the above, Q(k) is the probability for a Gaussian random variable with mean m and variance σ^2 to have an observed value greater than $m + k\sigma$.

Under these hypotheses, the optimum positioning of the decision threshold is midway between I_0 and I_1 [Car86]: $I_{TH} = (I_0 + I_1)/2$.

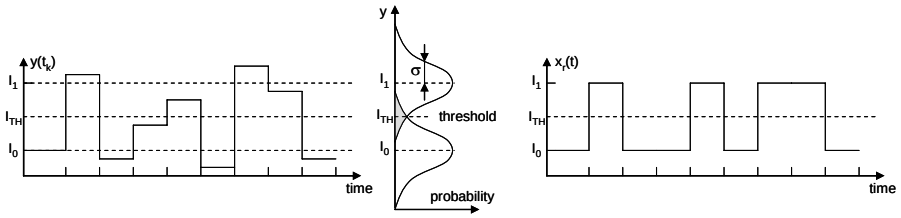

Figure 20: The signal $y(t_k)$ is the received signal after filtering and sampling, x_r is the digitized data stream after the decision block. On the right, the conditional PDFs (not to scale) of receving a certain amplitude value given the transmission of a logical "0" or "1". The shaded area corresponds to the probability of a bit error.

As illustrated in Figure 20, the probability of committing a decision error is equivalent to the area of the shaded region. This area grows with the RMS value of the noise σ in the conditional PDFs and diminishes with the difference in signal amplitude $(I_1 - I_0)$. Thus, the bigger the SNR ratio, which is proportional to $(I_1 - I_0) / \sigma$, the smaller the error probability. In particular, as the noise is Gaussian, the error probability can be expressed as:

$$P(1/0) = \int_{I_{TH}}^{\infty} p_N(y - I_0)dy = \int_{I_{TH}-I_0}^{\infty} p_N(z)dz = Q\left(\frac{I_{TH} - I_0}{\sigma}\right) = Q\left(\frac{I_1 - I_0}{2\sigma}\right)$$

$$P(0/1) = \int_{-\infty}^{I_{TH}} p_N(y - I_1)dy = \int_{I_1+(I_1-I_{TH})}^{\infty} p_N(y - I_1)dy = \int_{I_1-I_{TH}}^{\infty} p_N(z)dz =$$

$$= Q\left(\frac{I_1 - I_{TH}}{\sigma}\right) = Q\left(\frac{I_1 - I_0}{2\sigma}\right)$$

where the last equality in both lines holds because $I_{TH} = (I_0 + I_1)/2$.

3.3.2 Conditional error probability given a rad-induced current

A particle hit on the photodiode induces a current which affects the decoding decision, in particular if the hit happens shortly before the decision sampling instant and if the particle deposits a sufficient amount of energy. In the following, a calculation of the error probability in case of particle hit is carried out. As these calculations assume a particle strike has happened, what is evaluated is in fact a conditional error probability: P_{e0} (P_{e1}) is the probability that a decoding error happens on the transmission of a zero (one) given that a particle has hit the photodiode.

The following calculations are based on the assumption that radiation hits on the photodiode happen rarely enough not to affect the evaluation of the decision threshold, which holds as in $I_{TH} = (I_0 + I_1)/2$.

In case the transmitted signal is a logic "low", then the photo-induced signal current is close to zero ($I_0 \sim 0$); the total received signal formed by photo-induced current and additive white noise is detected as a logical low if it is below threshold at the sampling instant. The radiation-induced current I_{RAD} is an additional component which effectively lowers the amount of white noise that can be tolerated in the system in order for the total signal to remain below threshold.

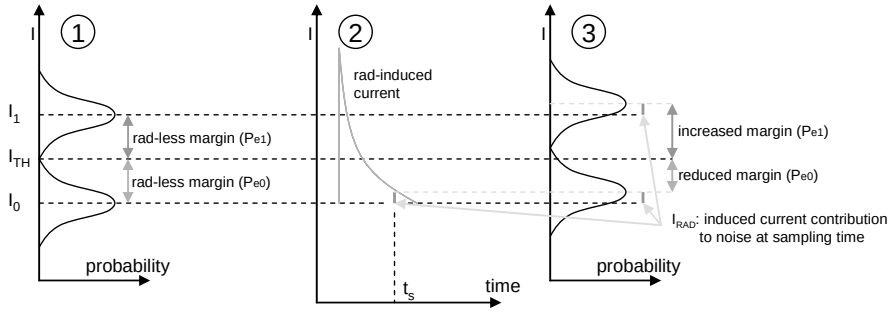

Figure 21: Variation of error probabilities with radiation-induced currents: 1) error probabilities in the absence of radiation; 2) radiation-induced current and radiation-induced component at sampling time t_s; 3) variation of error probabilities and noise margins in the case of radiation-induced current.

In case the transmitted signal is a logic "high", then the photo-induced signal is approximately I_1, and is detected correctly if its sum with the receiver noise at the sampling instant is above the threshold I_{TH}. In this case the radiation-induced current I_{RAD} is also an additional component to the

calculation, but it raises the amount of noise that the system can tolerate before committing a detection error.

Consequently, the radiation-induced current at the sampling instant is added to the noise margin in P_{e1}, but gets subtracted to the noise margin in P_{e0} (see Figure 21, 3). The conditioned error probabilities, thus, are not symmetrical as in the case of absence of radiation, and what was previously calculated for P(0/1) and P(1/0) varies according to the following:

$$P_{e0} = Q\left(\frac{I_{TH} - I_0 - I_{RAD}}{\sigma}\right) = Q\left(\frac{I_1 - I_0 - 2I_{RAD}}{2\sigma}\right),$$

$$P_{e1} = Q\left(\frac{I_1 - I_{TH} + I_{RAD}}{\sigma}\right) = Q\left(\frac{I_1 - I_0 + 2I_{RAD}}{2\sigma}\right).$$

In the second expression I_{TH} is substituted by $I_{TH} = (I_0 + I_1)/2$, which still holds due to the rarity of the SEU events. Noted, though, that I_{TH} is not optimally placed in case of particle strike as it had been in case of absence of radiation (see Figure 21, 3).

The radiation-induced current $I_{RAD} = I_{RAD}(t)$ has to be evaluated. I_{RAD} is generated by the deposited charge Q induced by the particle hit in the photodiode. The charge Q can be approximated as being deposited instantly due to the fact that the deposition process is much faster than the photodiode response and electrical response of the receiver. If the whole system between the photodiode and the decision circuit is simplified to be a first order system, then I_{RAD} is shaped as follows (see Figure 21, 2):

$$I_{RAD}(t) \propto \delta(t) * \left(e^{-t/\tau}u(t)\right) \quad \text{so that} \quad I_{RAD}(t) = \frac{Q}{\tau}e^{-\frac{t}{\tau}}u(t),$$

where I_{RAD} in the second expression is normalized such that the integral over time gives a total deposited charge Q. Thus, while the total charge Q is constant, the current I_{RAD} dies away faster (or slower) according to the τ = RC time constant of the front end circuit.

An important parameter in the evaluation of the probabilities P_{e0} and P_{e1} is the arrival time of the particle relative to the sampling instant. As illustrated in Figure 22, the particle-induced current can originate at different times, i.e. t_1, t_2 or t_3, depending on the particle time of arrival. The

closer the arrival time t_i to the sampling time t = 0, the higher the current I_{RAD} at the sampling time, and the more likely it is to generate a bit error.

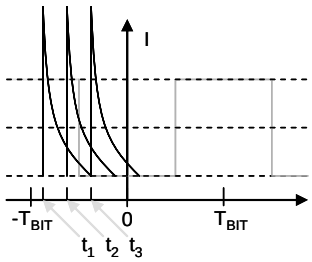

Figure 22: Particle-induced currents at different arrival times t_1, t_2 or t_3. The instant t = 0 is defined as the sampling time.

Concentrating on P_{e0}, the probability for one particle hit to corrupt successive bits can be evaluated over successive time intervals of length T_{bit}, which is equivalent to evaluating how likely it is for the particle hit to corrupt the first bit after the strike, the second, the third and so on.

$P_{e0,1}$ is the probability that the SEU corrupts the first bit after the impact, and it can be calculated by averaging over possible arrival times within the period ($-T_{bit}$, 0) having fixed the sampling instant to t = 0 [Bri93].

$$P_{e0,1} = \frac{1}{T_{bit}} \int_0^{T_{bit}} Q\left(\frac{I_1 - I_0 - 2I_{RAD}(t)}{2\sigma} \right) dt = \frac{1}{T_{bit}} \int_0^{T_{bit}} Q\left(\frac{1}{2\sigma} \left(I_1 - I_0 - 2\frac{Q}{\tau} e^{-\frac{(t)}{\tau}} u(t) \right) \right) dt$$

$P_{e0,2}$ is the probability that the SEU corrupts the second bit decision after the impact. For the calculation, while the sampling instant is kept constant (t = 0), the particle arrival time is moved back in time and averaged over ($-2T_{bit}$, $-T_{bit}$). Similarly, $P_{e0,3}$ is the conditional error probability affected by impacts within ($-3T_{bit}$, $-2T_{bit}$).

$$P_{e0,2} = \frac{1}{T_{bit}} \int_0^{T_{bit}} Q\left(\frac{1}{2\sigma} \left(I_1 - I_0 - 2\frac{Q}{\tau} e^{-\frac{(t+T_{bit})}{\tau}} u(t+T_{bit}) \right) \right) dt$$

$$P_{e0,3} = \frac{1}{T_{bit}} \int_0^{T_{bit}} Q\left(\frac{1}{2\sigma} \left(I_{TH} - I_0 - 2\frac{Q}{\tau} e^{-\frac{(t+2T_{bit})}{\tau}} u(t+2T_{bit}) \right) \right) dt .$$

The expressions above can be rewritten by considering $I_0 \sim 0$ and factoring the SNR term together ($I_1 / 2\sigma$), leaving the effects of the radiation induced current as a normalization on the signal current. For example $P_{e0,1}$

takes the following form, where it becomes evident that the radiation induced current is a worsening factor for the BER, as it diminishes the SNR:

$$P_{e0,1} = \frac{1}{T_{bit}} \int_0^{T_{bit}} Q \left(\frac{I_1 - I_0 - 2\frac{Q}{\tau} e^{-\frac{(t)}{\tau}} u(t)}{2\sigma} \right) dt \cong \frac{1}{T_{bit}} \int_0^{T_{bit}} Q \left(\frac{I_1}{2\sigma} \left(1 - \frac{2\frac{Q}{\tau} e^{-\frac{(t)}{\tau}} u(t)}{I_1} \right) \right) dt$$

The normalization factor I_{factor} can be studied in order to understand the implication on SNR of the radiation induced current:

$$I_{factor} \overset{def}{=} \left(1 - \frac{2\frac{Q}{\tau} e^{-\frac{(t)}{\tau}} u(t)}{I_1} \right)$$

The average ionization charge Q is estimated to be of the order of 10^5 electron/hole pairs per event, generated by protons ionizing an InGaAs photodiode's sensitive layer [Fac01, Mar94]. The factor of 10 in increased luminosity for the SLHC machine [DeR06] will effectively increase the number of events to be studied, and consequently the rate of SEUs, but will not increase the energy of the single particles, so the value for $Q = 10^5$ electron/hole pairs per event is still valid in this calculation.

The time constant $\tau = RC$ of the front end circuit is usually set for NRZ receivers so that the receiver 3-dB bandwidth f_{3dB} (Hz) is approximately two thirds the line frequency f_{bit} [Sac05]:

$$\tau = R \cdot C = \frac{1}{2\pi\ f_{3dB}} = \frac{1}{2\pi \frac{2}{3} f_{bit}} \approx \frac{1}{4} T_{bit}$$

The current signal I_1 at the photodiode is the ratio between the number of optically-generated electron-hole pairs N_e and the bit period T_{bit}. In turn, the number of electron-hole pairs N_e generated per μW of incident power at the line data rate f_{bit} is estimated through the ratio between the incident optical energy divided by the energy of a photon at the chosen wavelength (1310 nm).

$$N_e = \frac{E_{optical}}{E_{photon}} = \frac{P_{\mu W} T_{bit}}{h\upsilon} = \frac{P_{\mu W} T_{bit} \lambda}{hc} = \frac{10^{-6}\,\mu W \cdot 284\,ps \cdot 1310\,nm}{6.626 \cdot 10^{-34}\,J \cdot s \cdot 3 \cdot 10^{8}\,\frac{m}{s}} \approx 1900\ pairs/\mu W$$

The factor I_{factor} can then be studied for different incident optical powers and for the different probabilities $P_{e0,i}$. The factor I_{factor} is negative in the first T_{bit}, ($P_{e0,1}$) meaning that a bit error is certain if the SEU happens to fall on a transmitted logical "0" for the optical power levels considered. I_{factor} is plotted in Figure 23 for four different power levels and for $P_{e0,2}$ through $P_{e0,5}$, that is for two to five T_{bit} after impact. For higher optical powers, I_1 is relatively higher and thus I_{factor} is closer to unity.

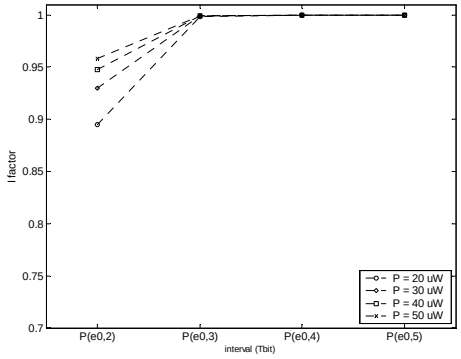

Figure 23: I_{factor} for different power levels and timing intervals (i.e. $P_{e0,1}$,…, $P_{e0,5}$).

Only the second bit period risks being affected by the particle hit, while the other bit decisions are practically unaffected by the strike as $I_{factor} \sim 1$. From what is shown in Figure 23, it can be deduced that a significant impact of the SEU hit on the decisions can be observed only in the first and second bit period after the hit, while in the next bit periods the effect is practically null.

3.3.3 Link errors

A different approach can be taken by estimating the link BER by similarity with other reported error cross-section studies on optic links to be used in HEP or satellite environments.

It is widely established [Fac01, Mar01, Mar04] that the error cross section is dominated by the errors on the photodiode. The cross section decreases with an increase on the level of incident optical power and it

increases with data rate. In particular it has a peak when the angle of incidence between the particles and the photodiode is 0 degrees (i.e. grazing angle), as the hitting particle passes through the sensitive material for a longer time interval; this allows the deposition of a higher amount of energy. An example of dependence of the error cross section on the angle of incidence and optical power is shown in Figure 24 for the 10 Gb/s link reported in [Mar04]. A lower amount of incident optical power makes the receiver more sensitive to SEUs (as already discussed), while highlighting the importance of I_1 in the evaluation of I_{factor}. Grazing angles of incidence imply up to two-orders of magnitude larger cross-section.

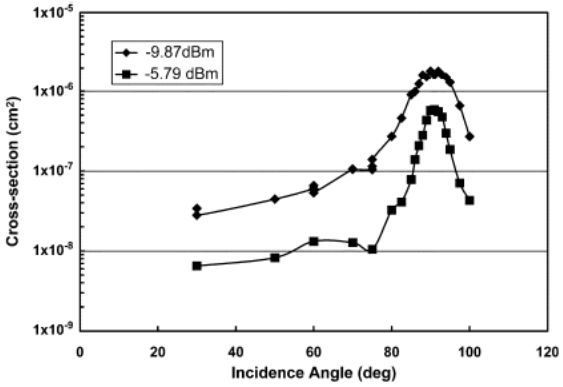

Figure 24: The 10 Gb/s receiver in [Mar04] showed much higher sensitivity to bit errors from protons when incident near the plane of the receiver's p-i-n diode.

Taking under consideration a total flux in the order of $1.5 \cdot 10^6$ part. s^{-1} cm^{-2} in the inner parts of the detectors (e.g., the Outer Tracker of CMS, [CMS98], multiplied by a factor 10 due to SLHC operation) and reported worst case error cross-sections of 10^{-6} cm^2 [Mar01, Mar04], an error rate of about of 1.5 errors/s can be estimated. This translates into a bit error rate of the order of $4 \cdot 10^{-10}$ for the link under study.

$$P_e = BER = \frac{flux \cdot error\ cross\ section}{link\ rate} = \frac{1.5 \cdot 10^6 \cdot 10^{-6}}{3.52 \cdot 10^9} = 4.26 \cdot 10^{-10}$$

According to this bit error rate value, the probability of two independent error events falling in the same frame of total length $N_{tot} = 88$ bits (Bernoulli trials, [Pap91]) is:

$$P(2\ indip.\ err.) = \frac{1}{2} N_{tot} \cdot N_{tot} \cdot P_e^2 = 6.95 \cdot 10^{-16}$$

This result leads to choosing for the line code an error correction capability of at least one error per frame so that the BER is P(2 indip. err.) ~ 10^{-16} for the link protected by the error correction code.

Additionally, in light of the results derived in section 3.3.2, an error generated by an SEU event might extend over two consecutive bits, so that the error correction scheme has to take also this characteristic into account.

3.4 Existing line codes and error correction

Existing line codes are presented in section 2.1.2. They are created in order to provide the encoder output data stream with properties such as data transitions abundance and DC-balance, but they do not in general have error correction capability. They often provide some error detection capability, but this is not sufficient in broadcast or unilateral transmissions.

In order to use commercial line codes in the VDB link, which is affected by photodiode SEUs, an error correction scheme has to be foreseen and integrated in the system. By nature of line coding, its encoder and decoder are placed next to the communication channel, and consequently the error correction blocks are to be placed externally to line coding (see Figure 25).

Figure 25: Concatenation of line coding and error correction.

Errors on the channel are modified according to the line code decoding, so that the error correction scheme has to be tailored to fit the combination of the chosen commercial line code and the most common errors on the channel. In fact, in general line errors get multiplied by line decoding, and thus a more complex error correction code is needed in case of error correction applied after line decoding instead of directly after the channel. These ideas are explored in the following for the cases of the popular line codes 8b/10b and 64b/66b.

3.4.1 8b/10b

The 8b/10b code [Wid83] is presented in section 2.1.2.2. The choice of the code mapping table is based on the optimization of characteristics as

DC-balance and number of transitions. This results in error multiplication, though, when an error corrupts the line coded block.

In Table 2, examples of single errors corrupting line coded blocks are reported. For the messages in the first column, the blocks in the second column are the corresponding coded blocks according to the code map. Given the error patterns in the third column, the received block is shown in the fourth column and the decoded block in the fifth. The effect of the error pattern can be evaluated by computing the difference between the first and fifth columns: two or three bit are often wrong and depending on their relative position they can extend up to a 5-bit burst.

Table 2: Examples from 8b/10b coding table. Depending on the error pattern that is summed to the transmitted message, different words are decoded. Single-bit channel errors are converted into bursts of length up to 5 bits.

Message	Transmitted	Error pattern	Received	Decoded	Burst length
00000	011000	100000	111000	11100	3
"	"	000100	011100	01110	3
"	"	000010	011010	01101	4
"	"	000001	011001	01100	2
01000	010010	100000	110010	11001	5
"	"	000100	010110	01011	2
"	"	001000	011010	01101	3
000	0100	1000	1100	110	2
"	"	0010	0110	011	2
"	"	0001	0101	010	1
010	0101	0010	0111	111	3
"	"	1000	1101	001	2

Concatenating an error correcting code with 8b/10 then requires the error correction scheme to be able to correct at least bursts of length 5 bits.

With an interleaved Hamming error correction scheme (see sections 4.1.1.1 and 4.3), a 64-bit data packet can be error protected and 8b/10b encoded with a total frame length of 120 bits for a 55% efficiency. Reed-Solomon codes (RS, see section 4.2) could be the answer to the burst error correction problem after 8b/10b, if the 8b/10b blocks or sub-blocks are mapped into RS symbols to optimize the error correction capability. A RS error correction code based on 5-bit RS symbols concatenated with 8b/10b can protect a 60-bit data packet with a total of 100 bits, for a 60% efficiency.

Better efficiency results are difficult to achieve mainly due to the fact that 8b/10b alone requires a bandwidth increment of 25%, and this is to be added to the redundancy required by the error correction capability.

3.4.2 64b/66b

The 64b/66b code is presented in 2.1.2.3 and is based on a 64-bit scrambled field paired with two additional bits per frame. The 64b/66b code itself does not provide any error correction capability. In the Ethernet stacked system, though, error control is provided in higher levels so that in fact the scrambler is chosen such that it does not impair that error control capability [Iee00].

Detailed discussion on scrambler follows, in section 5.1, with regards to the choice of a scrambler for the proposed line code. Here it is only pointed out that due to the scrambler implementation chosen for 64b/66b, each bit error on the scrambled line stream is multiplied to three errors in the descrambled stream, and the distance between the first and second error is 39 bits, while 19 bits separate the second error from the third one (this is due to the chosen scrambler polynomial, that is $x^{58}+x^{19}+x^{0}$). Consequently an error-correction capability of three errors per frame is required, when only a single channel error is present.

An error correction scheme designed for 64b/66b encoded links has been presented in [Raa05]. This scheme takes into account error multiplication by the scrambler while performing error correction, and performs single bit error correction and double-bit error detection. It has a 12.5% additional overhead and is characterized by a latency of 64 bytes.

3.5 Summary

This chapter presents the requirements of the VBD link. The TTC link is first introduced as one of VBD link's functions will be the upgrade of the TTC system. The VBD link itself is presented next, conceptually, for different functionalities and transceiver ASIC. The characteristics of the photodiode SEU errors are studied along with the possible use of commercial line codes.

Chapter 4

Introduction on error-correction theory

If a system has a certain Signal-to-Noise Ratio (SNR), which cannot be improved, and the resulting error rate is unacceptably high, or in a system like the VBD link, where SEUs on the photodiode cannot directly resolved, then system reliability must be strengthened via other means, i.e. error-control coding. Error-control coding improves the reliability of the transmission by systematically adding extra digits (also called *redundancy* digits) to the transmitted message. These extra digits do not convey any information by themselves, but they are chosen in such a way that they make it possible to detect or even correct errors in the regenerated message digits. The choice of the error-correcting or detecting code is the choice of how to associate a message with the corresponding extra digits so as to obtain data protection. The desired improvement in reliability is gained at the cost of reducing the message bit rate (or increasing the transmission bandwidth) and increasing transmitter and receiver complexity due to the need of encoder and decoder circuitry.

This chapter introduces the basic concepts of error correction theory (section 4.1) and Reed-Solomon (RS) codes (section 4.2). The technique of code interleaving is also briefly introduced (section 4.3).

4.1 Basics of Error Correction Theory

4.1.1 Forward Error Correction systems

Error control capability can be exploited in two different ways, depending on system architecture. Errors can be detected through an error

detecting code, and this is feasible when a two-way channel exists between source and destination and the receiver can request retransmission of information which is found to contain errors. This error control strategy is called Automatic Repeat reQuest (ARQ). However retransmission is impossible or impractical for many systems, as for example in the case of uni-directional links or broadcast transmissions. In these cases error control must take the form of Forward Error Correction (FEC) which uses an error-correcting code and allows the receiver to autonomously and correctly reconstruct the corrupted packet. ARQ systems are not explored further in this work due to the broadcast functionality required for the link under study (see section 3.2.2).

Two main families of error correction codes are in use today: convolutional codes and block codes [Lin04]. The encoder for convolutional codes has memory (implemented with sequential circuits), while the encoder for block codes is memoryless. The convolutional structure is especially well suited to space and satellite communication systems [Car86] that require very simple encoders and achieve high performance by sophisticated decoding methods.

On the contrary, a block code encoder divides the information sequence into message blocks of k information bits (for binary transmission) and transforms each message independently into an n-tuple of discrete symbols called a *codeword* (see Figure 26). The set of 2^k different possible codewords of length n is called an (n, k) block code. Each message is encoded independently and the encoder, being memoryless, can be implemented with a combinational logic circuit. Reed-Solomon codes are an example of block codes. Block codes are chosen for the VBD link because the framed structure is well-suited to the synchronization functionality of the link.

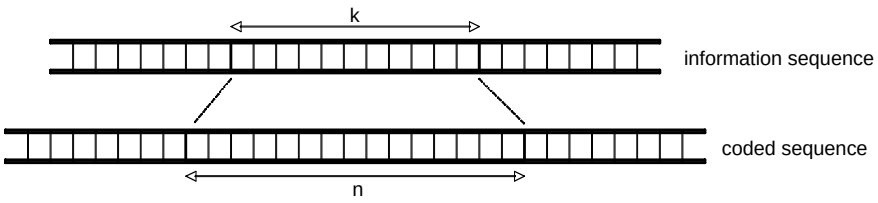

Figure 26: In block codes, k information symbols correspond to n code symbols. The correspondence is independent of previous and successive codewords.

Figure 27 shows am FEC system block diagram [Car86]. The source input message has rate r_b (in bits/s) and is fed to the encoder block. A binary (n, k) error-correcting block code is used, so that (n-k) redundant bits are added per message block and the code rate is $R_c = k / n$. The encoded stream is then transmitted over the channel at a rate $r = r_b / R_c = r_b\, n / k$. Noise is added to the signal when passing through the channel before it is received and regenerated, possibly altering the signal. At the decoder, each received block is inspected for errors. If the number of errors is small compared to the code error correction capability, the received block does not in general belong to the set of possibly transmitted words (2^k out of 2^n). In this case the "closest" codeword is assumed to have been transmitted and passed on as output message. This technique is called *maximum likelihood decoding* and is based on the assumption that a bigger number of errors in the same block is less likely to have happened than a smaller one.

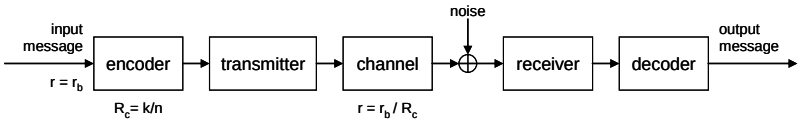

Figure 27: Scheme of a FEC system.

The channel rate r is higher than the source rate r_b as the code rate R_c is lower than unity: the increase in rate is at the cost of bandwidth for a more reliable transmission. The code rate R_c is a measure of the code efficiency. The smaller the redundancy, the smaller the increase in bandwidth, the more efficient the code.

The previously introduced concept of "closeness" between block words is quantified by a parameter called *Hamming distance* between two words. An arbitrary n-bit codeword can be visualized in an n-dimensional space as a vector whose elements or coordinates equal the bits in the codeword. As an example, $\boldsymbol{x} = (x_0, x_1, x_2, \ldots, x_{n-1})$ is an n-bit vector and x_i (for $0 \leq i < n$) are its coordinates. The Hamming distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as the number of different elements between $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \left| \left\{ i \middle| x_i \neq y_i \right\} \text{ with } 0 \leq i < n \right|$$

The *minimum distance* of a code is defined as the smallest distance between two distinct codewords. This parameter is closely linked to the error correction/detection capabilities of the code itself: if the minimum

distance of the code is d_{min}, then the code can detect up to (d_{min}-1) errors or correct up to $\lfloor (d_{min}-1)/2 \rfloor$ [Lin04]. The fundamental strategy of block coding is then to choose the 2^k n-symbol code vectors such that the minimum distance is as large as possible.

Various bounds have been proven so far in literature [Lin04] linking the code parameters n and k to the minimum achievable distance. In particular the Singleton bound for linear (n, k, d_{min}) codes states that the minimum distance of an (n, k) block code is upper-bounded: $d_{min} \leq n - k + 1$ [Byl80]. Codes for which the equality sign holds, that is for which the minimum distance is one greater than the number of parity check symbols, are called *Maximum Distance Separable* (MDS) codes. Reed-Solomon codes are an example of MDS codes [Ber68, Lin04].

If transmission errors occur randomly and independently with probability $P_e = \alpha$, then the binomial frequency function gives the probability of *i* errors in an n-bit codeword as [Car86]:

$$P(i,n) = \binom{n}{i}\alpha^i(1-\alpha)^{n-i} \approx \binom{n}{i}\alpha^i \quad \text{if } \alpha << 1$$

If P_{we} is the word error probability, and the code corrects at least up to *t* errors per word, then:

$$P_{we} \leq \sum_{i=t+1}^{n} P(i,n) \approx P(t+1,n) = \binom{n}{t+1}\alpha^{t+1}(1-\alpha)^{n-t+1} \approx \binom{n}{t+1}\alpha^{t+1}$$

where the approximations hold if $\alpha << 1$ and because the probability to obtain (t + 1) errors is much higher than the probability for more than (t + 1) errors to occur. If the number of errors is higher than the code error correction capability, then the decoder probably corrects the wrong symbols, as the closest to the received block is not the transmitted code word. So, on average, there will be $\frac{k}{n}(2t+1)$ message bit errors per uncorrected word: k/n as the remaining will be in the check bits, and (t+1) original errors added to t erroneously corrected errors.

If Nk bits are transmitted in N >> 1 words, then the total number of erroneous message bits at the output is $\frac{k}{n}(2t+1)NP_{we}$. Thus:

$$P_{be} = \frac{\#errbits}{\#transmbits} = \frac{\frac{k}{n}(2t+1)NP_{we}}{Nk} = \frac{2t+1}{n}P_{we} \approx \frac{2t+1}{n}\binom{n}{t+1}\alpha^{t+1}$$

For example, in the case of Gaussian-distributed noise and optimized transmission system [Car86], the Q function can be used to express the transmission error probability as a function of γ_b (the system SNR at the receiver):

$$\alpha = Q\left(\sqrt{2R_c\gamma_b}\right) \approx \left(4\pi R_c\gamma_b\right)^{-1/2} e^{-R_c\gamma_b} \text{ for } R_c\gamma_b \geq 5$$

$$P_{be} = \frac{(2t+1)}{n}\binom{n}{t+1}\left[Q\left(\sqrt{2R_c\gamma_b}\right)\right]^{t+1} \propto e^{-(t+1)R_c\gamma_b}$$

$$P_{ube} = Q\left(\sqrt{2\gamma_b}\right) \propto e^{-\gamma_b}$$

A comparison between the last two equations stresses the importance of the parameters $t = (d_{min}-1)/2$ and $R_c = k/n$. The added complexity of an FEC system is justified provided that t and R_c yield a value of bit error rate P_{be} significantly lower than the uncorrected system error probability P_{ube}. The exponential approximation shows that this essentially requires $(t+1)\,R_c > 1$. Hence, a code that only corrects single or double errors should have a relatively high code rate, while more powerful codes may succeed despite lower code rates. The channel SNR γ_b also plays a role in the comparison.

4.1.1.1 Examples of simple codes

Three examples of simple error control codes are reported here: repetition codes, parity check codes and Hamming codes. A summary of the parameters of the codes is reported in Table 3 for comparison.

A binary n-bit *repetition* code is a code in which a single bit of information is repeated n times identically to build an n-bit codeword. Any transmission error alters the received codeword and if the number of errors is less than n bits, the codeword is detected as corrupted. If the number of errors is less than half the repetition length and maximum likelihood decoding is applied, a majority voting can reconstruct the original information. This capability is gained at the cost of reducing the message bit

rate by a factor of 1/n, which makes the code very inefficient. Odd-length repetition codes are a trivial example of both MDS codes.

A *parity check* code is much more efficient but has only single error-detection capability. The parity of a binary word is said to be even when the word contains an even number of ones, while odd parity means that there are an odd number of 1s. The codewords for a parity-check code are constructed with (n - 1) message bits and one check bit chosen such that all codewords have the same parity. Thus, when a received codeword has odd parity, a transmission error is detected. Clearly, only odd numbers of errors can be detected, and no error correction can be performed as the detected error cannot be located.

A *Hamming* code is an (n, k) linear block code with $q > 2$ check bits, total block length $n = 2^q - 1$ and number of information bits $k = n - q$. The minimum distance is 3 (independently of q), thus Hamming codes can be used for single error correction or double error detection. The efficiency increases with the length of the block, as the number of check bits increases only logarithmically with the total number of bits. Hamming codes can be combined with a parity check code in order to correct 1-bit errors and additionally detect 2-bit errors. This can be done simply by appending to the Hamming codeword a single parity bit.

Table 3: Basic coding techniques and relative parameters: n, k, minimum distance d_{min} and code rate R_c (q integer).

Code	n	k	d_{min}	R_c
repetition	n	1	n	1/n
parity	n	n-1	2	(n-1)/n
Hamming	2^q-1	n-q	3	1-q/(2^q-1)

4.1.2 Linear block codes

A code is called a *linear* code if it includes the all-zero vector and if the sum of any two code vectors produces another vector that belongs to the code. it should be noted that the sum of two vectors is defined as the element-by-element module-2 addition (in binary transmission); as a consequence, addition is equivalent to subtraction. For an (n, k) linear block code it is possible to find k linearly independent codewords ($\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{k-1}$) such that every codeword $\boldsymbol{v} = (v_0, v_1, \ldots, v_{n-k})$ is a linear combination of these k codewords [Lin04]:

$$\boldsymbol{v} = u_0\boldsymbol{g}_0 + u_1\boldsymbol{g}_1 + \ldots + u_{k-1}\boldsymbol{g}_{k-1} \text{ where } u_i \in \{0,1\} \text{ for } 0 \le i < k$$

The k linearly independent codewords can be arranged as the rows of a $k \times n$ matrix:

$$G = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix}$$

so that $\boldsymbol{v} = \boldsymbol{u}$ G if $\boldsymbol{u} = (u_0, u_1, \ldots, u_{k-1})$

As the rows of G generate the (n, k) linear code, G is called *generator* matrix for the code.

A *systematic* block code consists of vectors whose last k elements (or first k elements) are identical to the message bits, the remaining n-k elements being check bits. For a systematic code the generator matrix takes the following form, which is a combination of a $k \times (n-k)$ P matrix and a $k \times k$ identity matrix I_k:

$$G = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \left[\begin{array}{cccc|cccc} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} & 1 & 0 & \cdots & 0 \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 0 & 0 & \cdots & 1 \end{array}\right] = [PI_k]$$

For a systematic code, the codeword $\boldsymbol{v} = (v_0, v_1, \ldots, v_{n-1})$ takes the form:

$v_{n-k+i} = u_i$ for $0 \leq i < k$ and $v_j = u_0 p_{0,j} + u_1 p_{1,j} + \ldots + u_{k-1} p_{k-1,j}$ for $0 \leq j < n-k$

For any linear code a generator matrix G exists. Together with G, another matrix can be defined, H, so that any vector in the row space of G is orthogonal to the rows of H, and any vector that is orthogonal to the rows of H is in the row space of G. H is an $(n-k) \times n$ matrix and $H G^T = 0$. In other words, the linear code can now be defined as follows: an n-tuple v is a codeword in the code generated by G if and only if $\mathbf{v} H^T = 0$. The matrix H is called the *parity-check* matrix of the code, and for a systematic code takes the form $H = [I_{n-k} P^T]$.

If $\boldsymbol{v}$ is the transmitted codeword and $\boldsymbol{r} = (r_0, r_1, \ldots, r_{n-1})$ is the received codeword, then $\boldsymbol{r} = \boldsymbol{v} + \boldsymbol{e}$, where the error vector (or error pattern) is $\boldsymbol{e} = \boldsymbol{r} + \boldsymbol{v}$; and $e_i = 0$ if $r_i = v_i$; $e_i = 1$ if r_i and v_i differ (for $0 \leq i < n$).

The syndrome $\boldsymbol{s}$ of $\boldsymbol{r}$ is defined as follows: $\boldsymbol{s} = \boldsymbol{r}\,H^T = (s_0, s_1, \ldots, s_{n-k-1})$. If the syndrome is different from 0, then the received block is not a codeword as the syndrome $\boldsymbol{s}$ equals 0 if and only if $\boldsymbol{r}$ is a codeword, and transmission errors have occurred. It should be noted that if the error pattern is a codeword itself, then that error is undetectable. There are $2^k - 1$ undetectable error patterns, as there are $2^k - 1$ codewords which are different from the null word. The syndrome $\boldsymbol{s}$ computed from the received vector $\boldsymbol{r}$ depends only on the error pattern $\boldsymbol{e}$ and not on the transmitted codeword $\boldsymbol{v}$: $\boldsymbol{s} = \boldsymbol{r}\,H^T = (\boldsymbol{v} + \boldsymbol{e})\,H^T = \boldsymbol{v}\,H^T + \boldsymbol{e}\,H^T = \boldsymbol{e}\,H^T$.

The syndrome can also be seen as the vector sum of the received parity digits and the parity check digits recomputed from the received information digits. If the two sets are the same, their sum equals zero, while if they are different in some positions, the sum is different from zero and transmission errors are detected. The syndrome evaluation is in fact the error detection step. As the syndrome calculation is equivalent to computing parity check digits, this operation can be performed by a circuit similar to that used in encoding. Associating the syndromes with some error pattern and correcting the received word for this pattern is what is called “error correction”.

There are 2^k error patterns that result in the same syndrome: for any error pattern, its vector sum with any of $(2^k - 1)$ codeword gives a different error pattern but the same syndromes (as codewords have syndromes equal to zero). This means that the decoder has to determine the true error vector from a set of 2^k candidates. To minimize the probability of decoding error, the most probable error pattern that satisfies the syndrome constraints is chosen as the true error vector. If the channel is a binary symmetric channel, the most probable error pattern is the one that has the smallest weight (smallest number of nonzero digits).

4.1.3 Cyclic codes

A linear code is *cyclic* if every cyclic shift of a code vector is another code vector in the code. As an example, if $\boldsymbol{c} = (c_0, c_1, c_2, \ldots, c_{n-2}, c_{n-1})$ is a code vector, then all other cyclically shifted words are codewords, i.e. $\boldsymbol{c}' = (c_1, c_2, c_3, \ldots, c_{n-1}, c_0)$. Codewords can also be represented in polynomial notation: $\mathbf{c}(x) = c_0 + c_1 x + \ldots + c_{n-1}x^{n-1}$ and $\mathbf{c'}(x) = c_1 + c_2 x + \ldots + c_0 x^{n-1}$. Cyclic codes are an important subclass of linear block codes as the encoding (and part of the decoding) process can be implemented easily by employing shift registers with feedback connections.

It can be proven [Lin04] that if a code is cyclic, then it can be uniquely defined using a *generator polynomial*, a polynomial of degree n - k which takes the form:

$\mathbf{g}(x) = 1 + g_1 x + g_2 x^2 + \ldots + g_{n-k} x^{n-k}$.

A binary polynomial of degree n-1 or less is a code polynomial (belonging to the set generated by $\mathbf{g}(x)$) if and only if it is a multiple of $\mathbf{g}(x)$ [Lin04].

Given the generator polynomial $\mathbf{g}(x)$ of a cyclic code, the code can be put into systematic form. If the message $\boldsymbol{u} = (u_0, u_1, \ldots, u_{k-1})$ is to be encoded, and the corresponding message polynomial is $\mathbf{u}(x) = u_0 + u_1 x + \ldots + u_{k-1} x^{k-1}$, then:

$x^{n-k}\, \mathbf{u}(x) = \mathbf{a}(x)\, \mathbf{g}(x) + \mathbf{b}(x)$

where $\mathbf{a}(x)$ and $\mathbf{b}(x)$ are the quotient and the remainder respectively. Equivalently:

$\mathbf{b}(x) + x^{n-k}\, \mathbf{u}(x) = \mathbf{a}(x)\, \mathbf{g}(x)$

In the second expression the left-hand side is a multiple of g(x), and therefore a code polynomial belonging to the cyclic code generated by $\mathbf{g}(x)$. Moreover:

$\mathbf{b}(x)+x^{n-k}\mathbf{u}(x) = b_0 + b_1 x + \ldots + b_{n-k-1} x^{n-k-1} + u_0 x_{n-k} + u_1 x^{n-k+1} + \ldots + u_{k-1} x^{n-1}$

This corresponds to the codeword $(b_0, b_1, \ldots, b_{n-k-1}, u_0, u_1, \ldots, u_{k-1})$ and this means that the code is systematic.

In summary, the steps that have been performed are:

- multiplication of the message $\mathbf{u}(x)$ by x^{n-k};
- division of $\mathbf{u}(x)\, x^{n-k}$ by $\mathbf{g}(x)$ to obtain the remainder $\mathbf{b}(x)$;
- creation of the codeword as $\mathbf{b}(x) + \mathbf{u}(x)\, x^{n-k}$.

The generator matrix G for the code can be obtained from the generator polynomial $\mathbf{g}(x)$:

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & g_3 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & \cdot & g_{n-k} & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & \cdots & \cdot & \cdot & g_{n-k} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & & & & & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & g_{n-k} \end{bmatrix} =$$

$$= \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & \cdots & b_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ b_{1,0} & b_{1,1} & b_{1,2} & \cdots & b_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ b_{2,0} & b_{2,1} & b_{2,2} & \cdots & b_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\ b_{k-1,0} & b_{k-1,1} & b_{k-1,2} & \cdots & b_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

where the second matrix generates a systematic code.

This proves that the encoding process can be implemented by performing polynomial divisions. One very common implementation uses a feedback shift register with connections, as illustrated in Figure 28.

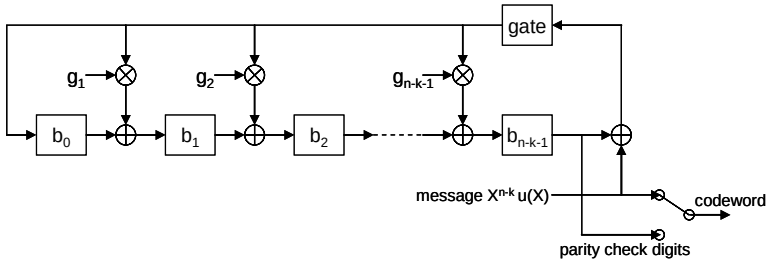

Figure 28: Encoding circuit for an (n, k) cyclic code [Lin04, pag 147] in systematic form. The generator polynomial is in the form: $g(x) = 1 + g_1x^2 + ... + g_{n-k-1}x^{n-k-1} + x^{n-k}$.

The received codeword might have an error pattern added to the transmitted codeword **v**(x): **r**(x) = **v**(x) + **e**(x). If **e**(x) = 0, then dividing **r**(x) by **g**(x) gives a null remainder, as **r**(x) = **v**(x) = **a**(x) **g**(x). If **e**(x) is different from zero, then the remainder of the division by **g**(x) is equal to the syndromes: **r**(x) = **v**(x) + **e**(x) = **a**(x) **g**(x) + **s**(x) [Lin04, pp.150]. As it is the syndrome polynomial (the remainder of a division by **g**(x)), its degree is (n – k – 1) or less, as expected. It is also confirmed that the syndrome polynomial only depends on the error pattern, and not on the transmitted codeword.

As stated previously, the syndrome calculation can be implemented through similar circuitry to that used for encoding. Different circuit

implementations are presented in [Lin04], for example the blocks shown in Figure 29 and Figure 30, which differ in the received word input. Figure 29 in particular is the same circuit implementation as shown in Figure 28 for encoding.

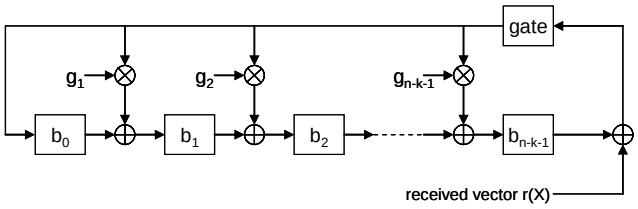

Figure 29: An (n, k)-stage syndrome circuit with input from the right end [Lin04, pag.153].

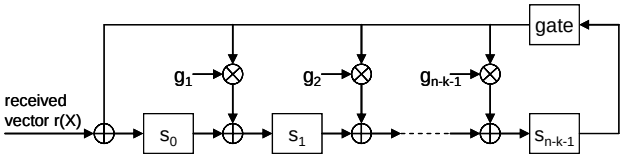

Figure 30: An (n, k)-stage syndrome circuit with input from the left end [Lin04, pag.150].

4.2 Reed-Solomon codes

Reed-Solomon codes were invented by Irving Reed and Gustave Solomon, who first presented the codes in 1960 [Ree60]. The idea that ultimately led to these codes is the use of non-binary finite field symbols in byte-level operations, based on Galois fields theory [Wic94].

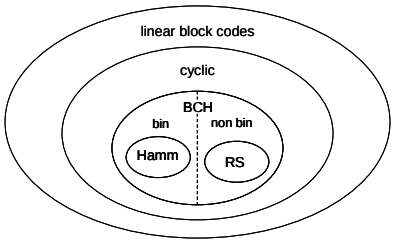

Figure 31: Block codes sets. Linear block codes include cyclic codes, which in turn include BCH codes. BCH codes can be divided into binary and non-binary codes. Example of binary BCH is the Hamming code, examples of non-binary are RS codes.

RS codes are a subset of BCH codes [Lin04]. BCH codes are thus called because of their discoverers Bose, Chaudhuri and Hocquenghem ([Hoc59], [Bos60]). BCH codes are a generalization of Hamming codes and are cyclic

[Lin04]. A set diagram is displayed in Figure 31 showing how RS codes are non-binary BCH codes, which in turn are linear cyclic block codes.

In the following, an introduction on Galois field algebra is given in 4.2.1. RS codes are introduced in 4.2.2 together with the encoding algorithm. The decoding algorithm is presented in 4.2.4. Two examples applications of RS codes are very briefly presented in 4.2.5: deep-space communications and the compact disc system.

4.2.1 Galois fields algebra

A field is a set of elements in which addition, subtraction, multiplication and division can be performed without leaving the set. Addition and multiplication must satisfy the commutative, associative and distributive laws. A zero element is defined as the identity element with respect to addition ($a + 0 = a$). A unit element is defined as the identity element with respect to multiplication ($a \cdot 1 = a$).

A field with a finite number of elements is called a *finite* field, or Galois Field, after the name of their discoverer [Lin04]. The field GF(q) is formed by q elements. For example, the set {0, 1} is the *binary* field GF(2) when considered together with modulo-2 addition and modulo-2 multiplication (truth tables in Table 4).

+	0	1
0	0	1
1	1	0

.	0	1
0	0	0
1	0	1

Table 4: Modulo-2 addition truth table (left) and modulo-2 multiplication truth table (right). The results of the operations are still elements of the field. Addition is equivalent to subtraction.

If *q* is a prime number, for any positive integer *m*, it is possible to extend the prime field GF(q) to a field of q^m elements GF(q^m), called an *extension* field of GF(q). The binary field GF(2) and its extension GF(2^m) are most widely used in digital data transmission and storage systems, as these systems are based on binary codification of data for practical reasons.

When multiplying one element of GF(q) times itself repeatedly, the result is unity for some power as there are only a finite number of elements in the field. In fact, the smallest positive integer n which verifies that $a^n = 1$ is called the *order* of the field element *a* [Lin04]. A non-zero element *a* is said to be *primitive* if the order of *a* is (q – 1), and GF(q) always contains at least

one primitive element. A similar property exists also for the module-q addition, when summing an element repeatedly to itself. The smallest positive integer λ which verifies that summing the number one, λ times, adds to zero is called *characteristic* of the field [Lin04]. In formulas:

$$n = \min_{l}\{a^l = 1\} \quad \text{and} \quad \lambda = \min_{l}\left\{\sum_{i=1}^{l} 1 = 0\right\}$$

For a primitive element α, all the powers of α are the distinct elements in the field: $\{1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$. That is, the elements of the field can be expressed as powers of a primitive element α (*power representation*). This representation is very effective when expressing a multiplication operation, as: $\alpha^x \cdot \alpha^y = \alpha^{x+y}$.

An alternative representation of GF elements is the *polynomial* representation, which is more appropriate for additions. The polynomial representation of the field elements of $GF(q^m)$ in based on a *primitive* polynomial over the field GF(q). Primitive polynomials are a subset of *irreducible* polynomials (i.e. cannot be factorized in lower order polynomials) and are listed in tables (as Table 2.7 in [Lin04]).

An example of the construction of a GF is given next, and different possible representations of the field elements are given in Table 5.

4.2.1.1 Example: GF(2^4)

$GF(2^4)$ is an extension field of the binary field GF(2) and has a total of 16 elements (15 elements different from zero, plus the 0 element).

A primitive polynomial over GF(2) is $p(x) = x^4 + x + 1$. Consequently, if α is a root of p(x), then $p(\alpha) = \alpha^4 + \alpha + 1 = 0$ and thus $\alpha^4 = \alpha + 1$. It follows that:

$$\alpha^5 = \alpha \cdot \alpha^4 = \alpha \cdot (\alpha + 1) = \alpha^2 + \alpha$$

$$\alpha^6 = \alpha^2 \cdot \alpha^4 = \alpha^2 \cdot (\alpha + 1) = \alpha^3 + \alpha^2$$

and so on for all other powers until:

$$\alpha^{15} = \alpha \cdot \alpha^{14} = \alpha \cdot (\alpha^3 + 1) = \alpha^4 + \alpha = \alpha + 1 + \alpha = 1.$$

The polynomial notation is convenient when an addition has to be performed, i.e.:

$$\alpha^{14} + \alpha^{9} = (\alpha^{3} + 1) + (\alpha^{3} + \alpha) = \alpha + 1 = \alpha^{4}$$

On the contrary, the power notation is convenient when a multiplication has to be performed, i.e.:

$$\alpha^{10} \cdot \alpha^{12} = \alpha^{22} = \alpha^{15} \cdot \alpha^{7} = 1 \cdot \alpha^{7} = \alpha^{7}$$

Table 5: List of elements in GF(4), indicated through different representations.

power representation	polynom.representation	4-tuple representation
0	0	0000
1	1	0001
α^1	α	0010
α^2	α^2	0100
α^3	α^3	1000
α^4	$\alpha + 1$	0011
α^5	$\alpha^2 + \alpha$	0110
α^6	$\alpha^3 + \alpha^2$	1100
α^7	$\alpha^3 + \alpha + 1$	1011
α^8	$\alpha^2 + 1$	0101
α^9	$\alpha^3 + \alpha$	1010
α^{10}	$\alpha^2 + \alpha + 1$	0111
α^{11}	$\alpha^3 + \alpha^2 + \alpha$	1110
α^{12}	$\alpha^3 + \alpha^2 + \alpha + 1$	1111
α^{13}	$\alpha^3 + \alpha^2 + 1$	1101
α^{14}	$\alpha^3 + 1$	1001

4.2.2 Code construction and encoding algorithm

If α is a primitive element of GF(q), the generator polynomial $\mathbf{g}(X)$ of a t-error-correcting RS code (RS codes are cyclic codes) with symbols from GF(q) has α, α^2, …, α^{2t} (2t consecutive powers of α) as all its roots [Lin04]:

$$\mathbf{g}(X) = (X - \alpha)(X - \alpha^2)...(X - \alpha^{2t}) = g_0 + g_1 X + g_2 X^2 + ... + g_{2t-1} X^{2t-1} + X^{2t}$$

with $gi \in GF(q)$ for $0 \le i < 2t$

Following the definition of $\mathbf{g}(X)$, the number of parity check symbols is 2t. The number of codeblock symbols is n = q - 1 and thus the available number of message symbols is k = q - 1 - 2t. It can be proven that the minimum distance is $d_{min} = 2t + 1$, and consequently RS codes are MDS codes. It follows also that the code rate is:

$$R_c = \frac{k}{n} = \frac{q - 1 - 2t}{q - 1}$$

In the expression reported above, the dependence of the code rate from both the number of correctable errors *t* and the code block length *q* stresses that the longer the block is, the higher the rate.

As digital transmission is based on bits for practical purposes, it is convenient to define $q = 2^m$ so that the code symbols, which belong to GF(q), are easily converted to a binary notation. The block length can then be expressed as $n = 2^m - 1$. It follows that each of the n transmittable symbols belongs to GF(2^m), and each of the n symbols consists of m bits. It follows that:

$$R_c = \frac{k}{n} = \frac{2^m - 1 - 2t}{2^m - 1}$$

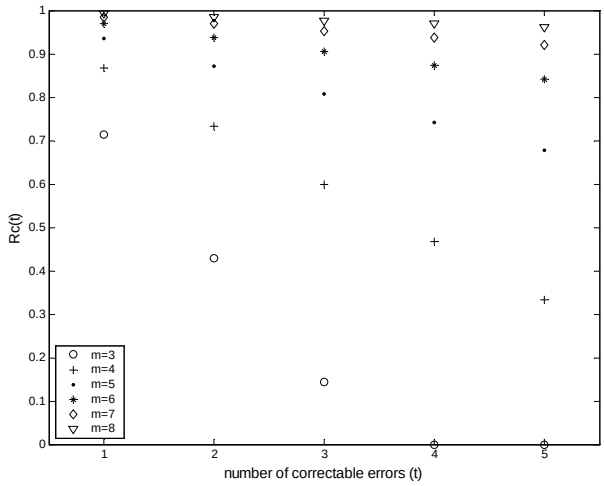

Figure 32: Variation of code rate R_c with respect to number of bits per symbol m and number of correctable errors t.

In this expression for the code rate R_c, the dependence on the number of correctable errors *t* and the symbol bit-width *m* is explicit. In Figure 32 the variation of R_c with *m* and *t* is shown: for each number of correctable errors *t*, the rate is higher for a higher number of bits *m*, that is a longer block *q*.

It has to be pointed out that code designers are not required to use the *natural* size of Reed-Solomon code blocks, block which could be characterized as (q – 1, q – 1 – 2t) in the usual (n, k) notation. A technique known as *shortening* can produce a smaller code of any desired size from a

larger code. In the *shortened* code, l leading information symbols are padded to zero, where $0 \leq l < q - 1 - 2t$, and thus the shortened code is a $(q - 1 - l, q - 1 - 2t - l)$ code. The minimum distance of the code is maintained. Shortening is equivalent to using only a subset of the code, that is keeping l information symbols to 0, and using only the remaining symbols for conveying the message. The rate of a shorted code is thus smaller than that of a natural code:

$$R_c = \frac{k}{n} = \frac{2^m - 1 - 2t - l}{2^m - 1 - l}.$$

4.2.3 RS encoding

Given the generator polynomial, the encoding algorithm and circuitry is straightforward, due to the cyclic structure of the code, as already explained in section 4.1.3. In Figure 33 a linear feedback shift register implementation of the RS encoder is shown. It should be noted that g_i are elements of $GF(2^m)$ and that b_i are m-parallel memory elements.

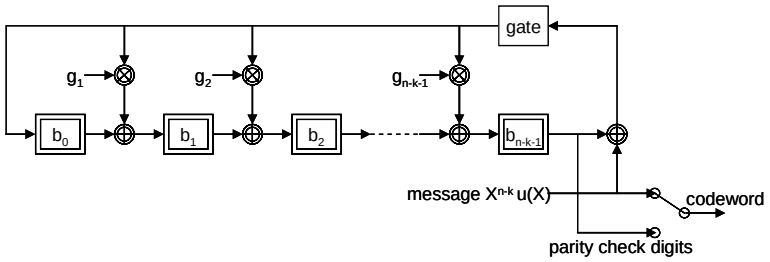

Figure 33: Encoding circuit for an (n, k) RS code [Lin04, page 147]. The generator polynomial is in the form: $g(X) = 1 + g_1X^2 + ... + g_{n-k-1}X^{n-k-1} + X^{n-k}$.

4.2.4 RS decoding

The decoding process consists first the evaluation of the presence of errors in the received block, and secondly the correction of the errors in case they are detected. Error detection is carried out through the syndrome evaluation. If errors are detected, the error pattern has to be derived so that the received codeword can be corrected.

In case of a Hamming error correction code, for example, finding the error pattern is equivalent to finding the position of the corrupted bit. Once the position is known, the error can be corrected by simply flipping that bit:

as only two values are possible for each bit, if a logic one is wrong, then a logic zero is the correct value and vice versa.

On the contrary, in the case of RS codes, evaluating only the error position is not sufficient to complete the error correction process. As the symbols belong to GF(q), with *q* in general bigger than two, the *error amplitude* has to be derived as well. As $q = 2^m$, each symbol is transmitted as a sequence of *m* consecutive bits: finding the error amplitude corresponds to correcting up to all the *m* bits in the symbol. The correction of one symbol corresponds to the correction of a sequence of *m* bits, for example the capability of correcting two symbol errors per codeword guarantees tolerance to any (m + 1)-bit burst. This makes RS codes very suitable for burst error correction (see Figure 34).

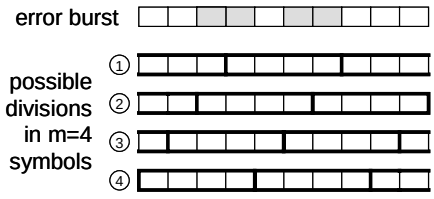

Figure 34: Example or (m+1)-bit burst correctable with an m-bit RS symbol width. Regardless of the symbol positions, a 5-bit burst can be corrected by correcting two 4-bit symbols.

Many different algorithms were invented for decoding RS codes. The first paper [Ree60] introduced the code structure and a simple decoding algorithm which could be useful only for the smallest RS codes. More practical algorithms were invented later, due to various other researchers like Berlekamp [Ber68], Forney [For65], Chien [Chi64], etc.

Out of all different algorithms, one common process flow can be extracted: syndromes calculation is followed by derivation of error locations and of error amplitudes (see Figure 35).

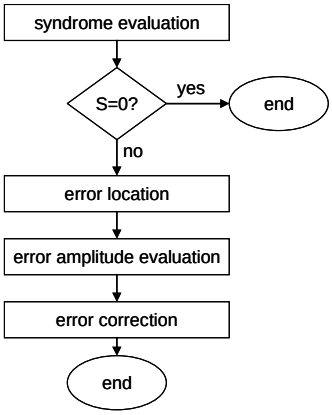

Figure 35: Error correction decoding algorithm. Syndromes are evaluated first. If they are zero, the received block is correct and no error correction is to be performed. If the syndromes are not equal to zero, the error has to be located, the amplitude evaluated and finally error correction performed by module-2 summing the error amplitude to the symbol received in the calculated error location.

Being RS codes cyclic, any transmitted codeword is a multiple of the generator polynomial, which has 2t consecutive powers of a primitive element α as roots. This means that for a transmitted codeword v(x), the 2t consecutive powers of α are also roots of v(x): $v(\alpha^i) = 0$ for $1 \leq i \leq 2t$. Explicitly that is:

$$(v_0, v_1, \ldots, v_{n-1}) \cdot \begin{bmatrix} 1 \\ \alpha^i \\ \alpha^{2i} \\ \vdots \\ \alpha^{(n-1)i} \end{bmatrix} = 0 \quad \text{for} \quad 1 \leq i \leq 2t .$$

The code parity-check matrix H has the property that $(v\, H^T) = 0$, and:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & (\alpha^2)^3 & \ldots & (\alpha^2)^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & (\alpha^3)^3 & \ldots & (\alpha^3)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & (\alpha^{2t})^3 & \ldots & (\alpha^{2t})^{n-1} \end{bmatrix} = 0$$

As the received block $\boldsymbol{r}$ is equal to the transmitted block $\boldsymbol{v}$ summed to an error pattern $\boldsymbol{e}$ (that is: $\boldsymbol{r} = \boldsymbol{v} + \boldsymbol{e}$), and since the syndromes are defined as: $\boldsymbol{s} = \boldsymbol{r}\,\mathrm{H}^{\mathrm{T}} = (s_0, s_1, \ldots, s_{n-k-1})$, then the syndromes depend only on the error pattern: $\boldsymbol{s} = \boldsymbol{e}\,\mathrm{H}^{\mathrm{T}}$. The expression $\boldsymbol{s} = \boldsymbol{r}\,\mathrm{H}^{\mathrm{T}}$ gives a means to evaluate the syndromes, while the expression $\boldsymbol{s} = \boldsymbol{e}\,\mathrm{H}^{\mathrm{T}}$ is why the calculation of the syndromes is useful, and why they depend only on the error pattern and not on the transmitted vector.

The syndromes can be calculated from the received word either with a division circuit (as anticipated in section 4.1.3), or directly from the polynomial expression (more on this in section 6.4.3.2.1):

$$S_i = r(\alpha^i) = r_0 + r_1\alpha^i + r_2\alpha^{2i} + \cdots + r_{n-1}\alpha^{(n-1)i} \text{ for } 1 \le i \le 2t$$

As the syndromes depend only on the error pattern, if ν errors have occurred, then the i-th syndrome takes the following form (for $1 \le i \le 2t$):

$$S_i = r(\alpha^i) = v(\alpha^i) + e(\alpha^i) = e(\alpha^i) = e_{j_1}\alpha^{i \cdot j_i} + e_{j_2}\alpha^{i \cdot j_2} + \cdots + e_{j_\upsilon}\alpha^{i \cdot j_\upsilon}$$

where j_k ($1 \le k \le \nu$) are the error positions, or positions of the corrupted symbols, and e_{jk} are the error amplitudes. The syndromes S_i can be calculated from the received word, and from them j_k and e_{jk} are derived in order to perform error correction. Different algorithms correspond to different ways of deriving error locations and amplitudes from the syndromes. Three are the most commonly used algorithms: Berlekamp's algorithm [Ber82], the Euclidean algorithm [Sug75] and frequency domain decoding [Bla79]. Only Berlekamp's algorithm will be presented here.

An *error locator* polynomial $\sigma(x)$ is constructed in such a way that its roots are the reciprocals of the error positions. Thus, finding the roots of the polynomial becomes equivalent to finding the error locations.

$$\sigma(x) = \left(1 - \alpha^{j_1}x\right)\left(1 - \alpha^{j_2}x\right)\cdots\left(1 - \alpha^{j_\upsilon}x\right) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_\upsilon x^\upsilon$$ The coefficients σ_i ($1 \le i \le n$) are to be computed from the manipulation of the syndromes S_i. Berlekamp's iterative procedure can be used to construct these coefficients and obtain:

$$\sigma^{(2t)}(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_{2t-1}x^{2t-1} + \sigma_{2t}x^{2t}$$

Description of the procedure details can be found in [Lin04].

By using 2t syndromes as parameters for the construction of a polynomial, though, only a polynomial of degree up to 2t can be derived. The constructed polynomial then coincides with the true error-location polynomial only if they both have the same degree or equivalently, if the number ν of errors in the error pattern does not exceed the error-correction capability t of the code: $\sigma(x) = \sigma^{(2t)}(x)$.

Once the polynomial coefficients are calculated, the roots of the polynomial have to be found. This step is often performed through Chien's search [Chi64]: the elements α^i of GF(2^m) are substituted in σ(x) cyclically, and if $\sigma(\alpha^i) = 0$, then α^i is a root of σ(x). It follows that the error-location number is given by: $\alpha^{-i} = \alpha^{q-1-i}$, with $q = 2^m$. The systematic search of the roots of the polynomial is computationally feasible (in $q = 2^m$ steps) due to the fact that only a finite number of candidates have to be tried for finding the roots.

Once the error locations are identified, the last remaining step is the evaluation of the error amplitudes. These are calculated through the following expression, which was derived by Forney [For65]:

$$e_{j_k} = \frac{-Z_0(\alpha^{-j_k})}{\sigma'(\alpha^{-j_k})}$$

where $Z_0(x) = \sigma(x)S(x)$ is the error-amplitude evaluator polynomial, and σ'(x) is the derivative of σ(x):

$$\sigma'(x) = \frac{d}{dx}\prod_{i=1}^{\nu}(1 - \alpha^{j_k} x).$$

4.2.5 Applications of Reed-Solomon codes

RS codes are used in many fields, of which two examples are deep-space communications and compact discs.

RS codes for deep-space communications are used in concatenated error-correction systems, where RS codes are concatenated with convolutional codes due to the fact that the channel is a power limited, wideband and additive Gaussian channel [Wic94]. The Consultative Committee for Space Data Systems in 1984 recommended a standard that uses a (255, 223) RS code over GF(2^8).

The compact disc system was standardized in 1980 by Sony and Philips. Concerning the RS codes used in the standard, two codes with elements from $GF(2^8)$ are used: the first is characterized by $n_1 = 32$, $k_1 = 28$ while the second is characterized by $n_2 = 28$, $k_2 = 24$, for a total code rate of 3/4. In the encoding process, RS coding is followed by the eight-to-fourteen modulation introduced in section 2.1.3 which, due to the use of three merging bits, has a rate of 8:17, or 47%. The error correction system is such that bursts of up to 4000 data bits can be corrected.

4.3 Interleaving

Given an (n, k) block code C, it is possible to construct a (λn, λk) block code by a technique called *interleaving* [Lin04] In the simplest form of interleaving, the interleaved (λn, λk) code C^{λ} is λ times as long with λ times as many information symbols, and is constructed by rearranging λ codewords of C into λ rows in a rectangular array and then transmitting the array column by column (see Figure 36). The parameter λ is called interleaving *depth*.

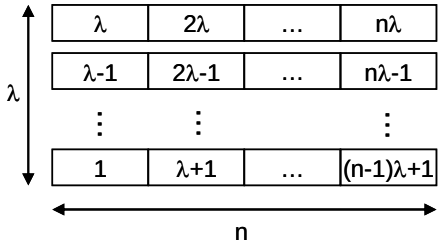

Figure 36: Construciton of a (λn, λk) interleaved code: the data are written row-wise and read column-wise. The rows belong to the (n, k) code, and the number inside each piece of information indicates the order of transmission (1, …, λn).

If the minimum distance of the code C is d_{min}, the minimum distance of the interleaved code C^{λ} is also d_{min}. The interleaving technique though is very effective for deriving long codes for correcting errors that cluster to form bursts from short codes.

In fact, a pattern of errors can be corrected for the interleaved code C^{λ} if and only if the pattern of errors in each single row is a correctable pattern for C. For example, a burst of length λ affects no more than one digit in each row, regardless of where it starts. Thus, if C corrects single errors, the interleaved code C^{λ} corrects single bursts of length λ. If C corrects any

single burst of length l or less, the interleaved code corrects any single burst of length λl or less.

To be noted that the required memory capacity of the interleaver is λn symbols instead of n symbols for the simple code C, and that interleaving increases the latency of the system, as λn symbols are to be collected, instead of n, before the decoding process of one word belonging to C^{λ} can be performed.

4.4 Summary

This chapter introduces the basics of error correction theory which are needed for the line code proposed in this thesis. Apart from general considerations on error control coding, Reed-Solomon codes are introduced along with encoding and decoding techniques. Finally, the technique of interleaving is also presented.

Chapter 5

GBT line code

This chapter introduces the line code proposed for the VBD link and the GBT ASIC. An overview of the line code desired functionality and building blocks is given in section 5.1. The line code main blocks are described in the following sections: the scrambler (section 5.2), the RS code (section 5.3) and the header addition (section 5.4). The line code is proposed in two different interleaving options which are introduced in section 5.5.

5.1 Line code overview

Line coding is a coding technique used to match the data stream to the characteristics of the channel. The line code proposed for the VBD link is designed to provide the encoded data, which constitutes the signal on the fibre, with the following properties:

1) A high number of transitions for clock recovery;
2) DC-balance for allowing AC-coupling in the receiver;
3) Non-periodicity of the serial data in order to have a smaller amount of pattern dependent jitter at the receiver;
4) Frame synchronization capability in order to reconstruct the parallel form of data with the exact phase;
5) High efficiency in order to keep link bandwidth and rate of terminal circuitry as low as possible;
6) A FEC scheme mainly in order to address the issue of SEUs on the photodiode;

7) Low latency of the coding and decoding algorithms for rapid delivery of L1 trigger information.

While the last two points of the list are a concern especially for the VBD link, the previous ones are common to many commercial links. For this reason, commercially adopted schemes were evaluated for our application, even though none of them was found fully compliant with our needs (refer to section 3.4). In particular it can be pointed out that conventional line codes have the drawback of error multiplication, which requires FEC codes characterized by higher error correction capability, which in general requires a higher amount of redundancy and more complicated decoding algorithms (resulting in increased power consumption of decoding time).

In order to avoid error multiplication caused by conventional line codes, the proposed scheme performs error correction before line decoding. In this way, errors produced on the line or on the photodiode can be corrected before being line-decoded, where they would be multiplied (see Figure 37).

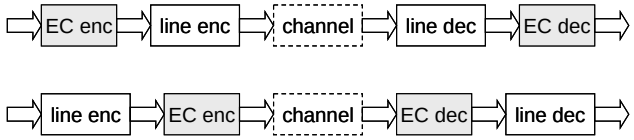

Figure 37: The two possible schemes based on the concatenation of EC block and line code block: line code placed internally to EC causes error multiplication (top), so a scheme which inverts them is used in the proposed scheme, placing line coding externally to EC (bottom).

In the proposed line code, the first three points of the list above are addressed through the use of a scrambler which can be simply implemented via a linear feedback shift register and requires no bandwidth increase (details in section 5.2). FEC is performed by a RS interleaved code (see section 5.3). A header is used for acquiring frame lock (section 5.4).

The concatenation of functional blocks that perform line encoding for the data to be transmitted is shown in Figure 38. Scrambling is followed by RS encoding and header addition before the word is serialized out to the channel. On the receiver side of the link, the inverse operation is performed in order to reconstruct the original data, as shown in Figure 39.

The remainder of this chapter discusses in detail the line code proposed to attain error-free transmission in a radiation environment. Two different implementation options are devised for the same scheme reported above.

From the choice of two different RS symbol widths and number of interleaved blocks, two different error correction capabilities are obtained. The options are presented in section 5.5 of this chapter.

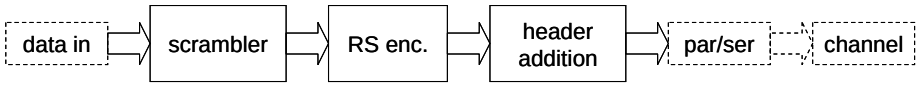

Figure 38: Block scheme of the proposed line encoding scheme: scrambling is performed first, followed by RS encoding and header addition.

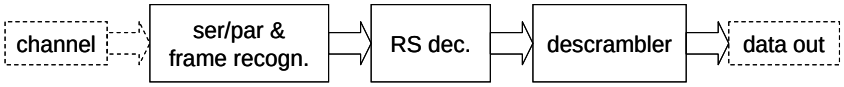

Figure 39: Block scheme of the proposed line code, decoder. The serial-to-parallel conversion is done through frame recognition. RS decoding and descrambling follow.

5.2 DC balance and abundance of transitions

5.2.1 Introduction on scrambling techniques

A scrambler is a device that manipulates a data stream before transmission for achieving DC-balance and reducing the occurrence of long sequences consisting of "0" or "1" only.

When data are scrambled, their statistics are "whitened" so that the scrambler output looks "random" [Lee94]. In case of random data, DC-balance is intrinsically guaranteed: the probability of occurrence of a 0 is the same as that for a 1 ($P_0 = P_1 = 1/2$). This means that on average, in a long stream, the same number of 0s and 1s are present. A high number of transitions is also statistically guaranteed as long sequences of identical data are unlikely to happen: for a n-bit sequence of all 0s or all 1s, the probability of occurrence is $(1/2)^n$, which exponentially decreases with n. This makes the probability of occurrence of long sequences negligible.

A scrambler requires no redundancy. This means that the scrambler input and output data space are the same. In other words, all sequences are still possible and the key point is to map "problematic" sequences (non-DC-balanced or low in number of transitions) which are fairly likely to occur into sequences which are less problematic as they look more random. Problematic scrambler outputs are still possible, but they occur more rarely, guaranteeing link operation de facto. A scrambler, then, provides only statistical guarantees, while requiring no bandwidth increase. On the contrary, the advantage of 8b/10b over other techniques is based on the fact

that DC-balance and short maximum run-length are strictly guaranteed, but this is paid with a 25% bandwidth increase.

Scramblers are based on the use of maximal-length shift-register sequences, which are periodic bit sequences with properties that make them appear to be random. They are also called “pseudorandom” bit sequences.

A second advantage of scrambling over other methods, along with the absence of bandwidth increase, is its straightforward implementation through linear feedback shift-registers, as pseudorandom sequences can be generated by n-bit feedback shift-registers. The feedback connections consist or exclusive-or operations as depicted in Figure 40. These shift-registers are called “maximal-length” shift-registers as the output binary sequence has period $r = 2^n - 1$, which is the maximum output period achievable by an n-bit shift register given that the feedback connections are carefully chosen.

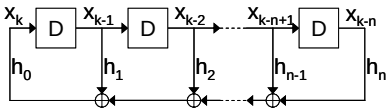

Figure 40: Linear feedback shift-register. The coefficients h_i are binary, and the sum is modulo-two (ex-or).

An n-bit shift-register can in general assume at most 2^n distinct values. In the case of feedback register as the one depicted in Figure 40, the all-zero state cannot be visited, if the aim is a long period, as the register would remain in the all-zero state due to the absence of an input. Moreover, two successive states must be different as, if the state is the same from one time increment to the next one, then it is forever the same. For an n-bit shift-register then the longest output period possible is $2^n - 1$, where all states are visited once apart from the all-zero state. Clearly, a higher value for n guarantees a longer period for the scrambler state sequence. The register has to be initiated with a non-all-zero state in order to produce the periodic sequence.

The mathematical theory of primitive polynomials [Lee94] lists which of the taps h_i in Figure 40 have to be connected in order to get a maximal-length register. The connections h_i which lead to maximal-length registers are tabulated as function of the polynomial order n [Lee94]. Often many primitive polynomials exist for the same order n, but usually only the “minimal weight” ones are reported, that is the ones which employ a

minimum number of taps, which minimizes the number of exclusive-or operations to be performed. In Table 6 the non-zero coefficients of the minimal weight polynomial for some orders n are reported. It can be seen that all maximal-length registers have at least three non-zero taps, out of which two are always the n^{th} tap, and the 0^{th} tap (if these two are zero, then it would be a lower order scrambler), which correspond to h_0 and h_n in Figure 40.

Table 6: Positions i of non-zero coefficients (h_i) of minimal weight primitive polynomials for orders n = 2 through n = 21 and n = 55 through n = 64. The most significant bit is $h_n = 1$, and the least significant bit is $h_0 = 1$. Taken from table 12-2 in [Lee94].

n	coefficients	n	coefficients	n	coefficients
2	2,1,0	12	12,7,4,3,0	55	55,24,0
3	3,1,0	13	13,4,3,1,0	56	56,22,21,1,0
4	4,1,0	14	14,12,11,1,0	57	57,7,0
5	5,2,0	15	15,1,0	58	58,19,0
6	6,1,0	16	16,5,3,2,0	59	59,22,21,1,0
7	7,1,0	17	17,3,0	60	60,1,0
8	8,6,5,1,0	18	18,7,0	61	61,16,15,1,0
9	9,4,0	19	19,6,5,1,0	62	62,57,56,1,0
10	10,3,0	20	20,3,0	63	63,1,0
11	11,2,0	21	21,2,0	64	64,4,3,1,0

The descrambler has the same structure as the scrambler but the data flow is reversed. In order to reconstruct the original bit-stream the shift registers in the scrambler and descrambler have to be synchronized. Two are the most used types of scrambling, which differ mainly on the synchronization technique [Lee95]: Frame-Synchronized Scrambling (FSS, or "additive scrambling") and Self-Synchronized Scrambling (SSS, or "multiplicative" scrambling).

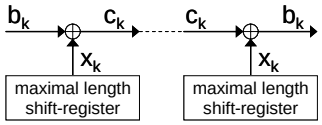

Figure 41: Frame-synchronized scrambler.

FSS transforms the input data stream by modulo-2 summing it to a pseudorandom sequence (Figure 41). In a FSS a synch command is required for synchronizing the scrambler and descrambler shift-register states, and this command has to be periodically sent. This type of scrambling is convenient only if framed signals are large.

In SSS the states are automatically synchronized without any additional synchronization command. While FSS uses “autonomous” shift register generators, SSS relies on “excited” shift register generators, where the excitement comes from an external input (see Figure 42). The descrambler automatically acquires synchronization when the number of received data reaches the number of memory elements in the “excited” shift register generator. However, if a bit error occurs on the scrambled signal, it stays in the descrambler shift register for a while, causing multiple bit errors in the descrambled signal: each error is multiplied to additional ones for each non-zero tap in the shift register. SSS is used for example 10G Ethernet [Iee00].

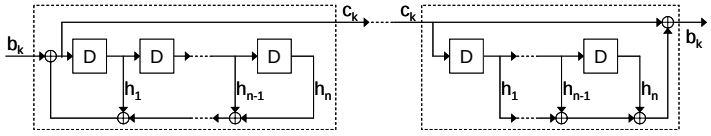

Figure 42: Self-synchronized scrambler.

From Figure 41 and Figure 42, it can be seen that in the FSS case, the input data stream is summed to the scrambler output to obtain the scrambled sequence, while in the SSS case the data input stream is summed directly into the scrambler states.

The periodicity of the scrambler output sequence depends on both the number of scrambler states and the input sequence [Lee94]. In case of a FSS, the output period is the least common multiple between the period of the input and the period of the pseudorandom generator. In case the input data have period equal or multiple to that of the generator, then the output can have a much shorter period that the generator. A general policy to reduce the occurrence of cases with short periodicity is to choose the generator period long and n a prime number.

In the case of SSS, if the input is periodic the output period might have two different periods depending on the generator initial state. The worst case is for one particular state (which has probability 2^{-n} of occurring, [Lee94]) for which the output sequence period is the same as the input period, in which case it might be excessively small. For the remaining states (with probability $1 – 2^{-n}$) the output has period equal to the least common multiple of the input and generator periods. Choosing a high value for n makes the probability of occurrence of the first and problematic case negligible.

For both FSS and SSS the most common case is the output sequence period being the least common multiple between the input period and the generator period. Choosing a high value for n translates to a high value for $2^n - 1$, i.e. to long output periods. Even in case the input has constant value (period equal to one), the output period is $2^n - 1$.

FSS and SSS have thus similar periodicity issues, which are eased by the choice of a large n for the register bit length. The advantage of choosing SSS over FSS lies in the fact that no synchronization command or code is needed, the disadvantage being error multiplication. FSS on the other hand does not multiply errors as the input data has no effect on the generator register states, but requires a reset procedure.

5.2.2 Scrambler for the GBT ASIC

The GBT ASIC is to be used in a broadcast link topology, and an implementation as SSS is chosen for this reason. In this way, in case one single receiver in the broadcast network looses lock, the same receiver can restart the locking procedure without having to wait for a particular command from the master transmitter, minimizing the time elapsed between loss of lock and lock acquisition. An example of implementation of a SSS scrambler and descrambler is shown in Figure 42. The new scrambled output c_k is evaluated by combining the present data b_k and previous scrambled outputs. Descrambling consists of recovering the original data by summing the present scrambled value and a combination of the previously received ones.

The scrambler internal state needs to be reset at the beginning of the transmission for the scrambled sequence to be predictable for testability. Concerning the descrambler, the register values are instead filled up by the incoming data, so that a reset is not needed. This is the reason why the implementation is self-synchronizing: the descrambler output is valid as soon as all the memory elements that make up the descrambler internal stata contain valid data (i.e. after n valid bits are received). This is also the origin of error multiplication: if an erroneous bit is received, this affects multiple output bits, until the corrupted one falls off the n-bit register, after n clock periods. The number of multiplied errors is equal to the number of non-zero taps h_i.

When resetting the scrambler internal state at the beginning of the transmission, the generator register can be filled in with any pattern but the

all-zero word due to the output periodicity issues introduced before. In the GBT implementation this was taken care of by starting the transmission with a sequence of idle patterns, patterns that are also used for keeping the transmission active in case of absence of user data. The idle pattern then is chosen to be different from the all-zero pattern; additionally it is chosen to be aperiodic in order to maximize the output sequence period.

The GBT self-synchronizing scrambler and descrambler use a parallel equivalent implementation of the serial scrambler and descrambler in order to match the ASIC N-parallel input and to diminish latency and minimize the operation frequency: a serial implementation requires N clock cycles to transform N data inputs in N scrambled data outputs, while the parallel implementation requires only one clock period, at the expense of increased logic and silicon area.

The order n of the scrambler is to be chosen as big as possible for periodicity reasons and, at the same time, lower than the user data parallel word width: in case of error multiplication, the carried-on errors are brought only one word forward and not more. At the same time, the memory register within the scrambler is dimensioned to have the width of a user data parallel word. The choice of the polynomial has also some impact on the amount of exclusive-or operations to be implemented, so that it is convenient to choose a polynomial that minimizes them. More details on the scrambler and descrambler are given in chapter 6 when dealing with implementation details.

5.3 Error correcting scheme

RS codes are introduced in section 4.2 of this thesis. They are chosen for the error correction scheme in the GBT line code as they are very efficient, they can correct multiple errors and are well suited for correction of burst errors, and because the code construction is very flexible.

In order to minimize link latency, it was chosen to keep the complexity of the code to the minimum by choosing to correct only one error per RS block ($t = 1$). In this way, the procedure of calculating the error locator polynomial, finding its roots, calculating error amplitudes (as presented in the third chapter) can be substituted by a faster procedure which is based on simple inversions and which requires fewer operations and thus clock cycles.

In case t = 1 correctable errors per block are chosen, two redundancy symbols (that is 2m redundancy bits) are appended to each data block. For a given symbol width m the total maximum number of codeword bits $N_{RS,max}$ is $[(2^m - 1)\ m]$ and the maximum number of message bits $K_{RS,max}$ is $[(2^m - 1 - 2t)\ m]$. The number of bits $N_{RS,max}$ and $K_{RS,max}$ for m between 3 and 6 are given in Table 7.

Table 7: Maximum number $N_{RS,max}$ of data bits per block and of information bits per block $K_{RS,max}$, given a symbol width m. The number of redundancy bits needed per each block is 2tm = 2m.

m	$N_{RS,max}$	$K_{RS,max}$	2tm (t = 1)
3	21	15	6
4	60	52	8
5	155	145	10
6	378	366	12

In order for the error correction algorithm to be able to address also SEU error events spreading over two contiguous bits, which possibly belong to different symbols, and in general burst errors of maximum length (m + 1), L Reed-Solomon blocks are interleaved (with L > 1). In this way, if an SEU error or a burst error corrupt more than one bit and cross symbol boundaries, the bits can still be corrected due to the fact that contiguous symbols belong to different RS blocks.

Interleaving improves the system error correction capability at the price of increased redundancy. Given that L blocks are interleaved, the number of redundancy bits is (2m L) for the overall encoding scheme. Once L is fixed, then, it is preferable to keep the symbol width m small in order to reduce code redundancy and improve its efficiency.

The utilization of L blocks implies an increase of the implementation complexity. Either L structures are to be used in parallel, or a pipelined structure has to be put in place. In order not to penalize the link latency, it is chosen to implement parallel interleaving. This has the additional advantage that the registers can operate at a frequency that is L times smaller. For these reasons, L decoding blocks are implemented and used in parallel.

5.3.1 Syndromes calculation

The calculation of the syndromes is introduced in section 4.2.3. In the case of correction of one symbol error per block, it gives two distinct syndromes S_1 and S_2:

$$\begin{cases} S_1 = r(\alpha) = r_0 + r_1\alpha + r_2\alpha^2 + ... + r_{n-1}\alpha^{(n-1)} \\ S_2 = r(\alpha^2) = r_0 + r_1\alpha^2 + r_2\alpha^4 + ... + r_{n-1}\alpha^{2(n-1)} \end{cases}$$

In the set of equations above, r is the received vector and α is a primitive element of the GF on which the code is defined.

The next part of the decoding algorithm assumes that a maximum of one symbol error occurred on the RS block (reason why the last equality in the following set holds). If v is the transmitted vector, e is the error vector, the error position is j and the error amplitude e_j, then:

$$\begin{cases} S_1 = r(\alpha) = v(\alpha) + e(\alpha) = e(\alpha) = e_j\alpha^j \\ S_2 = r(\alpha^2) = v(\alpha^2) + e(\alpha^2) = e(\alpha^2) = e_j\alpha^{2j} \end{cases}$$

Following the previous set of equations, the construction of the error locator polynomial is not required, but the error location and amplitude can be derived by direct inversion of the last terms of the equations:

$$\alpha^j = \frac{S_2}{S_1} \quad \text{and} \quad e_j = \frac{S_1}{\alpha^j} = \frac{S_1^2}{S_2} \ .$$

These solutions can thus be readily evaluated through few steps instead of the general and longer error locator polynomial algorithm.

5.3.2 Proof that RS encoding maintains the pseudorandom characteristics

The chosen RS coding scheme can be performed after scrambling in the encoding process as it maintains the pseudorandom characteristics that data gained with scrambling.

In fact, the chosen RS code is systematic, which means that the redundancy bits are appended to the input block without actually modifying the RS coder data input. This implies that the resulting RS coded word maintains the pseudorandom properties in the message bits.

Additionally, the pseudorandom characteristics of the data bits are also acquired by the redundancy bits. In fact, the encoding algorithm consists of a sequence of additions and multiplications on GF(2^m) as can be seen by the shift register implementation depicted in Figure 33. The bit-by-bit module-2 sums are equally likely on GF(2^m) given that the two addends have equally

likely statistics. This is due to the closeness of the field and i.e. proven by Table 18, for m = 3, and Table 20, for m = 4, in Appendix. The module-2 products are equally likely on GF(2^m) given that the two factors have equally likely statistics, and given that one of them is known to be different from zero. This is also due to the closeness of the field and i.e. proven by Table 19, for m = 3, and Table 21, for m = 4, in Appendix. In the encoding algorithm multiplications, this is guaranteed by the fact that one factor is always g_i, which is different from zero.

As the inputs to the encoding algorithm are symbols from GF(2^m) picked with equally likely probabilities due to the randomization property of the scrambler, the results of the algorithm, namely the RS check symbols, acquire the same pseudorandom properties.

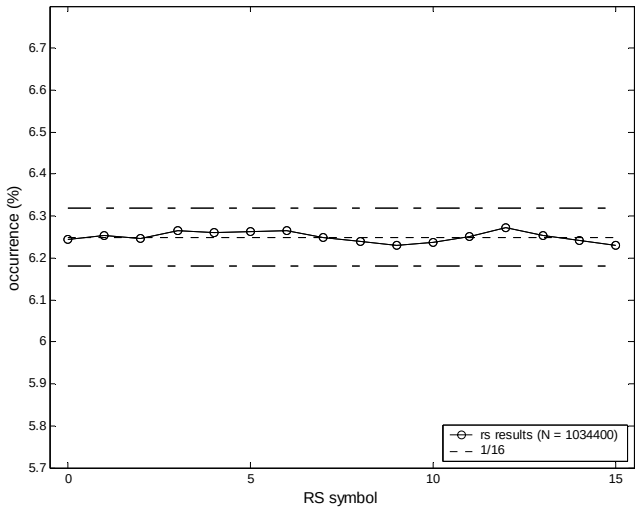

Figure 43: Probability of occurrence of RS check symbols for the (10, 8) code used in the first option. The symbols belonging to GF(2^4) are equiprobable: P_i=1/16 for $0 \leq i \leq 15$.

The argument is additionally proven by the following simulation results. The check symbols calculated for one million vectors are histogrammed, and the percentage of occurrence of each one of them is plotted in Figure 43 and Figure 44 for the GF(2^4) and GF(2^3) respectively, along with the expected average value (1/16 for the first case, 1/8 for the second one). The deviation from the average is small compared with the 1/sqrt(N) expected sigma.

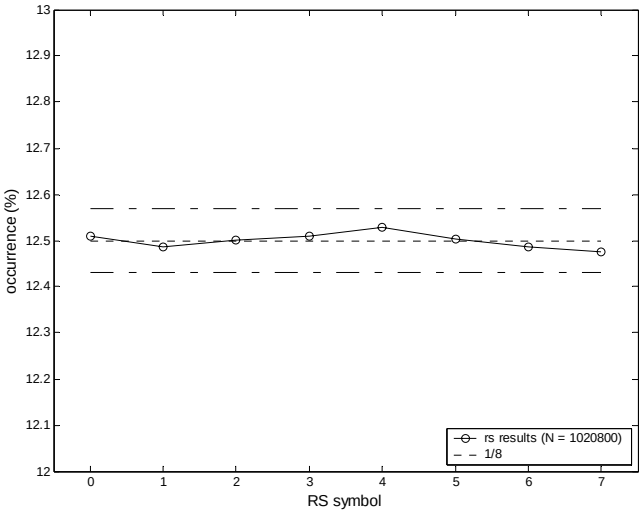

Figure 44: Probability of occurrence of RS check symbols for the (7, 5) code used in the second option. The symbols belonging to GF(2^3) are equiprobable: P_i=1/8 for $0 \leq i \leq 7$.

5.4 Frame alignment issue: error tolerant header

Deserialization is crucial in the receiver operation. Once the receiver CDR has reconstructed the transmitter bit phase and frequency, and thus bit lock is acquired, frame locking is necessary before RS error correction and descrambling can take place. Frame locking operates on frame by frame information and does not need to "understand" any of the frame content.

In the following, in section 5.4.1 the mechanism is introduced, while in sections 5.4.2 and 0 the locking and unlocking mechanisms are detailed. The header pattern choices are explained in section 5.4.4.

5.4.1 Frame locking and unlocking mechanisms introduction

A non–scrambled and non RS-coded header marks the beginning of a new data frame and is used for recognition of the framed data structure in the serial stream. Due to the presence of the scrambler it is very unlikely for a header pattern to occur exactly every N_{tot} bits in the RS coded field, as to simulate a framed structure. Consequently, repeated recognition of a valid header in a fixed position in the frame allows for frame-locking, and

repeated non-valid header recognition causes loss of frame lock. This general principle of frame locking is sketched in Figure 45.

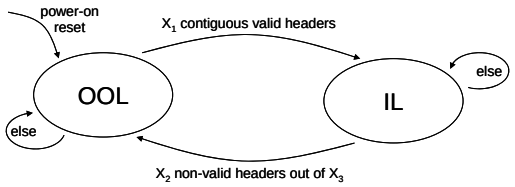

Figure 45: Basic frame synchronization algorithm: Out-Of-Lock and In-Lock states.

The Out-Of-Lock (OOL) state is the starting state after decoder reset. The In-Lock (IL) state is reached after recognition of a number X_1 of contiguous valid header patterns (i.e. headers belonging to contiguous frames). Frame-locking is considered lost if X_2 invalid headers are found within X_3 observed ones.

The header provides the additional functionality of allowing to distinguish different packet types through the use of different header patterns. In the line code designed for the GBT ASIC three header patterns are required to distinguish among three data types: user data, trigger data or idle pattern (see section 3.2.3). User data is used for the point-to-point link functionality, while trigger data is used in case of TTC-like functionality. The idle pattern is used in case of continuous transmission mode in case of absence of user data, so that the receiver can remain locked to the transmitter.

5.4.2 Locking mechanism

The locking mechanism for reaching the IL state from the OOL state is based on recognizing X_1 contiguous correct headers, where "correct" stands for one of the possible chosen header patterns. Depending on the header length H (in bits) and on the total frame length N_{tot} (in bits), the number X_1 is chosen so that it is extremely unlikely to lock to an incorrect position, i.e. incorrect frame phase. This dependence is studied next with the aid of error probability evaluations.

The probability of observing a valid header pattern in the RS coded packet depends on the header length H and on the number A of header patterns that are accepted according to:

$$P(H \ \ from\ scrambler) = P_H = A\left(\frac{1}{2}\right)^H ,$$

as it is proven in section 5.3.2 that the RS coded packet has pseudorandom statistics, so that binary values are equally probable ($P_0 = P_1 = 1/2$). Thus, a particular header pattern has probability $(1/2)^H$ of being observed in a certain position in the RS coded field, and A different patterns are accepted.

In Figure 46 the probability of observing a sequence of correct header patterns in the RS coded field is plotted as a function of the number of headers X_1 required for locking, and for different header lengths H. In the case of H = 2 and 4, two different valid headers are accepted (A = 2), while in case H = 6 and 8 three headers are accepted (A = 3).

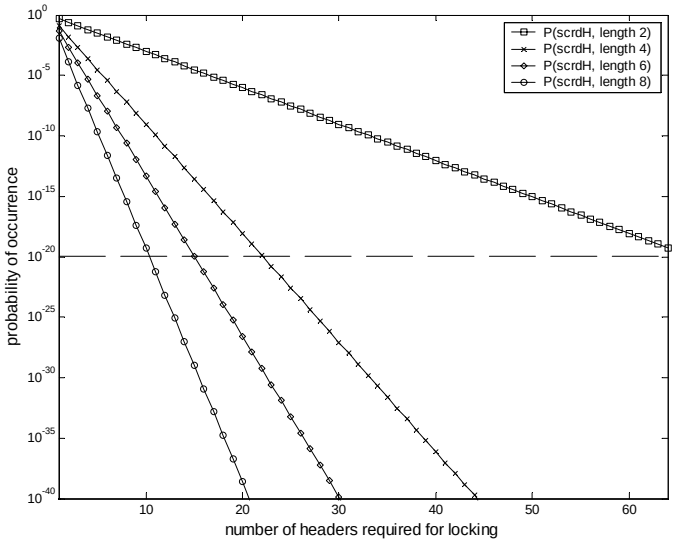

Figure 46: Probability of occurrence of a sequence of valid header patterns in the RS coded field, as a function of the number of headers required for locking X_1.

Clearly, the shorter the number of headers X_1 required for locking is, the more likely it is to lock to an incorrect frame phase. On the other hand, the longer the number of headers X_1 is, the longer the locking process becomes on average. For a fixed number of headers X_1, the longer the header length H, the less likely it is to lock incorrectly. For example, in order to have a

probability of locking incorrectly lower than 10^{-20}, more than 64 contiguous H = 2 headers are required, while 23 are required for H = 4, 16 for H = 6 and 11 for H = 8.

Once the header length H is fixed, also the number A of header patterns accepted for locking has an impact on the probability of locking to an incorrect frame phase. Clearly, the more different header patterns are accepted (i.e. the higher A), the more likely it is to lock to a position in the RS coded field instead of to the correct frame header. Additionally, a higher value for A implies that more comparisons have to be performed on the header bits, which complicates the logic circuitry responsible for locking.

The likelihood of locking to a sequence of random data is shown in Figure 47 for the cases in which H = 6 or 8, and A = 3, 8 or 12. In the case H = 8, a probability below 10^{-20} can be achieved with 11, 14 or 16 for A = 3, 8, 12, and 16, 23 or 28 headers in the case H = 6. Obviously the expressions for (H,A) = (8,12) and (6,3) are equivalent, showing that an increment in header length of two bits allows for four times more accepted header patterns to reach the same probability of obtaining such a sequence in the RS coded field.

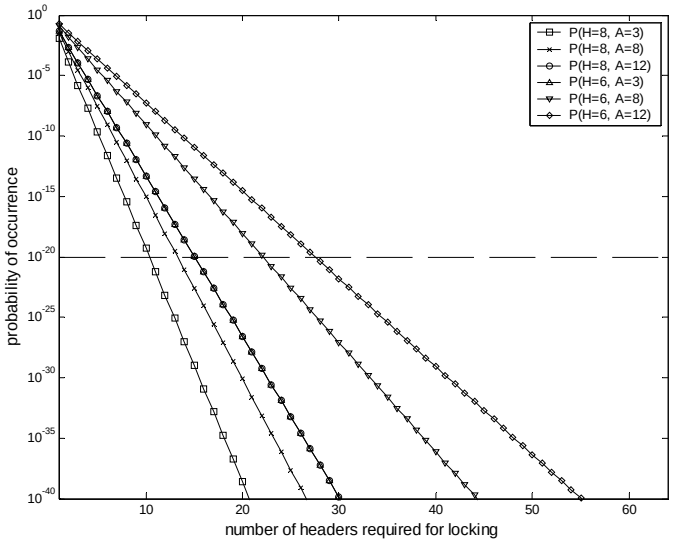

Figure 47: Impact of accepting different numbers of header patterns A on the probability of observing valid header sequences in the RS coded field.

It is important to allow for errors falling on the header even when trying to lock. Once a possible header position is found, the position should be assumed to be wrong only after encountering two incorrect header patterns. This follows from the fact that the probability of locking to the RS coded field (~ 10^{-20}) is much lower than the probability of an SEU falling on the header bits (~ 10^{-10}).

5.4.3 Loss-of-Lock mechanism

Once the locked state is reached, each frame header is verified for correctness to confirm locking. If many header patterns (more than X_2) are found to be incorrect within X_3 successive frames, then it is very likely that the receiver is locked to an incorrect frame phase, and thus the state machine should return to the OOL state and start searching for the correct frame phase. Given the fact that the header can be received with errors even when lock is correct (e.g. due to SEUs on the photodiode), some incorrect headers have to be tolerated within X_3 contiguous ones before declaring lock lost: as the locking mechanism causes many data packets to be lost while searching for the correct frame phase, it has to be initiated only if necessary.

Given a raw BER α on the link, the probability of having one bit error in a header is $P_{EH} = \alpha\, H/N_{tot}$ (a fraction H/N_{tot} of erroneous bits on average belongs to the header, where H is the header bit-length and N_{tot} is the frame bit-length). The probability of having T errors contained in any of X_3 continuous headers is given by:

$$P(T\ err.\ in\ X_3\ headers) = \binom{X_3}{T}\left(\frac{H}{N_{tot}}\alpha\right)^T\left(1-\frac{H}{N_{tot}}\alpha\right)^{N_{tot}-T} \cong \binom{X_3}{T}\left(\frac{H}{N_{tot}}\alpha\right)^T$$

In a worst case scenario of BER = 10^{-9} and for example $X_3 = 64$, the probability of occurrence of two errors is 10^{-17}, the probability for three errors is 10^{-26} and for four errors it is 10^{-35}. Consequently, it seems reasonable to choose $X_2 = 4$ as maximum number of incorrect header patterns to be accepted before initiating the locking procedure (movement to the OOL state from the IL state).

5.4.4 Header error tolerance

Errors can corrupt the header as well as the data field: the probability that an error corrupts any bit of the header is $\alpha\, H/N_{tot}$. Errors on the header though are not corrected by the error correction scheme. For a better

reliability of the link transmission, when the state machine is in the IL state, if the header is found to be possibly corrupted by one SEU event, it should be recognized as such and the packet information should anyway be passed on as valid. For this reason, the header is said to be “error tolerant”.

At the same time, though, the counter that keeps track of the amount of incorrect received headers should be incremented in such cases so that the possibility of lock loss is still open. This counter is incremented as well in case the header is not recognized as either a header pattern, or a header at Hamming distance one from them; in this case, the received packet is not passed on as valid as its content type is not known. When X_2 counts are reached, the frame is considered locked to an incorrect phase and so the state machine goes back to the OOL state. If instead X_3 correct header counts are reached first, then both the counters are reset

In order to have error tolerant headers, the Hamming distance between two different header patterns has to be at least three bits, so that a single-bit error and a maximum likelihood decoding can still associate the corrupted pattern with the correct header and consequently packet type. Two header patterns can be distinguished with one bit of information, and this together with a minimum distance of 3, makes a total minimum header length of 3 bits. In case of needing to distinguish among three different header types, then five bits is the minimum total header length.

While a minimum header length would optimize link efficiency by keeping the amount of redundancy minimal, a header length that matches the modularity of the RS scheme in symbols is preferable as it eases clock frequency division. This is equivalent to saying that the header length should be chosen to be a multiple of the RS symbol width chosen for the code (denoted by the letter m in section 5.3).

Additionally, as the header is not scrambled, the different patterns are chosen to be DC-balanced and abundant in number of transitions, in order not to impair the line balance and in order to decrease the average and maximum run-length of the code.

5.5 Two interleaving options for the proposed line code

The line code for the VBD link and the GBT ASIC is proposed in two different options. The main difference among the two is the number of RS blocks in which the scrambled data is divided, i.e. the number of RS blocks

that are interleaved. Additionally, many other code parameters are changed following the choice of number of interleaved blocks.

The first option is presented in section 5.5.1, while the second one in section 5.5.2. Section 5.5.3 presents a summary of the options parameter choices.

5.5.1 First interleaving option (L = 2, m = 4)

The first option is based on interleaving two RS blocks (L = 2). For a packet of 64-bit user data, the length of each RS block is 32 bits. The minimum symbol width that can protect 32 bits is m = 4, with 8 information symbols and a total of 10 symbols per block: the 64-bit packet is RS encoded by adding 16 redundancy bits, i.e. 2 redundancy symbols per RS block.

Given a symbol width of 4 bits, the maximum number of information symbols per block is 15. Only 10 symbols are effectively used in the proposed code option, and 5 are zero-padded: this choice allows for additional error detection. In case the number of errors per block is beyond the error correction capability of the code, as for example in the case of two error events falling in different symbols, then the decoding algorithm responds by trying to correct anyway one error. Two different situations are then possible. Either the decoding algorithm fails while calculating, or it corrects one symbol error, which can be placed anywhere in the word, giving in general origin to a third symbol error. As in this option five symbols are zero padded, they are not transmitted, and they are considered zeros while being decoded. If the algorithm tries to correct any of these zero padded symbols, then it can be deduced that some mistake was made in the calculation, rather than in the transmission. This situation can be flagged as a decoder error and the packet can be invalidated in order not to deliver information that is know to be corrupted. The flag denoting decoding failure has to be kept active during two successive data frames, as also the one that follows the decoding failure is corrupted due to the self-synchronizing nature of scrambling performed after RS decoding.

In this option the choice of the scrambling polynomial has to be optimized for a data-word length of 64 bits. The order of the scrambler is then to be chosen lower than 64 in order not to require keeping memory of more than one previous data word, and in order not to multiply errors beyond one additional data frame. At the same time, the scrambler order n is

to be kept as high as possible in order to have better randomization capability. The scrambler of order 63 is chosen as it also minimizes the number of exclusive-or operations needed when implemented in a parallel structure.

Concerning the header, an 8-bit length is chosen as it is the first multiple of the chosen RS symbol width being longer than 5 bits. The choice of an 8-bit header has the additional advantage of easing clock frequency division as the total frame length becomes 88 bits, which can be divided by 8. A frequency division by a factor of 8 matches also the fact that the algorithm can operate in parallel on two RS symbols at a time (one per block), i.e. 8 bits. This allows operation of the RS encoding and decoding blocks at a clock period that is 8 times the line bit period.

Examples of header patterns are 01011010 and 10100101 in the case of distinction among two different headers and data types. On the contrary, in order to distinguish among three different data types, two possible choices of header pattern sets are reported in Table 8. In the first set (Table 8, left) the minimum distance between header patterns is 5, which allows the error tolerance of two bit-errors per header. The first set though also has the disadvantage that one of the patterns is not DC-balanced. The second set (Table 8, right) consists of three DC-balanced patterns, but has a minimum distance of 4, which allows tolerating only one bit error per header.

Table 8: Two sets of header patterns for H = 8. The first pattern (left) maximizes minimum Hamming distance, but contains one non-DC-balanced header. The second set (right) has smaller minimum distance but consists of only DC-balanced patterns.

	Header Pattern	Hamming Distance
1	01001010	$d_H = 5$, $d_H = 6$, $d_H = 5$
2	01010101	
3	10110010	

	Header Pattern	Hamming Distance
1	01010101	$d_H = 4$, $d_H = 8$, $d_H = 4$
2	01011010	
3	10100101	

A scheme representing the modifications the data goes through when coded according to the first option is shown in Figure 48.

The total frame length is 88 bits and a total link speed is $f_{bit} = 88 \cdot 40$ MHz = 3.52 GHz (at the 40 MHz LHC frequency). The overall line code efficiency is 72.7%.

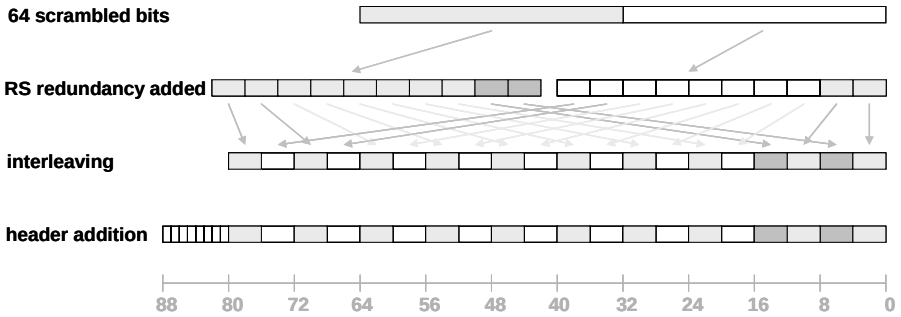

Figure 48: Modification of the data word through the line encoding building blocks for the first code option. The 64-bit user data packet is scrambled, then it is considered as divided in two blocks of 32 bits each. These are encoded through a RS scheme characterized by 4-bit symbol width, 8 information symbols and 10 symbols in total per block. Consistently with the fact that the code is implemented to correct one symbol error per block, two redundancy symbols (8 bits in total) are added to each of the two blocks. The two blocks are then interleaved, so that one symbol from one block is followed by one symbol from the other block. Last, an 8-bit header is appended.

5.5.2 Second interleaving option (L = 4, m = 3)

The second line code option is based on four-block interleaving (L = 4), as opposed to the first option which is based on L = 2 interleaved RS blocks.

In order to cover 64 bits with L = 4, four 16-bit blocks would be required. That would imply m = 4, and an added redundancy of 2mL = 32 bits for an overall efficiency of 64/(64+32+8) = 61.5%, which is extremely low. It is rather chosen to shorten the data packet to a 60-bit length, which allows for 3-bit RS symbol width. In this second case, the added redundancy becomes 2mL = 24 bits, and the code efficiency R = 60/(60+24+6) = 66.7%. In the case of m = 3, though, no zero padding is used, so that no extra error detection is available.

For a 60-bit data packet, and with reference to Table 6, an order n = 60 for the scrambler satisfies all the constraints enumerated previously and is well matched to a parallel implementation.

A 6-bit header pattern is sufficient to match both RS symbol width and minimum header length for error tolerance. A possible set of header patterns is shown in Table 9, along with the Hamming distance calculated between patterns. Table 10 shows a few examples of possible header patterns corrupted by SEUs. While the first column features the received header

pattern, the second column contains the valid header that is at Hamming distance 1-bit from the received one.

Table 9: Set of three headers for the case H = 6. All headers are DC-balanced and a Hamming distance of 4 bits among them guarantees tolerance of one bit error per header.

	Header Pattern	Hamming Distance
1	010101	$d_H = 4$ $d_H = 4$ $d_H = 4$
2	101001	
3	011010	

Table 10: Examples of different received header patterns and how they are handled.

received	corresponding to
110101	010101
011101	010101
111010	011010
011110	011010
101101	101001
101111	101001

A scheme representing the modifications the data goes through when coded according to the second option is shown in Figure 49, to be compared with Figure 48 that referred to the first option. The link speed is $f_{bit} = 90 \cdot 40\text{MHz} = 3.60$ GHz and the code efficiency is 66.7%.

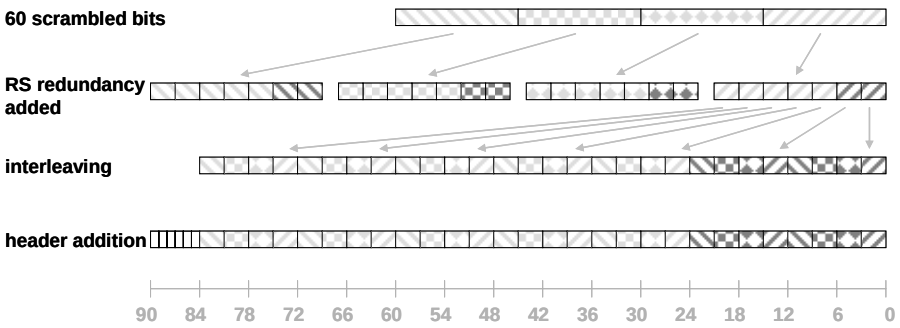

Figure 49: Modification of the data word through the line encoding building blocks for the second option. In this second option first 60-bit data are scrambled, then four 15-bit blocks are RS encoded (24 redundancy bits are added) and interleaved. A 6-bit header is then added, for a total 90-bit frame.

5.5.3 Comparison of the two code interleaving options

The summary of the parameters for the two options is given in Table 11. In the table the notation is consistent with the one used previously in the chapter: n is the scrambler order, m is the RS symbol width, L is the number of interleaved blocks, N_s is the number of symbols per RS block, out of which K_s are information symbols. N_b is the total number of bits after RS encoding, K_b of which are information bits. H is the header bit length and N_{tot} is the total frame bit length.

Table 11: Line code characteristics for the two presented options.

code characteristic	first option	second option
n	63	60
m	4	3
L	2	4
N_s	10	7
$N_b = N_sLm$	80	84
K_s	8	5
$K_b = K_sLm$	64	60
H	8	6
$N_{tot} = N_b + H$	88	90
$R = K_b / N_{tot}$	72.7%	66.7%
$f_{bit} = 1 / T_{bit}$ (GHz)	3.52	3.6

The efficiency R is the ratio between the number of information bits and total frame length. The frequency f_{bit} is the inverse of the bit period T_{bit} and corresponds to the delivery of N_{tot} bits every LHC clock period (T_{LHC} = 25ns, f_{LHC} = 40 MHz).

5.6 Summary

This chapter presents the line code proposed for the VBD link and the GBT ASIC. An overview of the line code building blocks is given: in the encoding scheme, scrambling is first implemented and then followed by RS error correction. This order is chosen as inverting the two blocks would cause error multiplication requiring a stronger error correcting scheme. A header is finally added to the packet for frame locking purposes, while allowing distinction among different data packet types (e.g. data, trigger and idle packets). The line code is presented in two different options which are characterized by different interleaving choices for the RS scheme. These two options result in different efficiencies, but at the same time achieve different error correction capability (details in the following chapter).

Chapter 6

Code performance & hardware requirements

This chapter carries out a detailed study on the properties of the line code for the GBT ASIC. The error correction performance is studied in 6.1; the DC-balance performance is studied in 6.2; the average and maximum run-lengths are derived in 6.3. The implementation cost of such line code is estimated in 6.4, for both the encoder and the decoder. The studies are carried out for the two code options considered.

6.1 Error correction performance

The line encoding scheme proposed for the GBT ASIC allows the correction of any single event upset per frame, even if the event extends over two consecutive bits. Single bit errors in the RS segment are corrected by the RS error correction scheme. Two-bit long errors are also corrected as two consecutive bits belong either to the same RS symbol, or to two different symbols which can anyway be corrected as they belong to different RS blocks due to interleaving. Single and two-bit long errors on the header can be tolerated if the header pattern choice is careful.

As interleaving of RS blocks is used, increased error correction capability is obtained with respect to the case of single bit error correction, as also independent double errors per frame can sometimes be corrected depending on the position in which they fall in the frame. Two errors can be tolerated if one falls on the header and one on the RS segment. In case the two errors fall on the RS segment of the frame, then two cases must be distinguished. If they fall in different RS blocks, they can be corrected. If they fall in the same RS block, but not in the same symbol, then they cannot be corrected

and the packet is corrupted. Detailed calculations and proofs follow, assuming 1-bit long errors only for simplicity and due to the fact that they are more likely than errors spreading over two consecutive bits (at least ten times more likely even for lower optical powers, see 3.3.2).

6.1.1 Two errors per frame

The probability α of having one bit error in the link is defined as the Bit Error Rate (BER). According to the theory of Bernoulli trials [Pap91], the probability of having T errors falling in the same frame composed of a total number of bits N_{tot} is:

$$P(\text{T errors in one frame}) = \binom{N_{tot}}{T} \alpha^T (1-\alpha)^{N_{tot}-T},$$

where the binomial coefficient denotes the number of subsets of T-elements in a total set of N_{tot} elements:

$$\binom{N_{tot}}{T} = \frac{N_{tot}!}{T!(N_{tot}-T)!}.$$

In particular, in the case of T = 2 errors per frame, the error probability takes the following form, with $\alpha << 1$:

$$P(2 \text{ errors in one frame}) = \binom{N_{tot}}{2} \alpha^2 (1-\alpha)^{N_{tot}-2} \cong \binom{N_{tot}}{2} \alpha^2 = \frac{N_{tot}^2 - N_{tot}}{2} \alpha^2.$$

In the case of three errors per frame the probability is proportional to α^3, in the case of four errors per frame the proportionality is with α^4 and so on. Being $\alpha << 1$ ($\alpha \sim 10^{-10}$ in section 3.3.3), the events characterized by T > 2 are much less likely to happen than the event comprising only two errors per frame (i.e. for $\alpha \sim 10^{-10}$: $P_2 \sim 10^{-17}$, $P_3 \sim 10^{-25}$, $P_4 \sim 10^{-34}$). Consequently the analysis that follows concentrates on the case of two errors per frame discarding the cases with more errors per frame.

Two-error events per frame can be grouped into three different categories, depending on the position in which they fall in the frame (refer to Figure 50):

A. two errors on the header: $\binom{H}{2} \alpha^2$;

B. one error on the header and one on the RS segment: $H\left(N_{tot}-H\right)\alpha^2 = HmN_sL\alpha^2$;

C. two errors on the RS segment: $\binom{N_{TOT}-H}{2}\alpha^2 = \binom{mN_sL}{2}\alpha^2$.

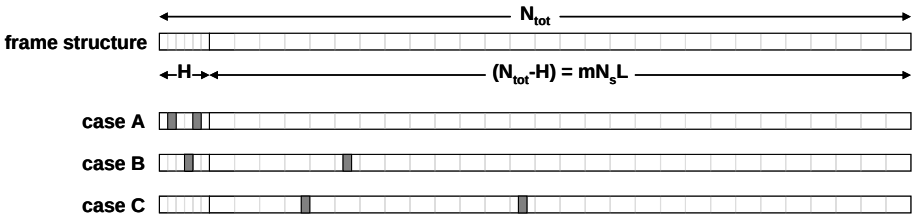

Figure 50: Allocation of frame bits (N_{tot}) to header or RS segment (top, for second option). Two errors (grey-shaded slots) per frame: division in case A, B or C depending on error events positions (bottom).

In the above and consistently with Table 11, H is the header bit-length, m is the RS symbol bit-length, N_s is the number of RS symbols per block, L is the number of interleaved RS blocks.

It can be verified that the three different categories add up to the total number shown before:

$$\binom{H}{2}\alpha^2 + H\left(N_{tot}-H\right)\alpha^2 + \binom{N_{tot}-H}{2}\alpha^2 = \binom{N_{tot}}{2}\alpha^2$$

This represents the probability of having two independent errors anywhere in the frame.

6.1.2 Two errors in the RS segment

In case two errors fall into the RS segment of the frame (case C above), two typologies of events can be distinguished (refer to Figure 51):

a. The probability that two errors fall in different RS blocks of a total of L blocks is: $m^2N_s^2\binom{L}{2}\alpha^2$;

b. The probability that two errors fall in the same RS block, for L different blocks: $L\binom{mN_s}{2}\alpha^2$.

Figure 51: Two errors falling on the RS segment: division in cases C.a or C.b (second option).

The two categories add up to the total of case C:

$$m^2 N_s^2 \binom{L}{2} \alpha^2 + L \binom{mN_s}{2} \alpha^2 = \binom{mN_s L}{2} \alpha^2$$

This represents the probability of having two independent errors anywhere in the RS segment.

6.1.3 Two errors in the same RS block

In the case that the two errors fall in the same RS block (case C.b), the events can be divided into two typologies of (refer to Figure 52):

I. The probability that two errors fall in the same symbol, for N different symbols, is: $N_s \binom{m}{2} \alpha^2$;

II. The probability that two errors fall in different symbols is: $m^2 \binom{N_s}{2} \alpha^2$

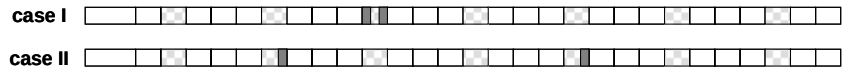

Figure 52: Two errors falling on the same RS block: division in cases C.b.I and C.b.II (second option).

Cases I and II add up to the total of case C.b:

$$N_s \binom{m}{2} \alpha^2 + m^2 \binom{N_s}{2} \alpha^2 = \binom{mN_s}{2} \alpha^2$$

This represents the probability of having two independent errors in any of the symbols within the same RS block.

6.1.4 Conclusions

The cases for which the line code can correct two independent bit errors are: B, C.a, C.b.I, which occur with probability:

$$\left(H\left(N_{TOT} - H\right) + m^2 N_s^2 \binom{L}{2} + \binom{m}{2} N_s L \right) \alpha^2$$

The cases which are uncorrectable for the line code are: A and C.b.II, which occur with probability:

$$\left(\binom{H}{2} + m^2 \binom{N}{2} L \right) \alpha^2$$

In other words, the first interleaving option, which interleaves two RS blocks as discussed in the previous chapter, can correct 62% of two events falling in the same frame while the second option can correct 81% as it interleaves four RS blocks. To be noted that the first option allows for additional error detection due to the use of zero-padding.

In the overall line code structure, case C.b.II causes the frame that follows the corrupted one to be incorrect as well, due to the error multiplication in the self-synchronizing scrambler. Consequently, the probability of error on the present frame can be calculated as the sum of two probabilities, one depending on errors on the present frame, and another one that depends on the errors occurring on the previous frame:

$$P(\text{frame error on current frame}) =$$
$$= P\left((A + C.b.II) \middle| 2 \text{ bit err. on current frame}\right) \cdot P(2 \text{ bit err. on current frame}) +$$
$$+ P\left((C.b.II) \middle| 2 \text{ bit err. on previous frame}\right) \cdot P(2 \text{ bit err. on previous frame}) =$$
$$= \left(\binom{H}{2} + m^2 L \binom{N}{2} \right) \cdot \alpha^2 + \left(m^2 L \binom{N}{2} \right) \cdot \alpha^2 = \left(\binom{H}{2} + 2m^2 L \binom{N}{2} \right) \cdot \alpha^2$$

Thus, for the first option:

$$P(\text{frame error}) = \left(\binom{H}{2} + 2m^2 L \binom{N}{2} \right) \cdot \alpha^2 = 2908 \cdot \alpha^2 .$$

For the second option:

$$P(\text{frame error}) = \left(\binom{H}{2} + 2m^2 L \binom{N}{2} \right) \cdot \alpha^2 = 1527 \cdot \alpha^2 .$$

Or equivalently, if seen as fraction of probability of two errors falling in one frame:

$$\frac{P(\text{frame error on current frame})}{P(\text{2 bit errors on current frame})} = \frac{\left(\binom{H}{2} + 2m^2 L \binom{N}{2} \right) \cdot \alpha^2}{\binom{N_{TOT}}{2} \cdot \alpha^2} =$$

$$= \begin{cases} 75.97\% \ (1^{\text{st}} \text{ intd. option}) \\ 38.13\% \ (2^{\text{nd}} \text{ intd. option}) \end{cases}$$

The line code error correction capability, then, improves the link performance from an uncoded transmission BER α to a coded Frame Error Rate FER = $3 \cdot 10^3 \cdot \alpha^2$ for the first interleaving option and $1.5 \cdot 10^3 \cdot \alpha^2$ for the second interleaving one (see Figure 53). This corresponds for example to an improvement from $\alpha = 10^{-10}$, i.e. ~1 error/s per link to FER = 3.8 errors/hour on 10000 links (as might be used in an experiment) for the first option or 1.9 errors/hour for the second option.

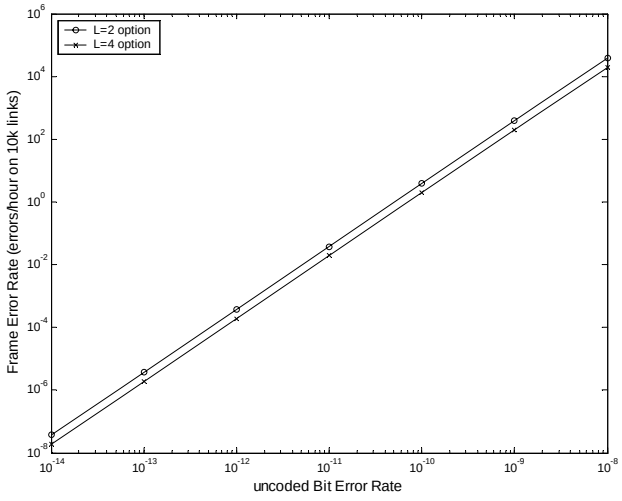

Figure 53: Frame error rate as function of uncoded link BER. A BER of 10^{-10} translates to a FER of ~1error/hour on 10000 links.

The BER after coding is proportional to the FER, but the proportionality is expressed by a factor that is difficult to evaluate: the error correction algorithm tries to correct the packet even in case more errors are present than what the error correction capability can handle. The outcome, then, is either a decoding failure (e.g. in case of inversion of a zero syndrome, so that the decoder cannot calculate the error pattern and "fails" in decoding) or correction of a random symbol (when decoding does not fail, correction is operated on any received symbol, many of which are in fact correct).

In 4.1.1 it is stated that for a code BER improvement to be worth the additional FEC complexity, the parameter $(t+1)\,R_c$ must be higher than one. This parameter is about 1.63 for the first interleaving option, and 1.75 for the second interleaving option.

6.2 DC-wander performance

An eye diagram can be used as a way to evaluate the performance of a digital transmission system by looking at the received signal quality.

The eye opening, when related to the signal noise, quantifies how likely it is to make an incorrect decision in the CDR circuitry: the wider the distance between high and low received values, the less likely the incorrect decision is. Additive noise and intersymbol interference are examples of causes of received signal degradation which result in eye closure.

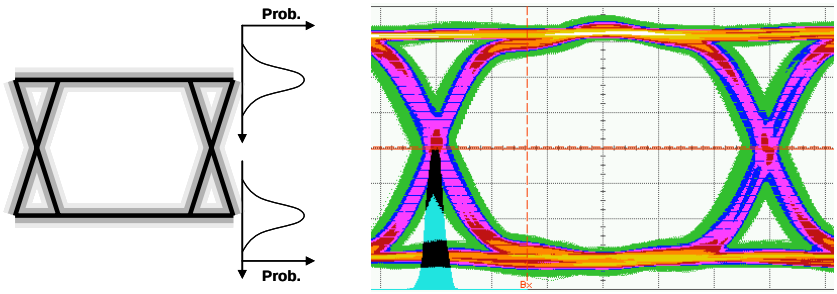

Figure 54: DC-wander affects the eye diagram by closing the eye opening. A sketch of an eye diagram is shown on the left, a real eye diagram in shown on the right.

DC-wander is the slow variation of the transmitted signal levels due to a null at DC of the transmitter/receiver frequency response. Like other impairments, DC-wander can cause eye closure. Its main effect is to spread the nominal high and low values to a certain distribution as sketched in

Figure 54 on the left. The amount of distribution spread effectively lowers the eye margin budget.

For example, in an AC-coupled optical receiver, the serial bit stream is high pass filtered by the amplification stages before reaching the decision threshold in the receiver CDR block. The amplifier output voltage drifts as a result of the low frequency cut-off, and this generates the so-called baseline wander in case the bit stream contains some low frequency or DC-component. DC-balanced signals are less affected by the low frequency cut-off since their spectral content is low near DC.

A simplified model can be used to interpret the effect of high-pass filtering the bit stream. If the high-pass filter is simplified to be a first order system, then a coupling capacitor is placed in series between the input voltage V_{bit} and the output voltage V_{out}, as drawn in Figure 55.

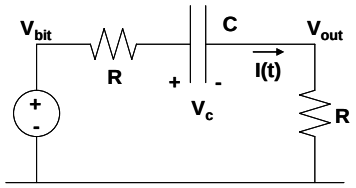

Figure 55: The data stream voltage is high pass filtered by a coupling capacitor before reaching clock and data recovery, approximation with a first order filtering.

The circuit represented in Figure 55 is characterized by a set of two equations:

$$\begin{cases} I(t) = \dfrac{1}{2R}\left(V_{bit}(t) - V_c(t)\right) \\ V_c(t+\Delta t) = I(t)\dfrac{\Delta t}{C} + V_c(t) \end{cases}$$

The above set can be solved for $V_c(t + \Delta t)$ and be used for a finite-timestep simulation of the filter:

$$V_c(t+T_{bit}) = V_c(t) + \frac{T_{bit}}{2RC}\left(V_{bit}(t) - V_c(t)\right).$$

This equation is valid if $T_{bit} << \tau = 2RC$, which is the case to be considered. The equation above is used to study the effects of the low frequency cutoff on the voltage V_c as a function of the data stream patterns and on the filter time constant $\tau = 2RC$. The voltage V_c is histogrammed in

order to analyze the DC-wander by studying the mean and standard deviation of the histogram. Examples of these histograms are reported in Figure 56, where the baseline wander (difference between V_c and the nominal voltage value, at zero) is reported on the x-axis, while its the probability of occurrence is plotted against the y-axis. The graphs are obtained for the following conditions: R = 50 Ω, C is varied between 0.1 μF and 0.01 nF, T_{bit} = 1/3.52 ns for the first option and T_{bit} = 1/3.60 ns for the second one.

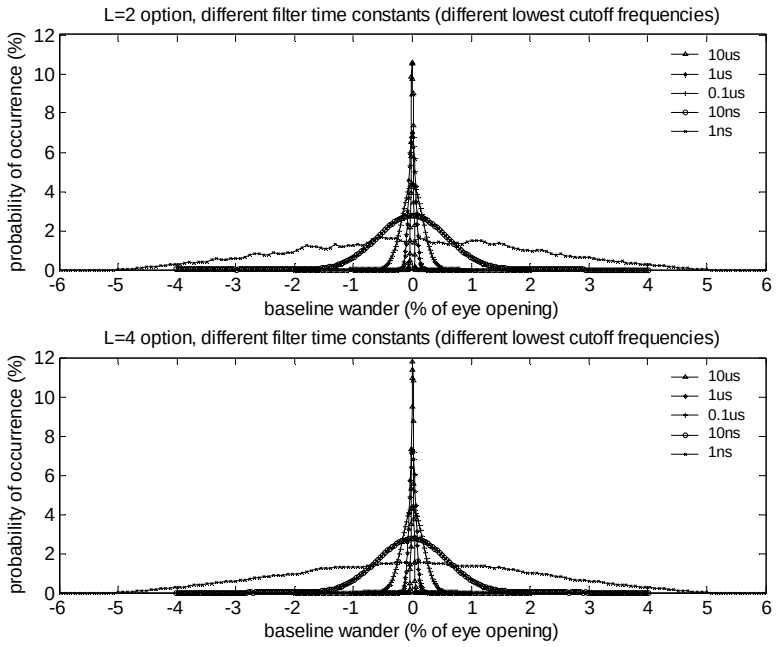

Figure 56: DC-wander probability depending on filter bandwidth (first option top, second option bottom).

In Figure 56, 5 Mbit of random data are encoded with the two different line code options (first interleaving option top graph, second interleaving option bottom graph) and filtered with 5 different RC filters. It can be seen that for the same data stream, the DC-wander depends highly on filter bandwidth. The values for signal standard deviation are reported in Table 12 for the two options and the filter time constants.

Table 12: Standard deviations for the histograms shown in Figure 56.

filter time constant [s]	σ ("L = 2" option)	σ ("L = 4" option)
10 μs	0.0019	0.0018
1 μs	0.0057	0.0055
0.1 μs	0.0180	0.0180
10 ns	0.0575	0.0579
1 ns	0.2124	0.2212

It is evident that the smaller the filter time constant (that is the higher the filter cutoff frequency ω), the larger the DC-wander. This can be qualitatively understood by seeing the filter as "observing" a data window of N-bit at a time, where the number of bits N depends on the ratio between the filter time constant τ (proportional to the product RC and inversely proportional to the filter cutoff frequency ω) and the bit period T_{bit}. As the bit period is fixed, a higher cutoff corresponds to a smaller time constant and consequently to an average over a smaller number N of bits.

The data stream obeys to "random walk"-like statistics, so that the expected wander over N bits is proportional to 1/sqrt(N). Thus, to a higher cutoff ω corresponds a smaller N, the standard deviation σ is at the same time expected to grow as 1/sqrt(N). The considerations above can be summarized as follows:

$$\omega \propto \frac{1}{\tau} \propto \frac{1}{RC}; N \propto \frac{\tau}{T_{bit}} \Rightarrow \sigma(N) \propto \frac{1}{\sqrt{N}} \propto \sqrt{\frac{T_{bit}}{\tau}} \propto \sqrt{T_{bit}\omega}$$

In fact in Table 12 an increment of a factor 10 in DC-wander sigma corresponds to an increment of a factor 100 in the cutoff frequency.

The behaviour of four different data types are studied as well for both options on a 5 Mbit data stream and a 100 KHz cutoff frequency. Four data types are considered consisting of random data, all-zeros, all-ones and idle frames. The four data types have the same baseline wander behaviour, as expected due to the presence of the scrambler. The histograms are reported in Figure 57 (first option top, second option bottom) and they clearly show that the DC-wander is qualitatively the same in the four cases.

The worst case transmission consisting of the user sending data mimicking the inverse function of the scrambler is not studied as it is very unlikely to happen (as already discussed in 5.1).

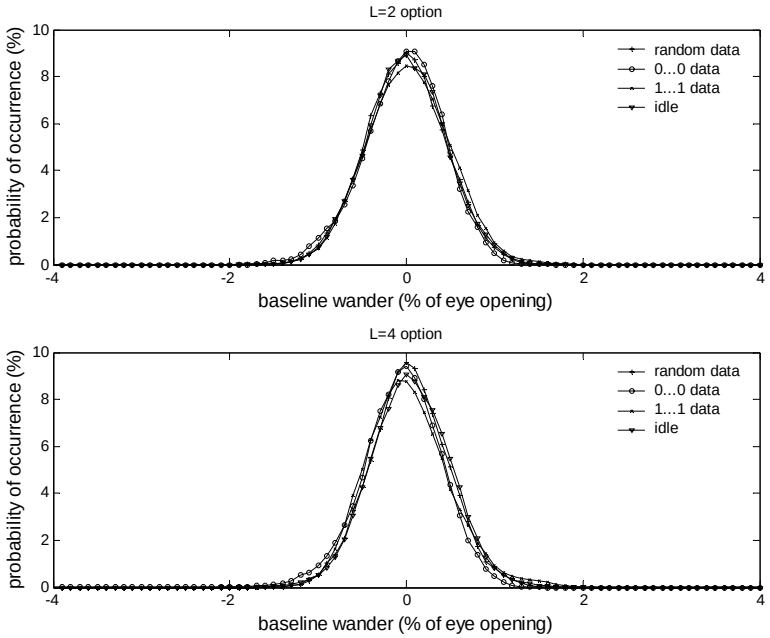

Figure 57: Baseline wander probability of occurrence.

Both options and all four data types show qualitatively the same wander, with means of the order of 0.01% of signal amplitude, and maximum σ of 0.47%. This value for σ is consistent with the values reported in Table 12, taking into account a factor sqrt(2π) that is due to the difference between expressing the cutoff as time constant τ or frequency f:

$$\omega = \frac{1}{\tau} \quad \text{and} \quad f = \frac{\omega}{2\pi} = \frac{1}{2\pi\tau}.$$

For comparison, purely random data treated in the same fashion result in σ = 0.0048 (or 0.48%) and mean consistent with the values obtained from the proposed line code. This proves that the impact of RS encoding and header addition on DC-wander is minimal, if not helpful.

The simulated amount of DC-wander is easily manageable in the system as it has negligible impact on the signal-to-noise ratio compared to other noise sources.

6.3 Average run-length performance

The *run length* is defined as the number of identical contiguous symbols which appear in the signal stream. For a binary code, the run length is the number of contiguous ones or zeros after encoding. Average and maximum run-length, then, measure the likeliness of occurrence of data transitions. A sample data stream and few run-lengths are shown in Figure 58.

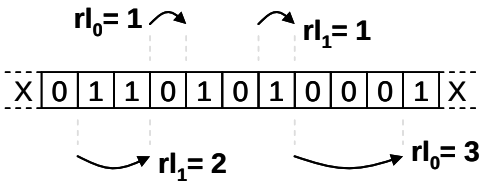

Figure 58: Examples of run-length evaluation. Run-lengths of 1, 2 and 3 bits are shown for a sample data stream. In total 4 1-bit, 1 2-bit and 1 3-bit run-lengths can be counted, for an average run-length of 1.5 bits.

In the case of purely random data, the probability of occurrence for run-lengths equal to one, two and three bits are:

$P(rl = 1) = P(01) + P(10) = 1/4 + 1/4 = 1/2,$

$P(rl = 2) = P(001) + P(110) = 1/8 + 1/8 = 1/4,$

$P(rl = 3) = P(0001) + P(1110) = 1/16 + 1/16 = 1/8.$

And in general, the probability of having a run-length equal to n bits is: $P(rl = i) = (1/2)^i$.

The maximum run-length for a stream of random data is not limited, and the average run-length $E\{rl\}$ is equal to:

$$E\{rl\} = \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i i = 2 .$$

In the case of the code proposed for the GBT ASIC, only part of the data stream can be treated as random, that is the RS segment, as opposed to the header that is deterministic. A model though can still be put in place for predicting the average run-length performance of the code. The model is presented for the simplified case in which only two headers are distinguished, and is later confirmed by simulation results.

6.3.1 Run-length theoretical model for the two options

In the first proposed option the header length is H = 8 bits, and the two chosen patterns are 01011010 and 10100101, which suffice for distinguishing data or idle packets (refer to section 5.3 for details).

For a frame, the probability of occurrence of 1-bit and 2-bits run-lengths is composed by two parts: a few transitions are certain due to the presence of the header, while a second part behaves as random data (as discussed before), so that for this segment the run-length is statistically determined. For the chosen header patterns, five elements have 1-bit run-length, and one element has 2-bit run-length. Concerning the rest of the data field, depending on the run-length under study, a different number of bits have to be taken into account (see Figure 59). For example all the RS segment bits, plus the right header boundary bit, have probability 0.5 of generating a 1-bit run-length. In the case of 2-bit run-length, all the RS segment bits, plus the right header boundary bit contribute to possibly generating such a run-length: the last bit of the sequence though, being the "boundary", just before the header, contributes with a different probability as opposed to all others belonging to the set (1/2 as opposed to 1/4, as can be seen in Figure 59). For each additional bit of run-length, one less bit belongs to the ones which can generate such a run-length, and the last of the set has double probability compared to the previous ones (refer to Figure 59). Thus:

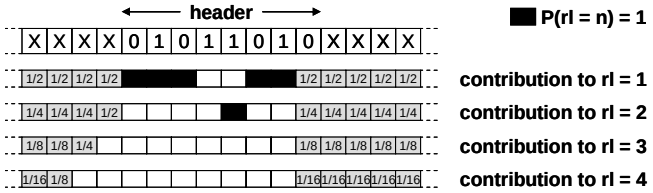

Figure 59: Participation of stream bits to run-length probability evaluation (first option). Dark bits contribute with probability one, grey bits contribute with probabilities that are powers of 1/2.

$P(rl = 1) = P(rl = 1|\text{header}) + P(rl = 1|\text{RS segm}) = (5 + 81 \times 1/2) / rl_{TOT} = 0.5260,$

$P(rl = 2) = P(rl = 2|\text{header}) + P(rl = 2|\text{RS segm}) = (1 + 80 \times 1/4 + 1/2) / rl_{TOT} = 0.2486,$

$P(rl = 3) = P(rl = 3|\text{header}) + P(rl = 3|\text{RS segm}) = (0 + 79 \times 1/8 + 1/4) / rl_{TOT} = 0.1142,$

$P(rl = 4) = P(rl = 4|\text{header}) + P(rl = 4|\text{RS segm}) = (0 + 78 \times 1/16 + 1/8) / rl_{TOT} = 0.0564,$ etc.

where: $rl_{TOT} = \sum_{i=1}^{\infty} events(rl = i)$.

Concerning the second option the header width is H = 6 and the chosen patterns are 010110 and 101001. considerations similar to the first option for building the run-length theoretical model can be done. Thus (refer to Figure 60):

$P(rl = 1) = P(rl = 1|\text{header}) + P(rl = 1|\text{RS segm}) = (3 + 85 \times 1/2) / rl_{TOT} = 0.5200,$

$P(rl = 2) = P(rl = 2|\text{header}) + P(rl = 2|\text{RS segm}) = (1 + 84 \times 1/4 + 1/2) / rl_{TOT} = 0.2400,$

$P(rl = 3) = P(rl = 3|\text{header}) + P(rl = 3|\text{RS segm}) = (0 + 83 \times 1/8 + 1/4) / rl_{TOT} = 0.1214,$

$P(rl = 4) = P(rl = 4|\text{header}) + P(rl = 4|\text{RS segm}) = (0 + 82 \times 1/16 + 1/8) / rl_{TOT} = 0.0600,$ etc.

With: $rl_{TOT} = \sum_{i=1}^{\infty} events(rl = i)$.

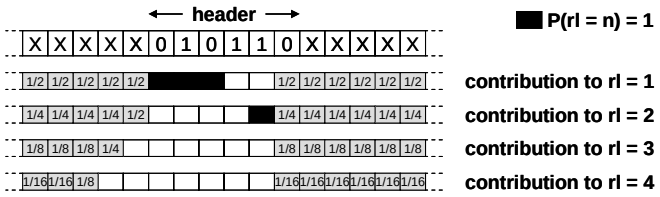

Figure 60: Participation of stream bits to run-length probability evaluation. Dark bits contribute with probability one, grey bits contribute with probabilities that are powers of 1/2 (second option).

The two models predict an average run-length of 1.93 bit for the first option and 1.96 for the second option.

The maximum run-length is strictly limited by the code structure, in particular due to the presence and structure of the header. The longest possible sequence of alike data happens when the scrambler output is the all-zero word. This is afterwards encoded by the RS scheme to an all-zero output, as RS codes are linear. This means that the maximum run-length is 82 bits for the first option and has a probability of occurrence of $1/2^{64} \sim 10^{-20}$. For the second option the maximum run-length is 86 bits which has a probability of occurrence of $1/2^{60} \sim 10^{-18}$.

6.3.2 Simulation results for the two options

The line code performance is evaluated on the basis of simulation data using the header patterns introduced in the previous section. For the two options, the transmission of 5Mbit of data has been simulated for four types of data: idles, random data, all-zeros word, all-ones word. The probabilities of occurrence of n-bit run-lengths for the different cases considered are

shown in Figure 61 and Figure 62. The probabilities of occurrence of 1, 2, 3 or 4-bit run-lengths are given in Table 13 and Table 14.

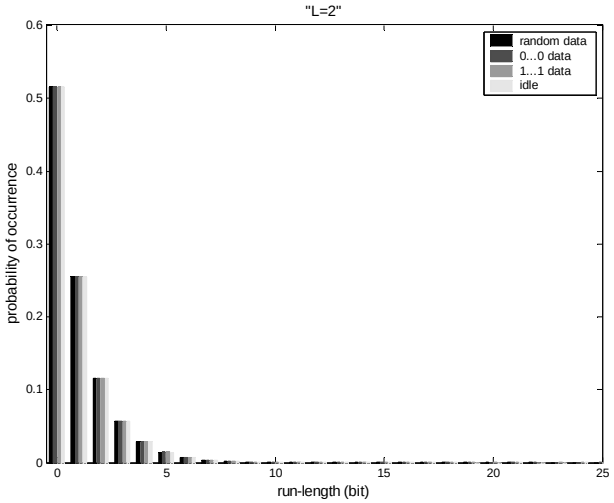

Figure 61: Run-length probability on 5 Mbit of four types of input data (first option).

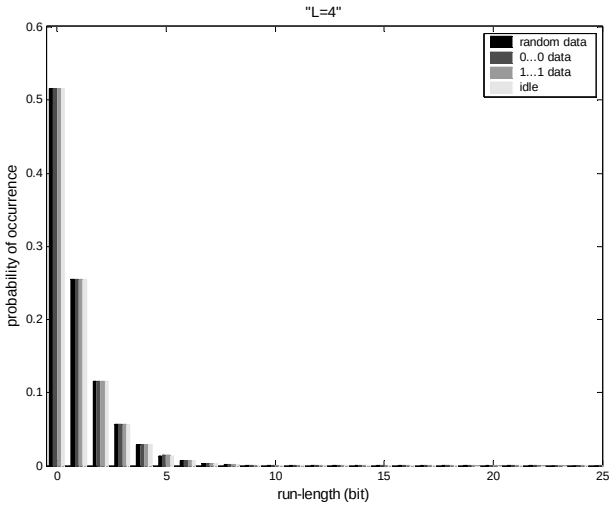

Figure 62: Run-length probability on 5 Mbit of four types of input data (second option).

Table 13: Probabilities of occurrence of different run-lengths (1, 2, 3 and 4 bits) for the L = 2 option and four different data types. Average values per option are reported in the last row.

	P(rl = 1)	P(rl = 2)	P(rl = 3)	P(rl = 4)
rn	0.5375	0.2448	0.1102	0.0544
0…0	0.5374	0.2452	0.1100	0.0542
1…1	0.5374	0.2450	0.1100	0.0545
idle	0.5374	0.2448	0.1103	0.0545
avg	0.5374	0.2449	0.1101	0.0544

Table 14: Probabilities of occurrence of different run-lengths (1, 2, 3 and 4 bits) for the L = 4 option and four different data types. Average values per option are reported in the last row.

	P(rl = 1)	P(rl = 1)	P(rl = 1)	P(rl = 1)
rn	0.5161	0.5161	0.5161	0.5161
0…0	0.5159	0.5159	0.5159	0.5159
1…1	0.5163	0.5163	0.5163	0.5163
idle	0.5164	0.5164	0.5164	0.5164
avg	0.5162	0.5162	0.5162	0.5162

The average run-length calculated from the simulation data are reported in Table 15.

Table 15: Average run-lengths for the two options and different data types.

data type	L = 2	L = 4
rn	1.8929	1.9364
0…0	1.8931	1.9372
1…1	1.8927	1.9358
idle	1.8930	1.9339
average	1.8929	1.9358

The model can predict the average run length with a 2% error for the first option and 1% error for the second one.

It can be concluded that the presented code has better performance compared to random data with respect to run-length issues. The average run-length is slightly lower (of a factor of about 5%), and the maximum run-length is strictly limited, whereas for random data it is not. In other words, the presence of the header enhances the probability of observing 1-bit run-lengths as opposed to run-lengths higher than 2 bits. As a comparison, the 8b/10b code, while 64b/66b has a maximum run-length of 65 bits.

6.4 Implementation details

This section discusses implementation details and complexity of the blocks performing line encoding and decoding. A short discussion about the

complexity of carrying out multiplication operations in Galois Fields is presented in section 6.4.1. A discussion on transmitter and receiver building blocks follows, in section 6.4.2 and section 6.4.3 respectively. Finally, results on the complexity of the implementation are given, as of number of cells and expected power consumption (section 0).

6.4.1 Multiplication in Galois fields

As introduced in section 4.2.1, depending on the notation chosen for Galois field elements, either addition or multiplication operations become more complex. In particular, in the case of polynomial notation, multiplication between two terms is the most computationally intensive operation to be computed.

For example for the Galois fields used in the two options (4-bit or 3-bit symbol widths), general multiply modules have the following structures.

For m = 4:

$y_0 = (a_0 \,\&\, b_0) \,\hat{}\, (a_1 \,\&\, b_3) \,\hat{}\, (a_2 \,\&\, b_2) \,\hat{}\, (a_3 \,\&\, b_1)$;

$y_1 = (a_0 \,\&\, b_1) \,\hat{}\, (a_1 \,\&\, b_0) \,\hat{}\, (a_2 \,\&\, b_2) \,\hat{}\, (a_3 \,\&\, b_1) \,\hat{}\, (a_3 \,\&\, b_2) \,\hat{}\, (a_1 \,\&\, b_3) \,\hat{}\, (a_2 \,\&\, b_3)$;

$y_2 = (a_0 \,\&\, b_2) \,\hat{}\, (a_1 \,\&\, b_1) \,\hat{}\, (a_2 \,\&\, b_0) \,\hat{}\, (a_2 \,\&\, b_3) \,\hat{}\, (a_3 \,\&\, b_2) \,\hat{}\, (a_3 \,\&\, b_3)$;

$y_3 = (a_0 \,\&\, b_3) \,\hat{}\, (a_1 \,\&\, b_2) \,\hat{}\, (a_2 \,\&\, b_1) \,\hat{}\, (a_3 \,\&\, b_0) \,\hat{}\, (a_3 \,\&\, b_3)$.

For m = 3:

$y_0 = (a_0 \,\&\, b_0) \,\hat{}\, (a_1 \,\&\, b_2) \,\hat{}\, (a_2 \,\&\, b_1)$;

$y_1 = (a_0 \,\&\, b_1) \,\hat{}\, (a_1 \,\&\, b_0) \,\hat{}\, (a_2 \,\&\, b_1) \,\hat{}\, (a_2 \,\&\, b_2) \,\hat{}\, (a_1 \,\&\, b_2)$;

$y_2 = (a_0 \,\&\, b_2) \,\hat{}\, (a_1 \,\&\, b_1) \,\hat{}\, (a_2 \,\&\, b_0) \,\hat{}\, (a_2 \,\&\, b_2)$.

In the above, a and b are the input symbols, y is the output symbol, the subscripts i, $0 \leq i < m$, indicate the bits forming the symbols, "&" is an "and" operation and "^" is an exclusive-or operation

If either the multiplicand a, or the multiplier b, are constants, then the above operations are simplified as it is known a priori which a_i, or b_i, are zero or one. This eliminates the need for the "and" operations, and in general it diminishes the number of exclusive-or operations that are needed for the computation.

For example, in case of multiplication by α^3, then for m = 4, $b_3b_2b_1b_0$ equals 1000 and for m = 3, $b_2b_1b_0$ equals 011, thus:

For m = 4: $y_0 = a_1$; $y_1 = a_1 \wedge a_2$; $y_2 = a_2 \wedge a_3$; $y_3 = a_0 \wedge a_3$;

For m = 3: $y_0 = a_0 \wedge a_2$; $y_1 = a_0 \wedge a_1 \wedge a_2$; $y_2 = a_1 \wedge a_2$.

6.4.2 Transmitter

The transmitter is formed by four main blocks: the scrambler, the RS encoder, the serializer and control logic that regulates the data flow. These main blocks are sketched in Figure 63. K_b bits enter in parallel, are scrambled and RS encoded. After encoding the codeword is an N_b-bit parallel. Finally the H-bit header is added and the whole frame is serialized. The control box receives control inputs (for example the data packet type) and provides controls to the other blocks.

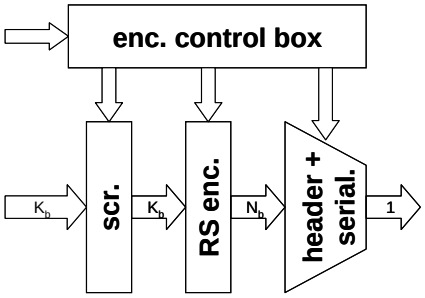

Figure 63: Transmitter blocks. On the data path, scrambling is followed by header addition and serialization. A control box contains all the logic required to control the encoding and serialization process.

6.4.2.1 Scrambler

Scrambling is introduced in section 5.1. In polynomial notation, the scrambler characteristic equation for the first option is written as:

$$S_i = D_i + S_{i-63} + S_{i-62}.$$

The above equation is a series operation which for each clock cycle evaluates the i-th scrambled term by combining the i-th data bit, and two previously calculated scrambled bits. The expression can be manipulated to generate the equivalent 64-bit parallel implementation ($K_b = 64$), where the 64-bit new scrambled word is $S_i \ldots S_{i+63}$:

$$\begin{cases} S_i = D_i + S_{i-63} + S_{i-62} \\ S_{i+1} = D_{i+1} + S_{i-62} + S_{i-61} \\ S_{i+2} = D_{i+2} + S_{i-61} + S_{i-60} \\ \vdots \\ S_{i+61} = D_{i+61} + S_{i-2} + S_{i-1} \\ S_{i+62} = D_{i+62} + S_{i-1} + S_i = D_{i+62} + S_{i-1} + D_i + S_{i-63} + S_{i-62} \\ S_{i+63} = D_{i+63} + S_i + S_{i+1} = D_{i+63} + D_i + S_{i-63} + D_{i+1} + S_{i-61} \end{cases}$$

In order to implement this set of equations, the 64-bit data are required ($D_i \ldots D_{i+63}$), and the previous 64-bit scrambled word ($S_{i-63} \ldots S_{i-1}$).

The last two equations of the set also involve the newly calculated scrambled bits, S_{i+62} and S_{i+63}. This can be avoided as shown by the last equalities in the set for S_{i+62} and S_{i+63} with only a small increase in the complexity of the computation. The small number of exclusive-or operations required by an order 63 scrambler compared to those of order e.g. 60, 62 or 64 motivates the choice of that order. However, the serial implementation has the advantage of working at the word rate instead of at bit rate.

While the serial implementation of the self-synchronizing scrambler requires only 63 memory elements, and a three-input exclusive-or, the parallel implementation requires 63 memory elements and 62 three-input exclusive-or and two five-input exclusive-or.

The same considerations lead to the choice of an order n = 60 for the 60-bit parallel implementation required for the first option ($K_b = 60$). The resulting set of equations follows:

$$\begin{cases} S_i = D_i + S_{i-60} + S_{i-59} \\ S_{i+1} = D_{i+1} + S_{i-59} + S_{i-58} \\ S_{i+2} = D_{i+2} + S_{i-58} + S_{i-57} \\ \vdots \\ S_{i+57} = D_{i+57} + S_{i-3} + S_{i-2} \\ S_{i+58} = D_{i+58} + S_{i-2} + S_{i-1} \\ S_{i+59} = D_{i+59} + S_{i-1} + S_i = D_{i+59} + D_i + S_{i-1} + S_{i-60} + S_{i-59} \end{cases}$$

In Figure 64 the structure of the 64-bit parallel 63-bit order scrambler (for the first option) and 60-bit parallel 60-bit order scrambler (for the second option) are shown.

For the two scrambler implementation 2 K_b memory elements (1-bit flip-flops) are required together with K_b 3-input exclusive-ors. Alternatively, few 3-input xors (two in the first option and one in the second one) can be substituted with 5-input xors. The calculation of D_{i+62} and D_{i+63} in the first option and D_{i+59} in the second one are the longest signal paths.

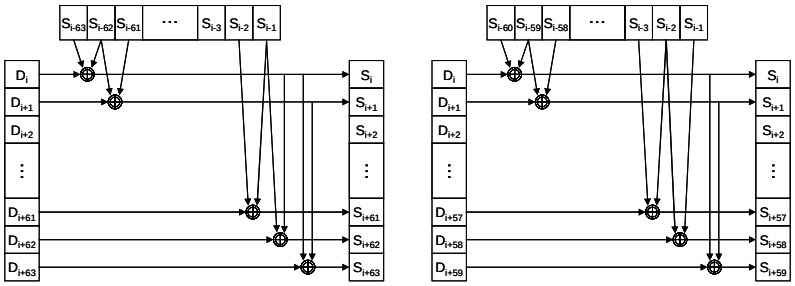

Figure 64: Order-63 64-bit parallel scrambler (left) and order-60 60-bit par. Scr. (rigth).

6.4.2.2 RS encoder

The RS encoder is based on the shift register structure as introduced in 4.2.2. The structure shown in Figure 33, is shown in Figure 65 for the choices made for the RS codes parameters. For both options of the code the number of errors corrected per block is t = 1, requiring the shift register to be constituted by only two memory elements.

Concerning the first option, the coefficients g_0 and g_1 in the generator polynomial $g(x) = g_0 + g_1x + x^2$ are respectively $g_0 = 8$ and $g_1 = 6$. For the second option, they are $g_0 = 3$ and $g_1 = 6$. The m-bit products are constituted in general of maximum m-input exclusive-ors (4 in the first option, 3 in the second one); in fact, for the RS choices made, the maximum number of exclusive-or inputs is three for both m = 4 and m = 3. The additions are 2-bit input xor operations. Consequently, the longest a signal has to go through is one 3-input xor and one 2-input xor.

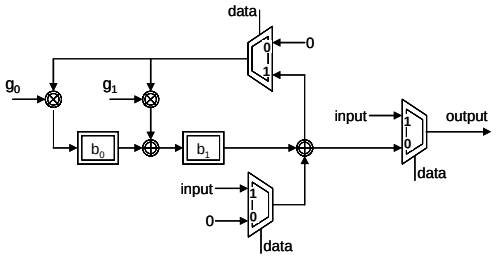

Figure 65: RS encoder for the first option, implemented through a shift register. The two factors (g0, g1) are equal to (8, 6) for the first option and (3, 6) for the second one. The signal "input" consists of the RS symbols constituting the input data block. On "output", for the first K_s clock cycles the data input is copied, for the last two clock cycles instead the redundancy symbols are present. The signal data enables data loading or syndrome outputting phases.

A control signal is needed for distinguishing between the first phase of encoding that consists of loading the K_s information symbols, and the second phase consisting of reading out from the shift register the two redundancy symbols.

L shift-register-based blocks as the one reported in Figure 65 are used in parallel in both options in order to encode the L different RS data blocks in which the data packet is divided. In the first option two shift-registers are used, in the second one four shift-registers are used. When writing the L shift-registers output to the output register of the whole RS encoder, the interleaving is performed, as depicted in Figure 48 and Figure 49.

6.4.2.3 Control logic

Given that $K_s + 2t = N_s$, N_s clock cycles are necessary for RS encoding. Another clock cycle is necessary for scrambling, thus $(N_s + 1)$ clock cycles are in total needed for line encoding: 11 clock cycles are needed for encoding in the first option, while 8 are needed in the second one.

For the first option, given that $N_{tot} = 88$, the line bit clock T_{bit} can be multiplied by a factor 8 in order to get to the 11 clock cycles per frame of the encoder clock: $T_{enc} = 8\ T_{bit} = 2.268$ ns and $f_{enc} = f_{bit} / 8 = 440.88$ MHz.

For the second option the solution is not as straightforward: $N_{tot} = 90$ and 8 encoder clock cycles T_{enc} are needed. Clock frequency division is easier if the factor is 6, which is both a RS symbol width multiple and the header

width. That gives 15 encoder clock cycles per frame (of which only 8 are needed): $T_{enc} = 6\ T_{bit} = 1.663$ ns and $f_{enc} = f_{bit} / 6 = 601.20$ MHz.

The mismatch implies that the logic has to go faster than what needed (which potentially implies power waste) and it is given by the fact that the RS segment has a factor of four multiplicity with the RS symbol width, while the header is only a factor of two multiple of the same width.

The addition of the header is the last step of the line encoding and it can be performed while loading the data in the serializer. The correct header for the chosen data type is selected by a dedicated control signal which keeps track of the data packet type.

All the control signals are generated centrally by a single control unit. Concerning the scrambler, this unit is responsible for enable and reset. The scrambler reset is needed only once at the beginning of the transmission in order to be able to control the state of the internal memory registers. The enable is active once per frame and controls the actual scrambling of the data packet.

Concerning the RS encoder, an enable signal is required, and a load signal, utilized for loading the scrambled data packet into the encoder, once per frame.

The control box reset is what starts the transmission by enabling the control state machine. This state machine loops cyclically through eleven states at frequency $f_{enc} = 1/T_{enc}$. In the first two states the scrambler is enabled and the RS encoder loaded, while the encoding process uses the remaining states. On the last states, it is verified if enough idles are sent to start the transmission. If so, data can be accepted for transmission.

Similarly in the second option, the control state machine loops cyclically through fifteen states after having been reset. The scrambler is enabled and the RS encoder loaded in the first two states. As the RS encoder needs less than the available states for performing the encoding, after seven states of operation it is put on hold by an appropriate control signal. In the last machine states, as in the previous case, the counter for idle signals and the loaded data type signal are updated.

The complete transmitter blocks with control signals are displayed in Figure 66.

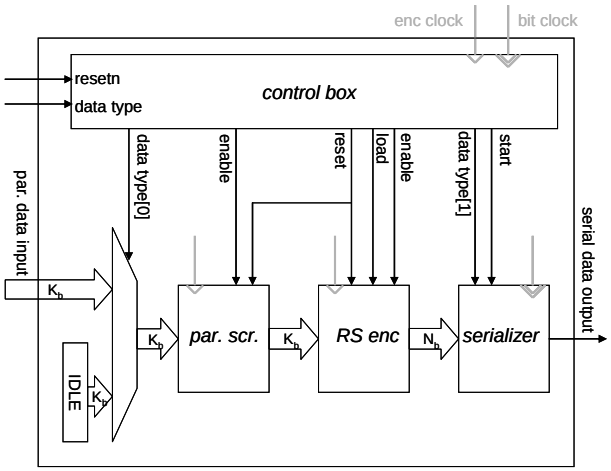

Figure 66: Transmitter diagram with control signals.

6.4.3 Receiver

The receiver is formed by four main blocks: the deserializer/frame synchronizer, the RS decoder, the descrambler, and control logic that regulates the data flow. These main blocks are sketched in Figure 67. The serial bit stream enters the frame synchronizer where the parallelization takes place and the packet type is recognized, removing the H-bit header. N_b parallel bits move from the frame synchronizer to the RS decoder. K_b RS decoded bits are finally descrambled, which concludes the line decoding. The control box groups the logic generating the control signals for the other blocks, and outputs information concerning the received data packet type.

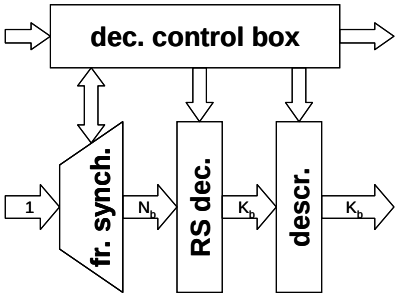

Figure 67: Main receiver blocks. On the data path, the serial bit stream is frame synchronized and the data packet type recognized. RS decoding is then followed by descrambling. A control box is defined as containing all the logic generating the control signals for the other blocks.

6.4.3.1 Frame synchronization

The frame synchronization state machine is discussed in section 5.3. Repeated recognition of a valid header pattern in a fixed position in the frame allows for frame locking, while repeated non-valid header recognition causes frame lock loss (refer back to Figure 78 and Figure 79).

The circuit receives the serial signal as an input, and, once locked, outputs a N_b-bit register containing the parallel RS segment. It is reset from the decoder reset, and apart from this, operates autonomously from the control logic block. It signals to the control logic block when lock is acquired and which header type was recognized for the present packet.

6.4.3.2 RS decoder

The RS code is based on the choice of correcting one symbol error per RS block, that is $t = 1$, and this noticeably simplifies the decoding algorithm. The equations that govern the decoding algorithm are discussed in section 5.2.

As discussed before, the algorithm itself can be divided into two parts: the first one consists of the syndrome calculation, while the second evaluates the error position α^j and the error amplitude e_j. If the two calculated syndromes are equal to zero, then the received RS block is an error free codeword and no error correction is needed, and thus the second part of the algorithm is not accessed.

The details of the implementation of the two parts of the decoding algorithm follow. All blocks are repeated for executing the algorithm on the L blocks in parallel. While executing the operations in parallel, the interleaving is also undone: while the N_b-bit word enters the RS decoder as interleaved data packets, the K_b bits are not interleaved anymore.

6.4.3.2.1 Syndrome evaluation

As already introduced in section 3.2.3 and used in section 5.2, the syndromes can be computed as:

$$S_i = r(\alpha^i) = r_0 + r_1\alpha^i + r_2\alpha^{2i} + \ldots + r_{n-1}\alpha^{(n-1)i} \text{ for } 1 \le i \le 2$$

The direct implementation of $S_i = r(\alpha^i)$ as in the formula above is schematized below, in Figure 68. The fact that the terms α^i are known simplifies the logic according to what anticipated in 6.4.1.

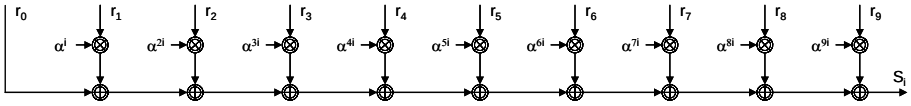

Figure 68: Implementation of $S_i = r(\alpha^i)$.

This implementation requires N_s-1 multipliers (to be implemented with exclusive-or operations) and a few of exclusive-ors for summing N_s m-bit addends to calculate one syndrome. Given the choice of $t = 1$ in the proposed scheme, only $2t = 2$ syndromes have to be evaluated, that is the structure shown in Figure 68 is to be implemented twice.

For SEU tolerance, triple voting can be inserted before the ex-or sums, so that each addend can be verified singularly. This choice requires a triplication of the logic that performs the multiplication, but the voting can be integrated with the rest of the operations without requiring one additional clock cycle.

A more compact choice can be obtained if the syndromes are evaluated taking advantage of Horner's rule [Bor95]. Horner's rule is a rule for polynomial computation which reduces the number of necessary multiplications: the powers of x are factored out, so that instead of calculating powers, only multiplications and additions are used according to the following formula:

$$a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 = (\ldots(a_n x + a_{n-1}) x + \ldots) x + a_0.$$

Applied to the syndrome calculation, the above gives the following:

$$S_i = r(\alpha^i) = (...(r_{n-1} \alpha^i + r_{n-2}) \alpha^i + \ldots) \alpha^i + r_0.$$

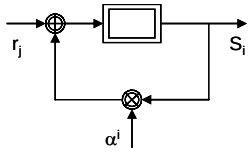

Figure 69: Implementation of S_i according to Horner's rule.

The above can be implemented with a structure as simple as the one in Figure 69, which only requires a GF sum and multiplication, plus an m-bit memory element. Concerning timing though, it requires a number of iterations equal to the degree of the polynomial (i.e. N_s), as opposed to one clock cycle of the previous solution. Triple voting in this case has to be

inserted at each of the N_s iterations (i.e. after the addition operation), or else the SEU induced error would be fed back.

Hybrid structures can be implemented as well, as for example a parallelization of Horner's rule as in Figure 70: little over twice the amount of logic is needed compared to Horner's rule implementation, but only half plus one clock cycles are required for completion.

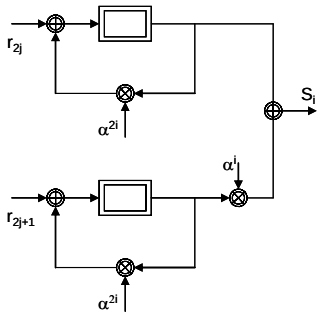

Figure 70: Hybrid implementation of the syndrome calculation.

6.4.3.2.2 Error position and amplitude evaluation

The second phase of the error correction algorithm consists of calculating the values for error position j and amplitude e_j from the syndromes values S_1 and S_2 according to the following:

$$\alpha^j = \frac{S_2}{S_1} \quad \text{and} \quad e_j = \frac{S_1^2}{S_2} \ .$$

The above calculations are performed through few simple calculations. The inversion of both S_1 and S_2, squaring S_1, multiplication of S_2 times S_1^{-1} and S_1^2 times S_2^{-1}. The inversion is implemented as a lookup table access, and it allows treating the division as a multiplication where both factors are unknown (as discussed earlier in section 1.4.1). These operations require two clock cycles in total. In the first clock cycle the inversions and the squaring are performed, in the second clock cycle the multiplications are performed (the multipliers are prepared in the first clock cycle).

A different rearrangement of the operations is possible which would result in lower amount of logic but additional latency (one more clock cycle): if inversion and squaring of S_1 are performed in the first period, inversion of S_2 and calculation of α^j in the second and calculation of e_j in the

third one, then only one lookup table and only one multiplier would be needed, while two of each are needed in the first proposed solution.

One additional clock cycle has to be counted in both cases in order to perform the actual error correction: summing the error pattern to the corrupted symbol, i.e. performing an exclusive-or operation between the bits in the j^{th} symbol and the error pattern e_j.

The block scheme for syndrome evaluation and error evaluation blocks is displayed in Figure 71.

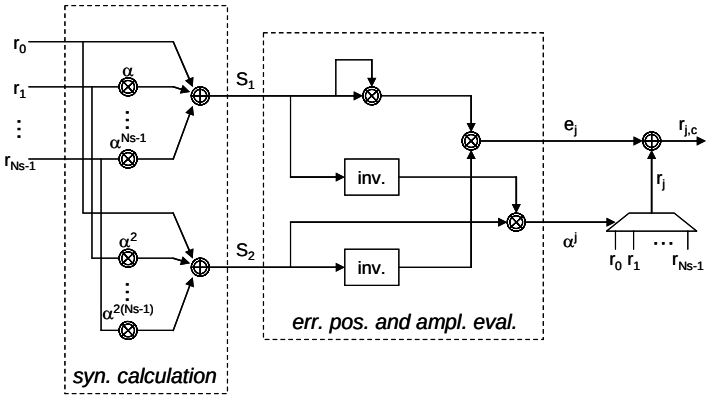

Figure 71: Syndromes and error evaluation block are followed by an exclusive-or operation that corrects the symbol in the j^{th} position. The outcome is the corrected symbol $r_{j,c}$.

6.4.3.3 Descrambler

Descrambling is introduced together with scrambling in section 5.1. As the exclusive-or operation makes the sum operation equivalent to subtraction, the descrambler characteristic equation can be derived to be, respectively for the first and second option:

$D_i = S_i + S_{i-63} + S_{i-62}$ and $D_i = S_i + S_{i-60} + S_{i-59}$.

In Figure 72 the 64-bit parallel 63-bit order descrambler (for the first option, on the left) and 60-bit parallel 60-bit order descrambler (for the second option, on the right) are schematized.

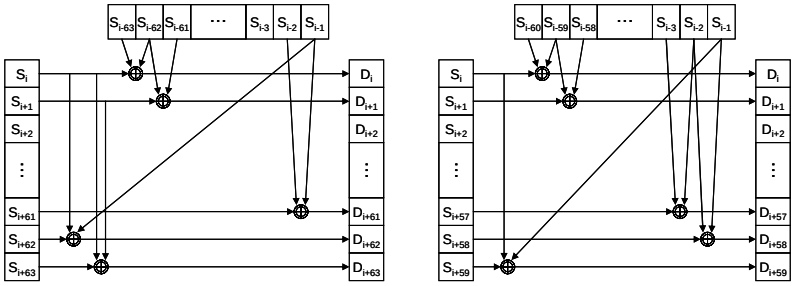

Figure 72: Order-63 64-bit parallel descrambler (left) and order-60 60-bit parallel descrambler (rigth).

For the two descrambler implementation $2\,K_b$ memory elements (1-bit flip-flops) are required together with K_b 3-input exclusive-ors. No reset functionality is needed in the descramblers due to the self-synchronizing implementation.

6.4.3.4 Control logic

After the frame has been synchronized, a minimum of five clock cycles are needed for decoding the line code. One clock cycle is needed for the syndrome calculation, three for the error evaluation and correction, one for descrambling. The total is independent of the choice of implementing the first or second interleaving option. If the same divided clock is used as in the encoding implementation (f_{enc}, that is ready for the other implementation and sufficiently fast), then, the whole decoding process can be done within one frame period.

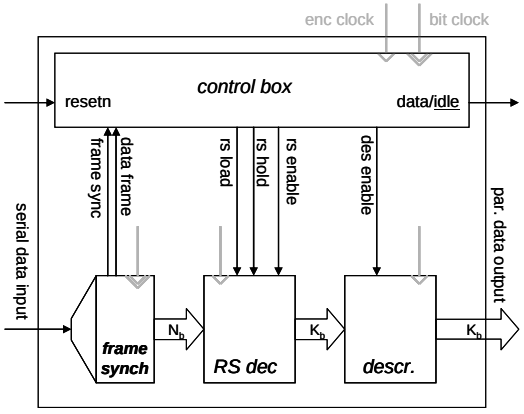

Figure 73: Receiver block diagram with control signals.

The control logic is composed of one main state machine which loops through N_{tot} / 2m states, that is 11 for the first option and 15 for the second one. Once the packet is available as output from the frame synchronizer, the RS decoder load is activated and the RS error hunt and possible correction is performed. After three clock cycles, the corrected and deinterleaved data is available for descrambling, so that the descrambler is enabled. After this last clock cycle, the K_b-bit data packet can be delivered as line decoding is finished. The complete receiver blocks with control signals are displayed in Figure 73.

6.4.4 Implementation complexity comparison

A Verilog model for the two interleaving options schemes was written, simulated and synthesized using a commercial standard cell library. More details on digital design flow and implementation techniques are given in section 7.2, while describing the ASIC implementation. Here, only the results are given as the aim is to compare the implementation costs of the two options.

The comparison of the two implementation complexities is carried out based on synthesis (see 7.2). In Table 16 the number of cells and power consumption of encoder and decoder blocks for the two options is reported. The results take into account only the blocks that perform scrambling, RS encoding/decoding and control logic. It can be concluded that the second option halves the error probability at the price of increased power consumption (due to higher operation frequency, not to cell count which is slightly smaller) and decreased efficiency. The power consumption is estimated for the frequency calculated in 6.4.2.3, that is f_{enc} = 440.88 MHz for the first option and f_{enc} = 601.20 MHz for the second option.

Table 16: RS code complexity for the two line code options. The power consumption is normalized to the value obtained for the transmitter (tx) for the first interleaving option.

Module	Cell count	Power cons.
tx.1st option	1066	5.2 mW
tx. 2nd option	1098	6.4 mW
rx. 1st option	2794	10.6 mW
rx. 2nd option	2402	13.3 mW

The header addition is not comprised in the estimates above. It can be in fact integrated in the serializer and deserializer blocks, which are to be implemented as full custom as to be optimized for speed. As a consequence, a digital implementation estimate is not meaningful.

6.5 Summary

This chapter studies the properties of the line code proposed for the GBT ASIC and the VBD link. The error correction performance is studied first: the code improves the link performance from an uncoded transmission BER α to a coded FER of $3 \cdot 10^3 \cdot \alpha^2$ for the first interleaving option and $1.5 \cdot 10^3 \cdot \alpha^2$ for the second interleaving one. The output data stream DC-wander is expected to be well below 1% of eye opening for a 100 KHz filter cut-off frequency. The average run-length is below 2 bits and the maximum run-length is strictly limited by the presence of the header. Implementation details are presented last. Power consumption is slightly higher for the second option due to the higher operating frequency. It can thus be concluded that the second interleaving option achieves slightly higher error correction capability at the price of reduced redundancy and increased power consumption.

Chapter 7

A demonstrator ASIC for the GBT line code

A test chip for the first option of the encoding scheme was fabricated in a commercial 0.13 μm CMOS technology. This chapter details the ASIC implementation. The functionality of the ASIC is described in section 7.1 along with the main functional blocks characteristics and input/output signals. The technology is described in section 7.2 along with an introduction on design tools and techniques and a short description of the standard cell library that is used. The implementation details are presented in section 7.3 and the test equipment, procedure and result in section 0.

7.1 Functionality

Most of the building blocks are introduced previously in section 6.4, thus in section 7.1.1 the discussion is limited to new blocks and the differences with what introduced previously. Additional functionality aimed to improve the ASIC testability is implemented and discussed in section 7.1.2. Aside from the logical blocks, all the signals required for the ASIC operation are listed in section 7.1.3. The complete block diagrams are discussed in section 7.1.4.

7.1.1 Block diagrams

The ASIC implements the encoding and decoding algorithms as described for the first line code option ($K_b = 64$, $N_{tot} = 88$). The transmitter performs line coding on a parallel input, then serializes the data out of the ASIC. The receiver on the contrary takes the serial stream, line decodes it and outputs parallel data. The main building blocks are shown in Figure 74,

along with the signal bus widths. Only two data packet types are possible for simplicity: data packet or idle packet.

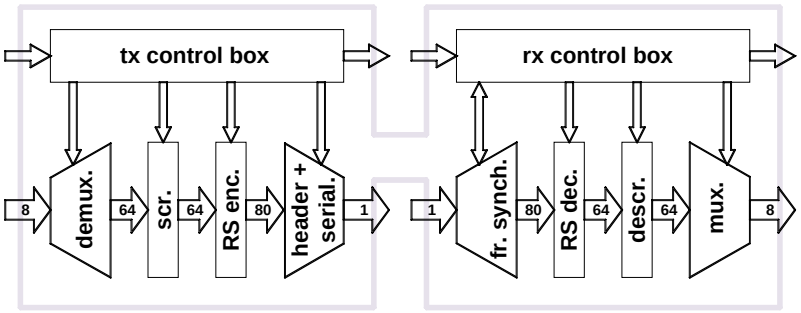

Figure 74: Block diagram of the transmitter (left) and receiver (right) implemented in the ASIC. Details on the control signals are discussed later.

Figure 74 can be compared with Figure 63 and Figure 67. The multiplexer and demultiplexer blocks are the main difference between the two sets of figures: they are necessary due to pad-number limitations, so that the 64 user bits enter the encoder bytewise, and the 64 recevied and decoded bits are multiplexed out of the decoder in 8-bit groups. This limits the number of required pads: from 64 input and 64 output pads to only 8 input and 8 output pads. As the pads occupy a large area (see section 7.2.2 for details), this choice saves area thus making the ASIC cheaper.

The function of the multiplexer is to break the 64-bit user packet into bytes. The function of the demultiplexer is to rebuild the 64-bit packet from user bytes. They are both based on a state machine that scans through eight states, and at each state a byte is read or written to a different location in the 64-bit register.

Scrambler and descrambler are consistent with the parallel implementation that is described in sections 6.4.2.1 and 6.4.3.3.

The addition of the header to the data packet is done in the block that performs serialization. This consists of a shift register that shifts out to the transmitter output one bit per clock period. The first bit of the header is shifted out while loading a new data packet, so that the shifting operation is continuous and at the same time one bit clock is available for loading the shift register.

Only one clock signal at frequency f_{bit} is propagated through the transmitter block and one through the receiver block. A N_{tot}-bit frame is

shifted out the serializer or read in the frame synchronizer every N_{tot} T_{bit} clock periods. Consequently, the control state machines loop through N_{tot} states, instead of (N_{tot} / 2m) states looping at f_{enc} as anticipated in section 6.4 (6.4.2.3 and 6.4.3.4). The timing relations among the control signals anticipated in section 6.4 are maintained in this implementation though, making most of the N_{tot} states unused.

RS blocks and frame synchronization are presented separately in the following sections.

7.1.1.1 RS encoder and decoder

The RS encoder is implemented as two parallel linear feedback shift registers as described in section 6.4.2.2 (refer to Figure 75).

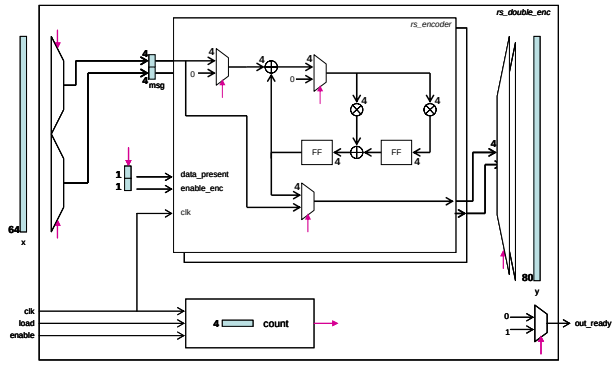

Figure 75: RS encoder implementation.

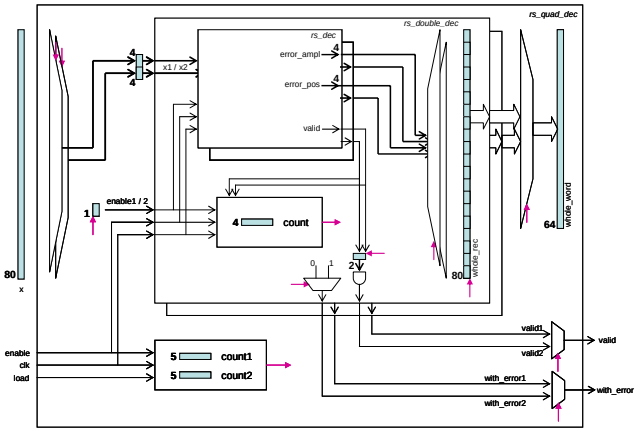

Figure 76: RS decoder implementation.

The RS decoder has the structure shown in Figure 76. Figure 77 displays a detail of the RS decoder implementation, showing that the calculation of the syndromes is done via Horner's rule, as described in section 6.4.3.2.1.

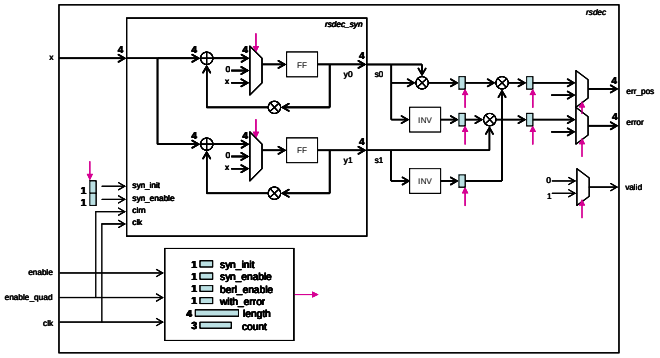

Figure 77: RS decoder implementation: detail on RS error detection and correction.

7.1.1.2 Frame synchronization

The frame synchronization block performs deserialization of the data stream by recovering the framed structure of the data. The state machine governing the mechanisms is shown in Figure 78 for the out-of-lock states and Figure 79 for the in-lock states. The out-of-lock states are characterized by the variable "frame synch" set to zero, while the in-lock states are characterized by the same variable being set to one.

In the first out-of-lock state (OOL-1), the machine is looking for a possible header position. If either a valid data header (dh) or a valid idle header (ih) are found, then the machine moves to the second out-of-lock state (OOL-2), which is characterized by the variable "header found" set to one. At the same time, a "frame index" (fi) value is set: it is incremented at every bit period and keeps track of when a frame shift is complete by scanning through $N_{tot} = 88$ different values. When fi is equal to 7, the state moves to OOL-3 and comparisons are carried out to verify the correctness of the new header. If in the new header position a valid pattern is found, the "correct header count" (chc) counter is incremented; when chc reaches the value X_1, the machine can move to the in-lock state. If the pattern found in the expected header position is not valid, then an "incorrect header count" (ihc) counter is found. When ihc reaches a value of two, then the position in which the machine is looking for the header is considered to be wrong, and

consequently the state moves back to OOL-1 where a new possible position has to be found.

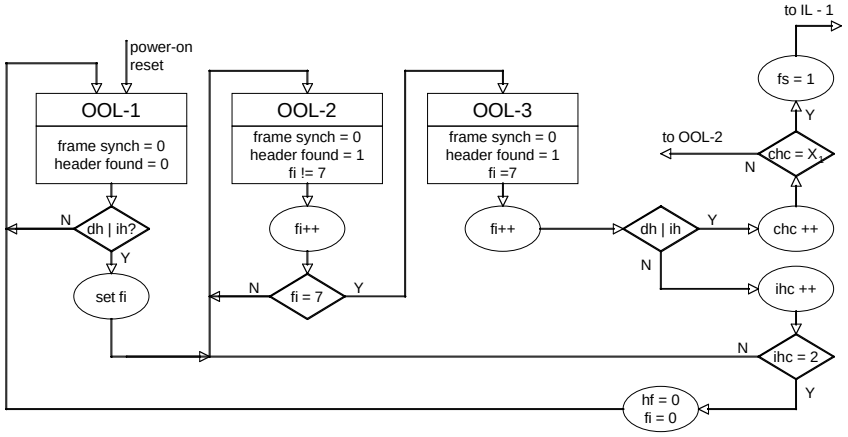

Figure 78: Frame synchronization state machine, out-of-lock states. In the ASIC implementation $X_1 = 10$ is chosen.

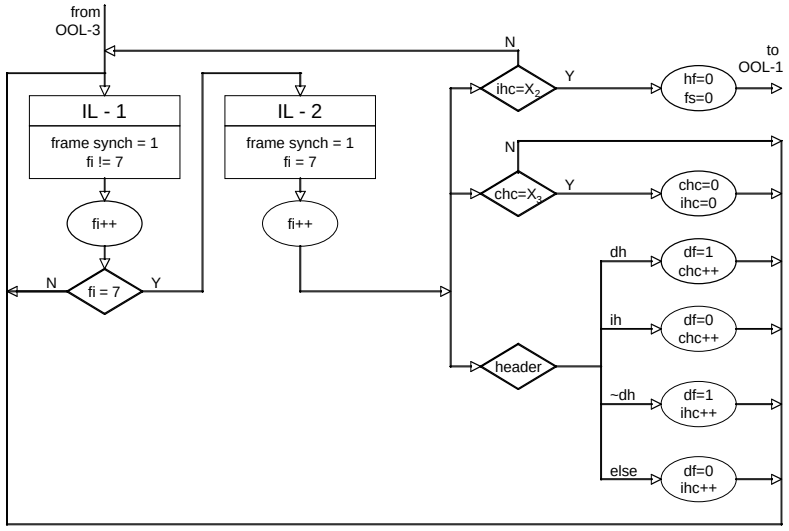

Figure 79: Frame synchronization state machine, in-lock states. In the ASIC implementation $X_2 = 2$ and $X_3 = 63$ are chosen.

The in-lock state shifts the frame index variable at every clock period, and once per frame (state IL-2, when fi = 7) evaluates the correctness of the header pattern. The header pattern is confronted with the data header dh and

the idle header ih, if it corresponds to any of the two, a “correct header counter” (chc) is incremented, if it does not correspond, an “incorrect header counter” (ihc) is incremented. If ihc reaches X_2 counts before chc reaches X_3, then the machine goes back to the OOL-1 state, and searches for a new header position.

The “data frame” df variable informs the decoder control logic if the received data packet is a data or idle packet: if a dh pattern is recognized, or possibly a dh pattern corrupted by a 1-bit SEU error event (~dh), df is set to one. On the contrary, if an ih or if a non-valid header is found, then df is set to zero.

The patterns chosen for the headers and the idle pattern are:

- dh = 01011010 (in binary notation);
- ih = 10100101 (in binary notation);
- idle pattern = 8ba2e8ba2eaaaae8 (in hexadecimal notation).

7.1.2 Testability

The main transmitter and receiver building blocks can be bypassed for testability. Three asynchronous signals are used to decide which blocks are active or not. These signals have to be set at power up, and are not changed during operation. The signal “rs on” controls the utilization of the RS encoder and decoder. The signal “scr on” validates scrambling and descrambling. The data packets follow an alternative path in the case RS or scrambling blocks are inhibited: they are stored in the appropriate registers (“bypass rs” or “bypass scr”).

The signal “fs on” allows skipping the frame locking mechanism to acquire frame lock. An appropriate “synch” signal is used instead for acquiring information about the starting point of a frame. This signal is output by the transmitter and it is an input to the receiver.

Complete block diagrams for the transmitter and receiver are shown in section 7.1.4, Figure 81 and Figure 82, including the alternative data paths allowing bypassing RS or scrambling blocks.

7.1.3 Input/output signals

A total of 32 input and output signals manage the operation of the ASIC. A diagram is shown in Figure 80, and a list follows:

- Two separate clock signals time the transmitter and the receiver respectively: clk enc and clk dec;
- Two separate active-low reset signals reset the transmitter and receiver: resetn enc and resetn dec;
- Eight parallel inputs to the transmitter: data enc in<7:0>;
- Eight parallel outputs from the receiver: data dec out<7:0>;
- One serial output for the transmitter: data enc out;
- One serial input for the receiver: data dec in;
- Two strobe signals controlling when to input parallel data and when to read out parallel data: data e strb and data d strb;
- One signal controlling if the transmitter input is a data packet or an idle packet: data enable;
- One signal informing if the receiver output is a data or idle packet: data/idle;
- One signal informing if the decoding algorithm failed (from the extended error correction capability due to zero-padding): error;
- Three signals are the aforementioned asynchronous controls: rs on, scr on, fs on;
- Two synch signals used in case of inhibited frame synchronization: synch enc out and synch dec in.

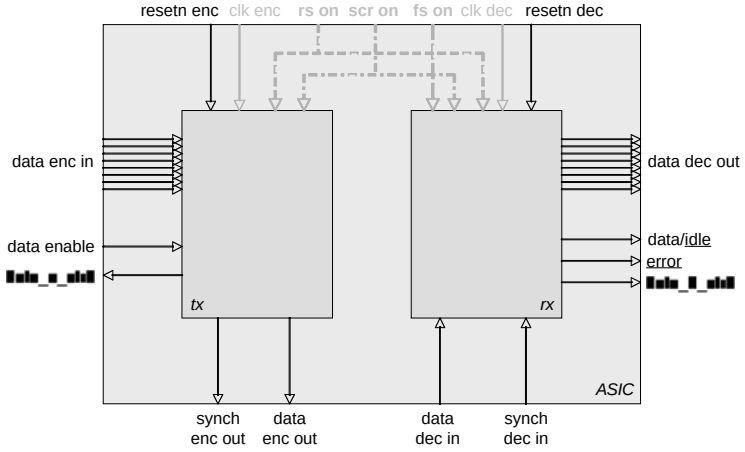

Figure 80: Diagram showing all ASIC input and output signals; in grey, asynchronous and clock signals.

7.1.4 Complete top-level block diagrams

The complete top-level block diagram of the transmitter is reported in Figure 81, and for the receiver in Figure 82.

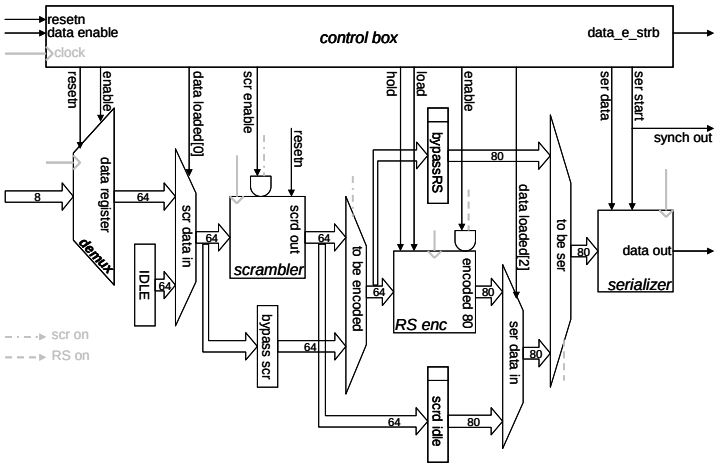

Figure 81: Transmitter full block diagram and control signals.

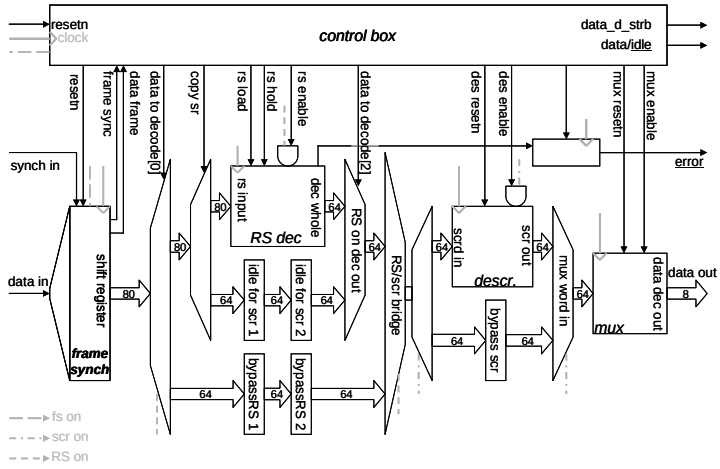

Figure 82: Receiver full block diagram and control signals.

7.2 Digital design aspects

In this section details about the technology and standard cell library used for the ASIC implementation are given. The technology is briefly introduced in 7.2.1. The techniques used for digital designs are introduced

in 7.2.3 along with the software tools and standard design flow. The standard cell library that is used for the implementation is presented in 7.2.2, even though very few features can be presented due to copyright issues.

7.2.1 Technology

The ASIC is implemented in a commercial 0.13 μm CMOS technology offering up to 8 metal layers (among which the last two are thick for lower resistivity). The minimum gate length is 0.12 μm and the minimum width 1.6 μm.

The supply voltages are 1.5 V for the core and 2.5 V for input/output, for compatibility with older technologies. The operating ranges are 1.35 V to 1.65 V for the core DC supply voltage, and -40 ºC to 125 ºC for the junction temperature (nominal 25 ºC).

Core and input/output transistors have different oxide thicknesses: 2.2 nm thick gate oxide for the core and thicker oxide (about 5 nm) for "input/output" transistors. This makes the core transistors more radiation resistant than the input/output transistors [Fac05].

7.2.2 The Artisan Standard Cell Library

A commercial standard cell library was used for the digital design ASIC implementation. The access and utilization of the library is subject to license agreement between the library owner [Arm06] and the user's company. For this reason, most of the information concerning the library is reserved.

The standard cell design style puts logic cells in rows of equal heights. As a consequence, all logic gates in the library have the same height, but may have different widths. Each cell has a power rail at its top and a ground rail at its bottom. The interconnections between gates are done over the cells since current processes allow several metal layers (i.e. 8 metal layers for the process in use). As a consequence, the rows may be abutted and flipped so that power and ground rails are shared between successive rows minimizing area occupancy. The choice of using a commercial library has many benefits, as the availability of a broad range of functions and drive sizes, at the same time allowing quick synthesis with lowered development and production risk.

The chosen library is optimized for speed and density. The typical propagation delay of one inverter varies between 10 and 20 ps depending on

the driving strength, for areas between 4.3 and 20.2 μm^2. Only lowest levels (as polysilicon and metal-1) of interconnect are utilized for intracell connections, so that all other metal layers are left for interconnection.

The library is delivered in form of views and models for leading Computer Aided Design (CAD) tools that allow designers to complete all simulation, synthesis and block-level place-and-route work on a design prior to tapeout. Accurate timing information (delays and constraints) for synthesis and place and route, area occupancy, pin connections, metallization blockages and dynamic and power consumption are among the specified characteristics.

Last but not least, the use of the library is free of charge.

7.2.3 Digital design flow

Digital design takes advantage of CAD tools like synthesizers and place and route tools for automating the steps between high-level functional description and final layout required for tape out. The typical design flow is shown in Figure 83, and details concerning the single steps are explained in the following.

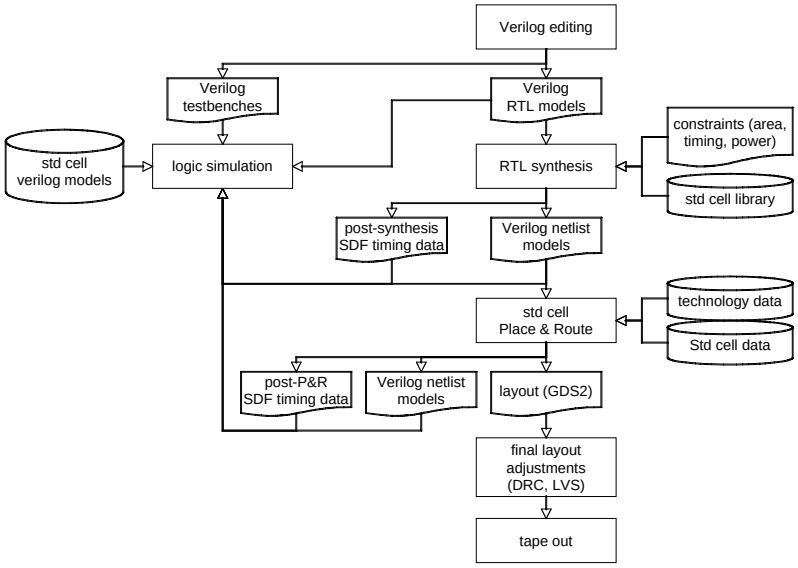

Figure 83: Synthesis with Design Compiler.

The first step is to describe the digital circuit through a Hardware Description Language (HDL) like Verilog [Pal96]. The HDL description can model the circuit at three different levels of detail: physical, functional or behavioral. A physical description (or gate-level netlist) consists of listing every component, net and connection among them: it is a very complete description, but can hardly be read by the designer without the aid of CAD tools. The functional description (or Register-Transfer Level, RTL) models what the circuit does and clarifies how it does it: it is usually clearer than a physical description, and can be translated into one by a tool called a synthesizer. A behavioral description models the circuit without specifying clearly its internal functionality: this description can be used for simulation, but cannot be translated into the other two.

The functional description is the one usually written by the designer for modeling the circuit. It is validated through simulation by means of a logic simulator tool and via a number of test benches also written in Verilog. When a functional description of the circuit is available, it can be fed to a synthesizer, a tool which can infer a possible gate-level realization of the input RTL description that meets user-defined constraints such as area, timing or power consumption. The target logic gates belong to a so-called "standard cell library" which typically includes hundreds of combinational and sequential logic gates. Each logic function is implemented in several gates to accommodate several fanout capabilities or drive strengths. The gate library is described in a tool-specific formats that define, for each gate, its function, its area, its timing and power characteristics and its environmental constraints.

Synthesis generates a gate-level Verilog netlist and a Standard Delay Format (SDF) description. The netlist is used for post-synthesis simulation and as input to the place & route tool. The SDF description includes delay information for simulation. Note that considered delays are at this step correct for the gates but only estimated for the interconnections.

The test benches used for RTL model validation can be reused for verifying the gate-level netlist. The gate-level simulation makes use of Verilog models for the logic gates that are provided in the cell library.

The place and route (P&R) step infers a geometric realization of the gate-level netlist so-called a layout by placing the logic cells listed in the physical netlist and routing the connections among them. The first step of P&R is the floorplanning, where the physical area available for the circuit is defined,

along with the positions of the pads and the main power and ground lines. The P&R step generates a geometric description (layout) in GDS2 format, a SDF description and a Verilog gate-level netlist. The SDF description now includes interconnect delay. The Verilog netlist may be different from the one read as input as the P&R step may make further timing optimizations during placement, clock tree generation and routing (e.g. buffer insertion). The post P&R gate-level netlist can be simulated by using the Verilog test benches and the more accurate SDF data extracted from the layout.

At each level, the designed system is simulated to verify the correctness of functionality and performance before proceeding to the next level. Static timing analysis is for example another important check to be performed as it verifies setup and hold time compliance of the clock signal reaching memory elements in the design. Prior to tape-out, Design Rule Check (DRC) and Layout Versus Schematic (LVS) are also performed.

The tools that were used for this ASIC design flow are: Verilog XL for logic simulation, Synopsys Design Compiler for synthesis, Cadence Silicon Ensemble for P&R, CTGen for clock tree generation, Pearl for static timing analyses.

7.3 Implementation details

A Verilog model for the ASIC as described in section 7.1 was written and simulated extensively for verifying the functionality. The Verilog model was successively synthesized using the commercial standard cell library presented in 7.2.2. Some results concerning synthesis are presented in section 7.3.1, followed by P&R results in section 7.3.2. Finally considerations on input/output pads are presented in section 7.3.3 along with the full ASIC layout image.

7.3.1 Synthesis

Synthesis was performed only on core modules. Transmitter and receiver were treated separately, and only one level of hierarchy was kept for each one of them for leaving the maximum freedom to the synthesizer. Some post-synthesis results are reported in Table 17: the number of cells per block and the estimated power consumption. The power consumption estimation takes into account cell internal power, cell leakage power and net switching power (through back-annotated capacitance load or by wireload model)

Table 17: Number of cells for encoder and decoder, and estimated power consumption.

Synopsys reports	Transmitter	Receiver
Tot Number of cells	1741	5131
Hierarchical	13	46
Leaf	1728	5085
Sequential	612	1369
Combinational	1116	3716
Total Dynamic Power	22 mW/GHz	48 mW/GHz
Cell Leakage Power	19.5859 uW	48.7853 uW

7.3.2 Floorplanning

The available core area is (770 x 490) μm^2, after that the area reserved for the input/output pads (see section 7.3.3) is subtracted from the total area (1.3 x 1 mm^2). The area reserved through floorplanning for the transmitter block is (250 x 450) μm^2, out of which (150 x 350) μm^2 is the active area remaining after having left space for the power rings. Given the rectangular shape of the active area, the standard cell rows are oriented in parallel with the longest dimension of the area. Concerning the receiver, the area reserved with floorplanning is (450 x 450) μm^2, out of which the active area is (350 x 350) μm^2. The ratio between active areas and total number of cells is consistent with (slightly lower than) the rule-of-thumb that predicts about 50000 gates per mm^2 in a 0.13 μm technology.

One single-phase clock is used through the transmitter, and one single phase clock is used through the receiver. Two levels of clock tree are implemented for the transmitter, and 9 components are used to cover 612 leaf pins. Concerning the receiver, four levels consisting of 39 components are used for 1369 leaf pins.

7.3.3 I/O pads and ASIC layout

The total number of pads required for the ASIC is 36. One pad is required for each of the signals listed in section 7.1.3, adding up to a total of 32 pads. Additionally, power and ground pads are required. In order to avoid noise injection into core power/ground lines, two different power supplies are used for the chip core and pads. One pad feeds the chip core and parts of input and pre-driver sections of all I/O cells. The other pad supplies power to all devices that are connected to the I/O power supply. In total, two power pads are used. With the addition of two ground pads, a total of 36 pads is reached.

These considerations make the ASIC a “pad limited” implementation, as opposed to “core limited” implementations. In fact, as a pad is about 250 μm x 70 μm, the required perimeter is 4.52 mm (36 times the pad short dimension, summed to the required corners, 250 x 8), which corresponds to a 1 mm x 1.26 mm die. On the contrary, it is explained in section 7.3.2 that the core area is not extremely dense. These considerations can be verified in the sketch in Figure 84.

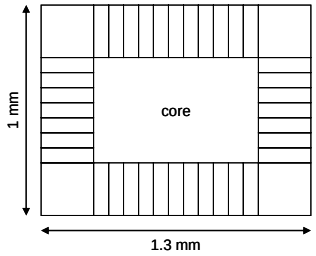

Figure 84: Sketch of die area occupation: 36 is the maximum number of peripheral pads that can be fit in a 1 mm x 1.3 mm area.

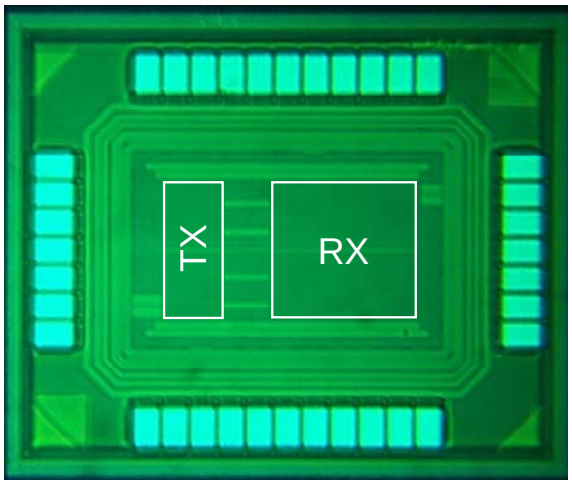

Figure 85: Full ASIC layout.

The standard cell library offers different possible driving strengths and different slew-rates for the output pads. A lower driving strength and slower output slew-rate guarantee lower noise. The pads are chosen to be output buffers with 16 mA direct output and slowest slew-rate.

An image of the full die is shown in Figure 85. The 36 pad openings are clearly visible, together with the pad power ring and the transmitter and receiver power rings. The transmitter and receiver active areas are pointed out with a white outline.

7.4 Test results

In this section the test procedure is first presented in section 7.4.1, the digital tester is introduced in 7.4.1.1 and finally the ASIC tests are presented and the results discussed in section 7.4.2.

7.4.1 Test procedure

Testing a digital ASIC consists of applying test vectors to the IC inputs and then confronting the outputs with the values obtained from simulation in order to verify the ASIC correct operation. In order to do this, a digital tester is used to apply the test vectors to the inputs, and read the outputs (see Figure 86).

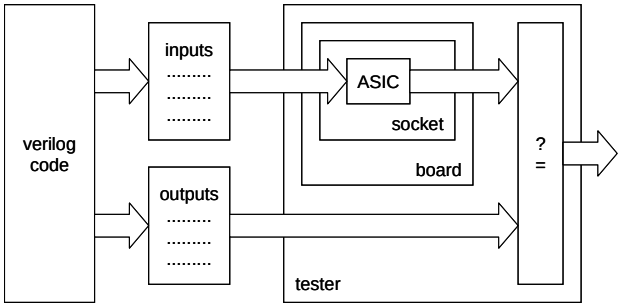

Figure 86: Digital testing basic block scheme. Simulation outputs are checked against die outputs as stimulated by inputs generated by simulation. The digital tester verifies the correspondence amongst the two output sets.

The ASIC is interfaced to the tester by being packaged and subsequently connected to an electronic board via a socket interface (refer to Figure 86). The socket allows good electrical connection between the package and the board based on pressure instead of soldering: this allows testing easily different die samples with the same board. The board used is standard and performs the connection to the digital tester itself.

The operation of the digital tester is presented in section 7.4.1.1. The ASIC tests and the relative results are presented in section 7.4.2.

7.4.1.1 The digital tester

The Mic Group is equipped with a digital ATS tester. The tester hardware provides all required signals to the ASIC: provides the vector inputs, reads the die outputs, provides supply voltages and clocks.

The tester hardware is provided also with a software package, IMS-link, whose task is to convert test vectors from the logic simulator into the format required by the tester logic master. Many parameters have to be specified: the tester configuration information, pin and channel mapping, information about voltage thresholds, drive levels, data formats.

The data conversion from a simulator output file to an IMS pattern file is done by the Automatic Pattern Translator (APT).

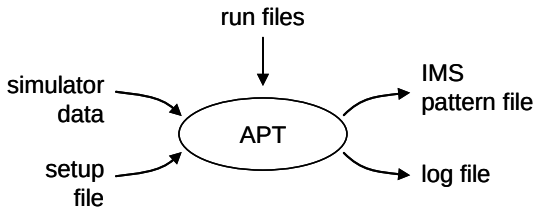

Figure 87: Schematic representation of input and output files required for the generation of the tester pattern files.

In Figure 87 the files required for the generation of the test pattern files is reported. The APT is provided with the simulator data, for example a Value Change Dump (*.vcd) file from a Verilog simulator. It is also provided with the IMS setup file, a file describing the hardware and software configuration of the test station, i.e. data formats and operating conditions. The APT run file contains a variety of directives for the APT, i.e. how to map simulator signal names to IMS channel names. The log file contains summary information about the conversion process, as for example the number of vectors converted. The IMS pattern file is the aim of the conversion operation.

To be noted that the tester uses an internal memory for storing the converted vectors, and this puts a limit to the number of test vectors that can be used in each test. Tests of indefinite length can be put in place through setting a “loop” mode.

7.4.2 Test results

Two types of tests were carried out on six ASICs. Electrical tests are aimed to verifying correct operation of the ASIC with respect to different electrical parameters. Functional tests are aimed to verifying the correct operation of the different blocks and different data paths. Electrical tests are presented in section 7.4.2.1, while functional tests are presented in section 7.4.2.2.

7.4.2.1 Electrical tests

A short discussion on electrical parameters test follows. The minimum supply voltage for which operation is correct is studied, along with the input and output signal margins. The power consumption simulated value is verified and the leakage current measured.

The first measurement tries to address the question of which is the minimum supply voltage which allows correct operation of the ASIC. The pad library is such that two different power supplies can be used to feed the pads and the core. The core is fed at 1.5 V, while the pads can be fed at a higher supply (2.5 V) for compatibility with older technology. Concerning the tests described in this thesis, only the lower supply voltage was used to feed both core and pads for in this case the test board would be cheaper. Thus, the pads are fed at a lower supply than from design, 1.5 V instead of 2.5 V, resulting in slower behaviour. Given these premises, correct operation at 10 MHz was obtained for power supplies as low as 1.25 V.

At a nominal power supply of 1.5 V and at a frequency of operation of 10 MHz, the margins on input and output signals were also tested. Concerning the input signals, the low drive can be as high as more than 500 mV, while the high drive can be as low as 1.25 V. Concerning the output signals, given a crossover value of 750 mV, the low threshold can be as high as 700 mV, while the high threshold can be as low as 800 mV, when the input low and high drive are nominal.

The power measurement is in agreement with the simulated values. As the simulated value is obtained through synthesis, it concerns only the core. On the contrary the pads power consumption is not comprised. This can though be estimated by the standard cell datasheet information, where for each pad the power consumption in μW/MHz is given. Thus, the agreement is between the value measured by the tester (as current drain), and the

simulated core consumption summed to the pad consumption estimated from the datasheet information at the operating frequency.

A leakage test was performed by measuring the leakage current of the device when all inputs are held constant and all outputs are disconnected. The leakage results of the order of 300 pA when measured for either the transmitter or the receiver.

7.4.2.2 Functional tests

These tests are aimed to verifying correct operation of the ASIC as a set of different building blocks and different data paths. The transmitter and the receiver can be tested in stand-alone mode for simplicity, and also in back-to-back mode for completeness. This second test mode consists of connecting the transmitter serial output to the receiver serial input.

The tests were carried out by exploiting fully the tester memory, which can store up to 65k test vectors. On the contrary, the loop test mode is not appropriate for testing the line code operation due to the presence of the scrambler.

Test input vectors were applied to the encoder and decoder for verifying data or idle transmission while applying different control settings. The asynchronous control inputs (“rs on”, “scr on”, “fs on”) were used to verity the correctness of the different data paths and the correct behaviour of the different building blocks. While testing the receiver in stand-alone mode, also data corrupted by errors were used to verify the operation of the error correcting circuitry.

All tests were successful.

The maximum frequency at which the tests were carried out is ~60 MHz due to limitations imposed by the test board.

7.5 Summary

This chapter describs the test chip implementation of the first option of the encoding scheme. The functionality of the ASIC is described along with the main functional blocks characteristics and input/output signals. The chosen commercial 0.13 μm CMOS technology is described along with an introduction on design tools and techniques and a short description of the standard cell library that is used. Implementation details, the test equipment, test procedure and test result are also presented.

Chapter 8

Conclusions and future developments

The use of a combination of line coding and error correction techniques is common in many digital communication systems. The utilization of optical links in high energy physics, though, poses original problems in this subject that need to be addressed with specially tailored solutions.

Line coding is used to match the data stream to the channel that supports the transmission. In the case of serial NRZ transmission, the extraction of the clock signal from the data stream requires the data to be rich in level transitions. Optical links often comprise amplification stages between the photodiode and the clock and data recovery circuitry which perform a low-frequency cutoff on the incoming data, and for this reason the data should be free of any DC component. Many communication systems operate a parallel to serial conversion in the transmitter posing the problem of reconstructing the parallel structure of the user data at the receiver.

While the previous properties are examples of general requirements on digital communication systems, the use of such systems in high-energy physics experiments requires additional features, as these are harsh radiation environments. The system components that sit inside the experiments need to be radiation resistant so that correct functionality can be guaranteed over the lifetime of the experiments. While most types of radiation effects on both electronic and optical components can be overcome, SEUs on the photodiode remain an issue as the photodiode responds to radiation as to light, causing transmission errors which deteriorate the system bit error rate.

The main contribution of this thesis is the design of a line coding scheme that performs error correction targeted to handling SEUs on the photodiode, under the additional requirement that a short time is available for performing the coding and the decoding functions. This constraint is imposed by the low latency requirement of the Versatile BiDirectional link (VBD) link for which the line code is designed.

The proposed coding scheme, which targets error-free transmission for SEU hit rates of about one error per second per link, is performed through data scrambling, followed by Reed-Solomon error correction and the use of a redundant header for frame synchronization.

In the encoding procedure, scrambling addresses the issues of DC-balance of the data stream and abundance of number of transitions, with the additional advantage of requiring no extra bandwidth.

RS error correction is chosen for its high efficiency and capability to handle burst errors. To minimize link latency, the code is chosen to correct only one error per block, and blocks are then interleaved to obtain extended error correction capability. The error correcting scheme maintains the pseudorandom properties acquired by data after scrambling.

A redundant header is added to each frame for frame synchronization purposes. Additionally, different header patterns allow distinction among different data packet types. The header patterns are chosen to be DC-balanced and rich in number of transitions as they are not scrambled.

The code is proposed to have two interleaving options that obtain different error correction capabilities at the price of different code efficiencies. One option is based on interleaving two RS blocks and obtains a 73% efficiency, while the second option interleaves four blocks, for a 67% code efficiency. System performance is improved from an uncoded BER α to $10^3 \cdot \alpha^2$ for both options. Both options achieve a low average run-length, about 2 bits, and a sufficiently DC-balanced data stream (imbalance well below 1% of peak-to-peak voltage for a 100 MHz cutoff frequency of the high-pass filter).

The estimated implementation cost for a 0.13 μm CMOS technology is approximately 1100 cells for the transmitter and 2800 for the receiver, with estimated power consumptions that are respectively 5 mW and 12 mW. The first code option was implemented in a 0.13 μm CMOS ASIC through a

standard digital design flow. The ASIC was produced and successfully tested.

What differentiates the proposed line code from commercially adopted schemes is its robustness against single event upsets. To achieve the same level of robustness by combining commercially adopted line codes with error correction methods, the results would be increased latency and reduced efficiency. In the VBD case, the aim is to provide the user with a reliable link, comprising error correction capability at the "physical layer". On the contrary, commercially adopted schemes are often part of more complex systems, so that they can rely on higher layers for performing error correction.

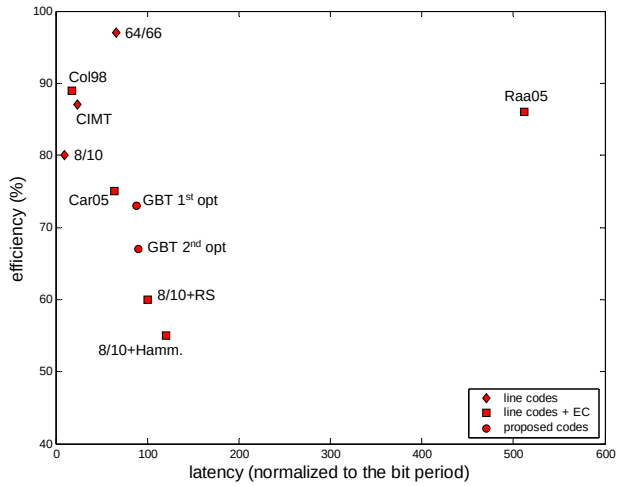

Figure 88: Comparison of code efficiencies as function of packet length. The error correction capability is not taken into account.

A comparison of the proposed code with alternative implementations is shown in Figure 88 where the code efficiency is plotted against the code latency (in number of bit periods). Line codes introduced in the second chapter of this thesis (section 2.1.3) have very high efficiency, but no error correction capability. When they are combined with error correcting schemes, as done in the third chapter of this thesis (section 3.4), then the efficiency decreases drastically. Only the performance of the code presented in [Car05] is comparable to the code proposed for the VDB link. It uses a two-error correcting BCH code, and conditionally inverts the data for DC-

balance, forcing at least a transition every 63 bits; it has a 51-bit input word (for a 75% efficiency). The code proposed in this thesis, though, is preferable for the VBD link as it is richer in number of transitions due to the scrambling operation, which allows a lower jitter reconstruction of the clock signal. The code presented in [Raa05] instead has the disadvantage of using interleaving extensively, requiring a latency that exceeds the VBD requirement.

Regarding future developments of the project, there remains much to be done concerning GBT blocks designs and VDB link component choices, as the project is still in a very early stage. For what concerns the completion of the work carried out for this thesis, it is extremely important to experimentally evaluate the error correction requirements of the link. The choice of the photodiode and its radiation tests, along with an estimation of the optical power budget, are critical for understanding the amount and characteristics of SEU errors, which in this thesis are only assumed to be likely and estimated in quantity by analogy with other links.

In this perspective, the error correction scheme needs to be conservative. Errors that are not RS corrected, in fact, are delivered to the descrambler and are consequently multiplied, so that this situation needs to be avoided by choosing an error correction scheme that is suitable for the expected SEU rate.

For example, it would be useful to perform studies on a two-error correcting RS scheme, which would guarantee an additional 6-7 orders of magnitude improvement in the system BER. The RS scheme proposed in this thesis is chosen for its very low latency, low power and ease of implementation, but in case latency is not as critical in the SLHC upgrade of the detector electronics, then a more complete error correcting scheme can be employed.

It should also be noted that the intrinsic value of the coding scheme holds. Different RS schemes can be chosen to achieve different error correction capabilities. The scrambler order can be changed in order to match different data word lengths. Header length can be chosen to match different RS code choices. Consequently, the code structure, i.e. concatenation of scrambling, RS coding and header addition, can be maintained and adapted to different link needs.

Appendix
GF additions and multiplications

Table 18: Modulo-2 addition over GF(2^3).

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

Table 19: Modulo-2 multiplication over GF(2^3).

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

Table 20: Modulo-2 addition over GF(2^4).

+	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
2	2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
3	3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
4	4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
5	5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
6	6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
7	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
8	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
10	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
11	11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
12	12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
13	13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
14	14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 21: Modulo-2 multiplication over GF(2^4).

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	0	2	4	6	8	10	12	14	3	1	7	5	11	9	15	13
3	0	3	6	5	12	15	10	9	11	8	13	14	7	4	1	2
4	0	4	8	12	3	7	11	15	6	2	14	10	5	1	13	9
5	0	5	10	15	7	2	13	8	14	11	4	1	9	12	3	6
6	0	6	12	10	11	13	7	1	5	3	9	15	14	8	2	4
7	0	7	14	9	15	8	1	6	13	10	3	4	2	5	12	11
8	0	8	3	11	6	14	5	13	12	4	15	7	10	2	9	1
9	0	9	1	8	2	11	3	10	4	13	5	12	6	15	7	14
10	0	10	7	13	14	4	9	3	15	5	8	2	1	11	6	12
11	0	11	5	14	10	1	15	4	7	12	2	9	13	6	8	3
12	0	12	11	7	5	9	14	2	10	6	1	13	15	3	4	8
13	0	13	9	4	1	12	8	5	2	15	11	6	3	14	10	7
14	0	14	15	1	13	3	2	12	9	7	6	8	4	10	11	5
15	0	15	13	2	9	6	4	11	1	14	12	3	8	7	5	10

Bibliography

[Agr05] G. P. Agrawal, "Lightwave technology", Wiley-Interscience, Hoboken, New Jersey, 2005.

[Agr02] G. P. Agrawal, "Fiber-optic communication systems", 3rd ed., Wiley-Interscience, New York, 2002.

[Ane99] G. Anelli, M. Campbell, M. Delmastro et al., "Radiation-tolerant VLSI circuits in standard deep submicon CMOS technologies for the LHC experiments: practical design aspects", IEEE Transactions on Nuclear Science, vol.46, no.6, December 1999, pp.1690-1696.

[Ane00] G. Anelli, "Design and characterization of radiation tolerant integrated circuits in deep submicron CMOS technologies for the LHC experiments", PhD Thesis.

[Ane06] G. Anelli, "Semiconductor technology for integrated circuit front ends", Notes of the Short Course of the IEEE Nuclear Science Symposium & Medical Imaging Conference, S. Diego, 29th October 2006.

[Arm06] Information available online at: http://www.arm.com/

[Bar06] S. Baron, “Status of the TTC upgrade”, Proceedings of 12nd Workshop on Electronics for LHC Experiments, Valencia, 25-29 September 2006.

[Ber68] E. Berlekamp, “Algebraic coding theory”, McGraw-Hill, New York, 1968.

[Ber82] E. R. Berlekamp, “Bit serial Reed-Solomon encoders”, IEEE Transactions on Information Theory, vol.IT-28, 1982, pp.869-874.

[Bla79] R. E. Blahut, “Transform techniques for error-control codes”, Proc. Conf. Information Sciences and Systems, Princeton, N.J., 1973, pp.495-97.

[Bor95] P. Borwein, T. Erdélyi, “Polynomials and Polynomial Inequalities”, Springer-Verlag, New York, 1995, p. 8. ["Horner's Rule"]

[Bos60] R. C. Bose, D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes”, Information Control, 3, March 1960, pp.68-79.

[Bri93] J. Bristow, J. Lehman, “Component tradeoffs and technology breakpoints for a 50 Mbps to 3.2 Gbps fiber optic data bus for space applications”, Proc. SPIE, vol. 1953, 1993, pp.159-169.

[Bro83] R. M. Brooks, A. Jessop, “Line coding for optical fibre systems”, Int. J. Electronics, vol.55, no.1, 1983, pp. 81-120.

[Byl80] P. Bylanski, D. G. W. Ingram, “Digital transmission systems”, Revised 2nd ed., Peter Peregrinus Ltd., 1980.

[Car86] A. B. Carlson, “Communication systems”, 3rd ed., McGraw-Hill International Editions, Singapore, 1986.

[Car05] D. T. Carney, E. W. Chandler, “Improving bit-error-rate performance in serial digital multi-gigabit communication

systems with error correction coding", News & Views, winter 2005 issue, pp. 22-24.

[Cer01] G. Cervelli, A. Marchioro, P. Moreira, F. Vasey, "A radiation tolerant laser driver array for optical transmission in the LHC experiments", Proc. of the 7th Workshop on Electronics for LHC Experiments, Stockholm, October 2001, pp 155-159.

[Chi64] R. T. Chien, "Cyclic decoding procedure for the BCH codes", IEEE Transactions on Information Theory, IT-10, October 1964, pp.357-63.

[Chr96] J. Christiansen, A. Marchioro, P. Moreira, "TTCrx, an ASIC for Timing, Trigger and Control Distribution in LHC Experiments", Proceedings of 2nd Workshop on Electronics for LHC Experiments, Balatonfüred, 23-27 September 1996.

[Chr03] J. Christiansen, A. Marchioro, P. Moreira, T. Toifl, "A Timing, Trigger and Control Receiver ASIC for LHC Detectors", CERN-EP/MIC, Geneva Switzerland, Version 3.8, Jan. 2003.

[CMS98] AA. VV., "CMS Tracker project tech. design report", CERN LHCC 98-6, 1998.

[Col98] A. Coles, D. Cunningham, "Low overhead block coding for Multi-Gb/s Links", HP Labs Technical Reports, Extended Enterprise Laboratory HPL-98-168, October 1998.

[DeR06] A. De Roeck, "The LHC Upgrade", in CERN Summer Student Lectures 2006, available online at: http://agenda.cern.ch/fullAgenda.php?ida=a062750.

[Duz03] S. Duzellier, "Radiation effects analysis: Single Event Effects", Notes of the Short Course of the 7th European Conference on Radiation and its Effects on Components and Systems, Huis Ter Duin, (The Netherlands), 15-19 September 2003, Section 3-C.

[Fac01] F. Faccio, G. Berger, K. Gill, M. Huhtinen, A. Marchioro, P. Moreira, F. Vasey, "Single event upset tests of an 80-Mb/s Optical Receiver", IEEE Transactions on Nuclear Science, vol.48, no.5, October 2001, pp.1700-1707.

[Fac05] F. Faccio, G. Cervelli, "Radiation-induced edge effects in deep submicron CMOS transistors", IEEE Transactions on Nuclear Science, vol.52, no.6, December 2005, pp. 2413-2420.

[Fai91] I. J. Fair, W. D. Grover, W. A. Krzymien, R.I. MacDonald, "Guided scrambling: a new line coding technique for high bit rate fiber optic transmission systems", IEEE Transactions on Communications, vol. 39, no.2, February 1991, pp. 289-297.

[For65] G. D. Forney, "On decoding BCH codes", IEEE Transactions on Information Theory, IT-11, October 1965, pp.549-57.

[Gil97] K. Gill, V. Arbet-Engels, J. Batten, G. Cervelli, R. Grabit, C. Mommaert, G. Stefanini, J. Troska, F. Vasey, "Radiation damage studies of optoelectronic components for the CMS tracker optical links", Proc. of RADECS, Cannes, 1997, pp. 276-281.

[Gil98] K. Gill, C. Aguilar, V. Arbet-Engels, C. Azevedo, J. Batten, G. Cervelli, R. Grabit, F. Jensen, C. Mommaert, J. Troska, F. Vasey, "Comparative Study of Radiation Hardness of Optoelectronic Components for the CMS Tracker Optical Links", Proc. of RADECS, Oxford, Sept. 1998 and the fourth workshop on electronics for LHC experiments, Rome, Sept. 1998, pp 96-99.

[Hoc59] A. Hocquenghem, "Codes correcteurs d'erreurs", Chiffres, 2, 1959, pp.147-56.

[Iee00] R. Walker, R. Dugan, "64b/66b low-overhead coding proposal for serial links", presentation to the IEEE 802.3ah (10GE) Task Force, January 2000. URL:

http://grouper.ieee.org/groups/802/3/10G_study/public/jan00/walker_1_0100.pdf .

[Imm81] K. Immink, "Digital Audio Discs with Optical read-out," Proc. IEEE ICASSP, June 1981, pp. 587-9.

[Kal04] A. Kalavagunta, R. Schrimpf, M. Neifeld, "Design considerations for optical systems in ionizing and nonionizing radiation environments", IEEE Transactions on Nuclear Science, vol. 51, no.6, December 2004, pp. 3595-3602.

[Jen99] F. Jensen, C. Azevedo, L. Bjorkman, G. Cervelli, K. Gill, R. Grabit, F. Vasey, "Optical links for HEP experiments", Proc. of the Workshop on the development of future linear electron-positron colliders for particle physics studies and for research using free electon lasers, Lund, 1999.

[Joh96] A. H. Johnston, "The influence of VLSI technology evolution on radiation-induced latchup in space systems", IEEE Transactions on Nuclear Science, Volume 43, Issue 2, Part 1, April 1996 pp.505-521.

[Lee94] E. A. Lee, D. G. Messerschmitt, "Digital communication", 2nd ed., Kluwer Academic Publishers, 1994, pp. 591-603.

[Lee95] B. G. Lee, S. C. Kim, "Low-rate parallel scrambling techniques for today's lightwave transmission", IEEE Communications Magazine, April 1995, pp.84-95.

[Lee00] C. G. Lee, H. H. Lee, D. Y. Kim, J. W. Kim, H. W. Jung, "A new line code for 10-Gigabit Ethernet: MB810", IEEE International Conference on Communications, 2000, vol.3, 18-22 June 2000, pp.1774-7.

[Lin04] S. Lin, D. J. Costello, "Error Control Coding", 2nd ed., Prentice Hall, Upper Saddle River, NJ, 2004, ch.2-7.

[Mar94] P. W. Marshall, C. J. Dale, M. A. Carts, K. A. LaBel, "Particle-induced bit error in high performance fiber optic data links for satellite data management", IEEE Transactions on Nuclear Science, vol.41, no.6, December 1994, pp.1958-1965.

[Mar01] C. J. Marshall, P. W. Marshall, M. A. Carts, R. Reed, S. Baier, K. LaBel, "Characterization of transient error cross sections in high speed commercial fiber optic data links", IEEE Radiation Effects Data Workshop, 2001, pp. 142-145.

[Mar04] P. W. Marshall, P. T. Wiley, R. N. Prusia, G.D. Rash, H. Kim, K. A. LaBel, "Proton induced bit error studies in a 10Gb/s Fiber Optic Link", IEEE Transactions on Nuclear Science, vol.51, no.5, October 2004, pp.2736-2739.

[Met06] E. Metral, S. Gilardoni, "Introduction to accelerators", in CERN Summer Student Lectures 2006, available online at: http://agenda.cern.ch/fullAgenda.php?ida=a062758.

[Mor01] P. Moreira, G. Cervelli, J. Christiansen, F. Faccio, A. Kluge, A. Marchioro, T. Toifl, "A radiation tolerant gigabit serializer for LHC data transmission", Proc. of the 7th Workshop on Electronics for LHC Experiments, Stockholm, October 2001, pp 150-154.

[Mor03] P. Moreira, A. Marchioro, "QPLL – a quartz crystal based PLL for jitter filtering applications in LHC", Proceedings of the 9th Workshop on Electronics for LHC Experiments October 2003, pp. xxx-xxx.

[Nir96] S. Niranjan, J. F. Frenzel, "A comparison of fault-tolerant state machine architectures for space-borne electronics", IEEE Transactions on Reliability, vol.45, no.1, March 1996, pp.109-113.

[Pal96] S. Palnitkar, "Verilog HDL: a guide to digital design and synthesis", SunSoft Press, New York, 1996.

[Pap91] A. Papoulis, "Probability, random variables, and stochastic processes", 3rd ed., McGraw-Hill International Editions, 1991, pp.43-47.

[Pap03] G. Papotti, J. Troska, K. Gill, R. Grabit, R. Macias, F. Vasey "Active component qualification for the CMS Tracker readout optical links", Proceedings of the 9th Workshop on Electronics for LHC Experiments October 2003, pp. 204-208.

[Raa05] B. Raahemi, "Error correction on 64/66 bit encoded links", Canadian Conference on Electrical and Computer Engineering, 1-4 May 2005, pp.412-416.

[Ree60] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields", SIAM Journal of Applied Mathematics, vol.8, 1960, pp.300-304.

[Rod03] J. C. Rodriguez, "Radiation effects analysis: total dose", Notes of the Short Course of the 7th European Conference on Radiation and its Effects on Components and Systems, Huis Ter Duin, (The Netherlands), 15-19 September 2003, Section 3-A.

[Sac05] E. Säckinger, "Broadband circuits for optical fiber communication", Wiley-Interscience, Hoboken, New Jersey, 2005.

[Sch94] J. R. Schwank, "Basic mechanisms of radiation effects in the natural space environment", Notes of the Short Course of the 1994 IEEE Nuclear and Space Radiation Effects Conference, Tucson, (Arizona), July 1994, Section II, pp.1-109.

[Sph06] P. Sphicas, "Trigger and DAQ systems (at the LHC)", CERN Summer Student Lectures 2006, available online at: http://agenda.cern.ch/fullAgenda.php?ida=a062849.

[Sug75] Y. Sugiyama, M. Kasahara, S. Hirasawa, T. Namekawa, "A method for solving key equation for decoding Goppa codes", Information Control, 27, January 1975, pp.88-99.

[Tay02] B.G.Taylor, "Timing Distribution at the LHC", Proc. 8th Workshop on Electronics for LHC Experiments, Colmar, France, 9-13 September 2002, CERN 2002-003, pp. 63-74.

[Tro98] J. Troska, J. Batten, K. Gill, F. Vasey, "Radiation effects in commercial off-the-shelf single-mode optical fibres", Proc. of the SPIE Volume 3440 Presented at "Photonics for Space Environments VI", San Diego, 22nd July 1998.

[Tro03] J. Troska, G. Cervelli, F. Faccio, K. Gill, R. Grabit, A. M. Sandvik, F. Vasey, A. Zanet, "Optical readout and control systems for the CMS Tracker" Nuclear Science Symposium, Norfolk, VA, November 2002.

[Tru01] T. E. Truman; S. Leilei, K. Azadet, "The simple link protocol: a zero-overhead packet delineation scheme for high-speed Ethernet", Proceedings of Technical Papers of International Symposium on VLSI Technology, Systems, and Applications, 18-20 April 2001, pp.65-68.

[Vas98] F. Vasey, V. Arbet-Engels, J. Batten, G. Cervelli, K. Gill, R. Grabit, C. Mommaert, G. Stefanini, and J. Troska, "Development of radiation-hard optical links for the CMS tracker at CERN", IEEE Trans. Nucl. Sci., vol.45, no. 3, pp. 331-337, Jun. 1998.

[Wal91] R. C. Walker, T. Hornak, C. Yen, J. Doernberg, K. H. Springer, "A 1.5Gb/s link interface chipset for computer data transmission", IEEE Journal on Selected Areas in Telecommunications, vol.9, no.5, June 1991, pp. 698-703.

[Wal92] R. C. Walker, "The design and implementation of a chipset for gigabit/second computer networks", Thesis presented to the faculty of California State University, 1992, pp. 16-19.

[Wic94] S. B. Wicker, V. K. Bhargava, “Reed-Solomon Codes and their applications”, Wiley-IEEE Press, New York, 1994.

[Wid83] A. X. Widmer, P. A. Franaszek, “A DC-balanced, partitioned-block, 8B/10B transmission code,” IBM Journal R&D. 27, 1983, pp. 440-451.

[Yen92] C. Yen, R. C. Walker, P. T. Petruno, C. Stout, B. Lai, W. J. McFarland, “G-Link: a chipset for Gigabit-Rata Data Communication”, Hewlett-Packard Journal, October 92.