P&O EAGLE

Bertold Van den Bergh
Yuri Murillo

September 2017

1 Introduction

Digital video broadcasting presents several advantages compared to the analog scheme, such as higher capacity, lower error rate, better image quality and a more efficient use of electromagnetic spectrum. Currently the most extended standard for direct TV broadcast via satellite is DVB-S2, which allows to transmit HDTV signals over satellite TV channels thanks to its spectral efficiency. When talking about terrestrial broadcasting, the DVB-T2 standard may be used, as it is a superset of DVB-S2. In fact, DVB-T2 is soon expected to become the worldwide preferred Digital Terrestrial Broadcast system.
This standard would be suitable for drone applications such as First Person View and research and rescue missions in case of emergencies, where the drone may need to fly hundreds of meters away from the remote control. The wireless communication submodule of the EAGLE project focuses on implementing a reliable video downlink that streams in real time the output of the camera attached on the drone to a PC.

2 Task

Students will have to complete the design of the DVB-S2 transmitter for the video downlink that runs inside the Zybo FPGA, implementing the parts of the standard that have been purposely left blank for them. Although DVB-T2 would make more sense for the application considered and a standard used for satellite communications may seem overkill for a drone link, the reduced complexity of DVB-S2 fits better within the scope of the project.
Since DVB-S2 standard compliance testing with an off-the-shelf receiver gives binary output (it works correctly when everything is well implemented and it does not work at all when something is wrong in the code), the students are given automated test-benches with official BBC test streams. They consist of raw data of predefined streams at several points inside the transmitter block chain, so if the implemented code behaves as expected the output of the FPGA transmitter should match the one from the test vectors. As an advantage of

designing the transmitter in software using VHDL, students would be able to trace back the waveforms of the different signals inside the transmitter and detect the exact moment in which any possible bug may occur by means of a wave viewer, such as GTKWave.
Development of the remaining VHDL blocks of the DVB-S2 transmitter, as well as integration on the FPGA and correct link management using the provided software tools and drivers is a crucial task within the scope of the EAGLE project, and therefore the mentioned tasks need to be fully functional. As an additional exercise, students may use the provided IP-based system to develop a data-logging system so the receiver can get real-time information on the status of the drone. Below, an overview of the full functional chain is given for completeness, but it is not necessary to understand everything to complete the EAGLE project. Of course, the more understanding, the more appreciation by your expert!
The following subsections give an overview of the aforementioned tasks.

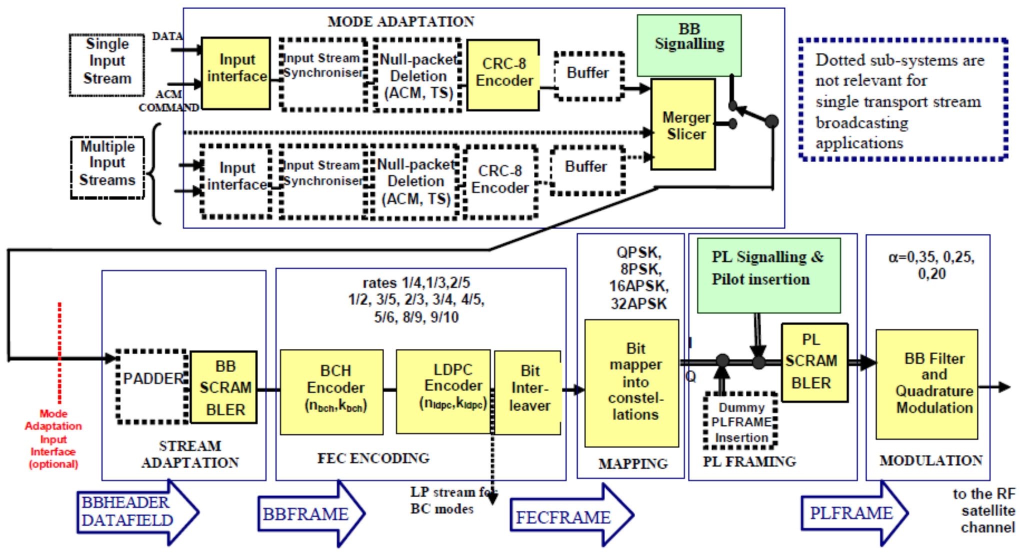
MODE ADAPTATION
Single Input Stream
DATA
ACM COMMAND
Input interface
Input Stream Synchroniser
Null-packet Deletion (ACM, TS)
CRC-8 Encoder
Buffer
BB Signalling
Merger Slicer
Dotted sub-systems are not relevant for single transport stream broadcasting applications
Multiple Input Streams
Mode Adaptation Input Interface (optional)
PADDER
BB SCRAMBLER
STREAM ADAPTATION
rates 1/4,1/3,2/5 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, 9/10
BCH Encoder (n_{bch},k_{bch})
LDPC Encoder (n_{ldpc},k_{ldpc})
Bit Inter-leaver
FEC ENCODING
QPSK, 8PSK, 16APSK, 32APSK
Bit mapper into constel-lations
MAPPING
I
Q
PL Signalling & Pilot insertion
Dummy PLFRAME Insertion
PL SCRAMBLER
PL FRAMING
α=0,35, 0,25, 0,20
BB Filter and Quadrature Modulation
MODULATION
to the RF satellite channel
BBHEADER DATAFIELD
BBFRAME
LP stream for BC modes
FECFRAME
PLFRAME

Figure 1: DVB-S2 block diagram.

2.1 DVB-S2 architecture

The full DVB-S2 block diagram can bee seen in Figure 1. It consists of five main groups:

1. Base-band processing blocks: where the incoming video data is pipelined, CRC-checked and processed to build base-band frames, after which the whole stream is scrambled.

2. Forward Error Correction (FEC) blocks: consisting of BCH and LDPC coding blocks.
3. Constellation mapping.
4. Pilot insertion and frame scrambling.
5. Baseband filtering and modulation.

A full implementation of the DVB-S2 standard has been developed for the Zybo board, with its architecture being represented in the figure below. It receives the compressed video stream from the camera into the CPU. Several controllers and drivers run in these cores in order to ensure real time operability and data delivery of the system. The transmitter VHDL blocks run in the FPGA, as well as the USB data plane in order to reduce latency and save CPU resources. However, the USB control plane is handled by the CPU for higher flexibility. Description of the interesting blocks, as well as software provided (which includes the explanation on the usage of the network interface) will be provided in the next sections.

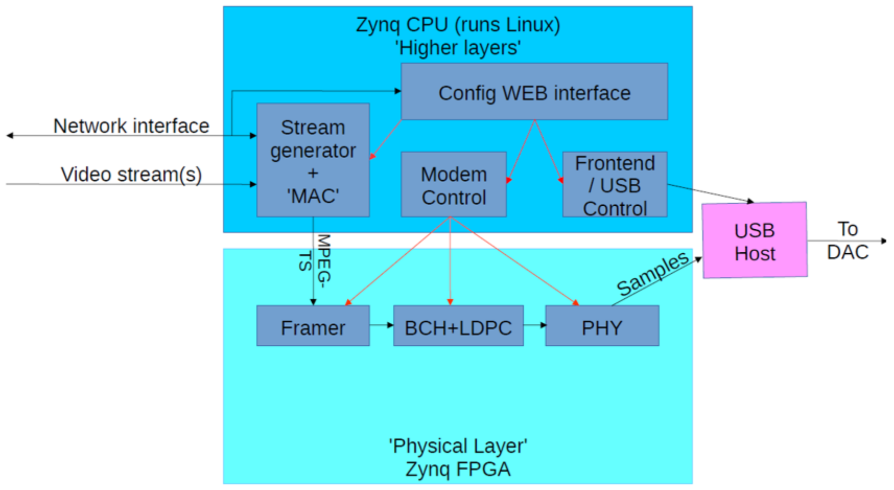

Figure 2: DVB-X2 architecture for the Zybo.

2.2 Overview of baseband processing and BCH coding blocks

The DVB-S2 architecture provided to the students is lacking a main block: the BCH coding. It is expected that they understand what this block does in the whole transmitter chain by reading the standard and design the code that implements its behaviour accordingly. This code needs to integrate with the rest of the platform, so documentation about the correct way of signaling connectivity between VHDL blocks is also provided.

2.2.1 Baseband processing

The block chain of the baseband processing module is represented in Figure 3. It consists of the mode adaptation and stream adaptation submodules.

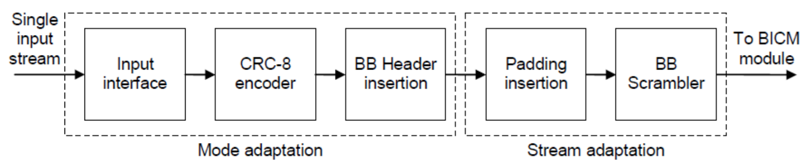

Figure 3: Baseband processing module for single stream.

The mode adaptation submodule ensures that the incoming video stream is divided into a certain number of video packets within the baseband frame, each one of them CRC-8 encoded, after which a baseband header is appended where all the information about configuration and parameter selection is defined. The submodule also needs to ensure that the incoming video stream does not suffer any loss of packets due to timing misbehaviour: the video data packets need to flow continuously in the receiver as they do when they come from the camera. A detailed view of the data stream is depicted below.

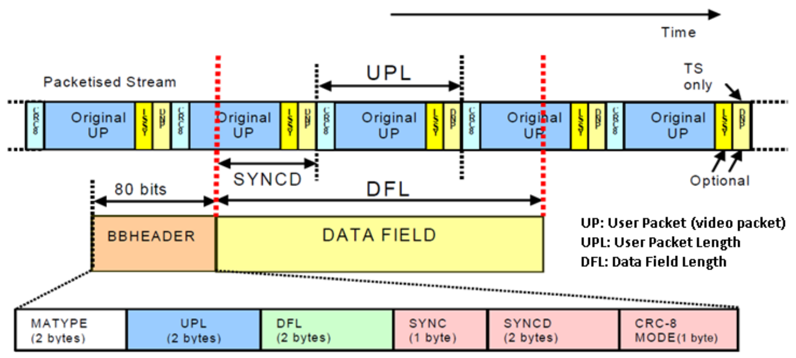

Figure 4: Single stream of baseband frames.

The stream adaptation submodule simply inserts zero padding until completing the length of the baseband frames if an exact number of video packets are chosen to be introduced in such frames. If segmentation of incoming video packets is chosen to ensure a more efficient performance this block is not needed. Finally, the whole data stream is scrambled by means of a feed-back shift register with polynomial $1 + x^{14} + x^{15}$ for generation of the Pseudo Binary Random Sequence (PBRS), as can be seen below. This block is given to the students as part of the DVB-S2 transmitter implementation, so there is no need to develop it but it is expected that they get familiar with the structure of the baseband frame.

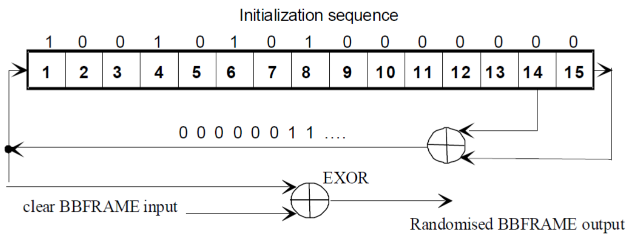

Figure 5: Possible implementation of the scrambler shift register.

2.2.2 BCH encoding

BCH are cyclic error correcting codes that may correct multiple bit errors in the received data stream. Moreover, based on the BCH polynomial chosen one may adjust the number of bit errors that the encoder/decoder will be able to correct. It is expected that the students understand the behaviour of such codes and implement an encoder module for the three BCH polynomials supported by the standard, after which the data stream will be fed into the LDPC encoder for further error correction.

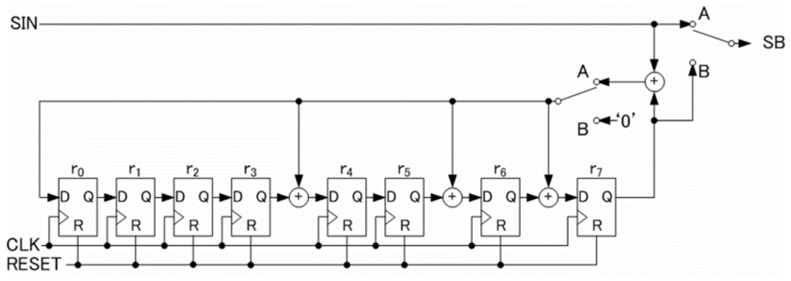

Figure 6: Implementation of a BCH error correcting block.

2.3 Automated standard compliance tests

As stated in the introduction, debugging a software implementation of the transmitter may be a complex task, since when using standard off-the-shelf receivers the system performs properly if everything is well designed, and completely doesn't if anything fails.
In order to speed up the debugging process, automated standard compliance tests are provided to search for bugs in the VHDL code. These testbenches use standard testvectors obtained from British Broadcasting Corporation (BBC).

They provide raw bitsreams of the output of every block in the transmitter block chain, so these outputs should match bit-by-bit when the same input is introduced in the DVB-S2 architecture if everything works as it should. If not, they will automatically point out the submodule causing the failure and in which stage of the whole stream.

2.4 Integration on the Zybo

Two programs running on the CPU have been developed for the Zybo, as well as the hardware for the frontend, which is depicted below.

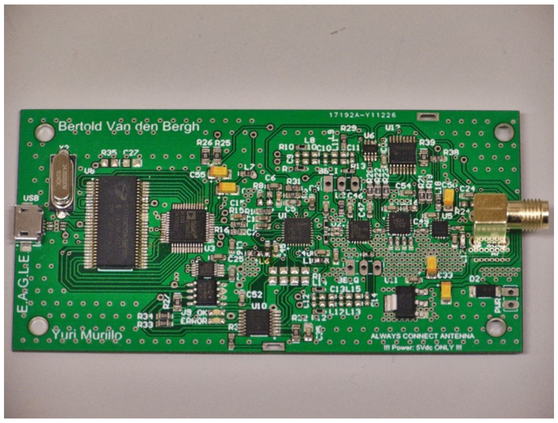

Figure 7: Developed frontend hardware.

The first one is the *EagleTxController*, which manages the frontend and sends samples from FPGA to the USB connection.

The second one, named *ULEFramer*, packs IP or Ethernet packets into ULE frames and delivers them to the FPGA for transmission. ULE is a protocol that enables the system to use the Ethernet port as a virtual network connection for encapsulating the video packets and sending them over IP.

The transmitter settings may be changed via *v4l2* controls, which can be done remotely to design the adaptive system.

2.5 Additional tasks

After successful development, testing and integration of the VHDL code on the drone, students should accomplish an additional task for the wireless video link. As the developed DVB-S2 system provides an IP link between transmitter and receiver, this could be used for data logging (sending information about the state of the drone).

2.6 Reporting of work

A short (about 5 pages) technical report that summarizes the work of the group for the wireless video link is expected. It should cover explanation of the coded VHDL blocks and their behaviour, the results of the testbenches and how were the possible bugs in the code fixed. It may also contain any additional information about the standard that may be relevant to the performance of the drone, as well as reporting of the two additional tasks.

2.7 Provided for the students

- Off the shelf standard compliant receiver
- PCB front-end for the transmitter
- Partial implementation of the DVB-S2 transmitter in VHDL
- Automated standard compliance testbenches
- Software for controlling the transmitter
- Documentation on the standard, VHDL and the DVB-S2 architecture

2.8 Different sub-tasks

First semester

1. Material reading	2 students / 2 weeks
2. VHDL coding	2 students / 4 weeks
3. VHDL debugging and testing	2 students / 3 weeks
4. Integration on the Zybo	2 students / 3 weeks
5. Finishing project / Additional tasks	2 students / 4 weeks
6. Documentation	2 students / 2 weeks

Second semester

As the wireless video link is a crucial part of the EAGLE infrastructure, it needs to be up and running during the first semester.

2.9 Milestones

- Understand and plan phase:
  1. Read VHDL documentation
  2. Read DVB-S2 standard
  3. Familiarize with VHDL compiler and environment

- Modeling phase:
  1. Understand the behaviour of the BCH encoder and how it fits within the complete standard
  2. Implement VHDL code for BCH encoder
  3. Complete VHDL code compiling without errors
- Module phase:
  1. Test VHDL code with provided testbenches and debugging
  2. Full standard compliant transmitter VHDL implementation → **Milestone $\mathbf{T_2}$**
  3. Implement and test VHDL blocks on FPGA
  4. Data-logging system using the provided IP link
  5. Documentation and video streaming demo up and running on the drone → **Milestone $\mathbf{T_3}$**
- Integration phase: wireless video link should already be complete by then.

3 Sub-tasks: detail

- *VHDL coding:* develop the VHDL blocks for the BCH encoder and fit them into the DVB-S2 software implementation in order to have a fully working DVB-S2 transmitter in VHDL. This would be done using *ghdl*, which is an open source VHDL compiler and simulator.
  Outcome: .vhd blocks for the BCH encoder.
- *VHDL debugging and testing:* run the written code through the provided automated testbenches to ensure proper behaviour of the transmitter and standard compliance.
  Outcome: debugged .vhd blocks with fully passed tests.
- *Integration on the Zybo:* after having a working standard compliant transmitter in VHDL, the code should be flashed on the FPGA of the Zybo. The students should also familiarize and configure the provided software for controlling the board in order to have a fully working transmitter working on the drone.
  Outcome: video streaming successfully working on the drone.
- *Finishing project / additional tasks:* Having a the wireless video link up and running on the drone is a mandatory task and should be completed by the students by all means. Additionally, students should benefit from the existing IP link in the provided DVB-S2 architecture and add some data-logging functionality, so the receiver can also get the status of the drone (position, error flags in case of failure, etc). This last part is purposely left-out to their criteria, after some introduction to the architecture is

given to them.
Outcome: video streaming successfully working on the drone / data-logging system.

- *Documentation:* five page small report about the developed VHDL code, reporting bugs and bugfixes after iterating with the testbenches. Open to creativity: other aspects that may influence performance of the drone can also be added.
Outcome: Short report.

4 Provided material explanation

- *Off the shelf standard compliant receiver:* with datasheet to reflect the most important parameters of the standard and their usual values.
- *PCB frontend of the transmitter:* the hardware is provided so the students only need to focus on understanding of the standard and developing the remaining VHDL blocks.
- *Partial implementation of DVB-S2 transmitter in VHDL:* as the standard is very large and complex for the students to fully implement, we are going to give them a working software implementation but without some core blocks. These correspond to the BCH coding (for error correction, where the code rate should be selected). These blocks are chosen for their complexity and the understanding of telecommunication systems that can bring to the students.
- *Automated standard compliance testbenches:* Students may finish their code and try to run it on the FPGA and receive the video stream. If their code does not correctly follow the standard the signal will not be received. To check where the error is, we provide BBC test stream bit-by-bit files of input and output of every block and automated software to compare them with the resulting output of the software DVB-S2 implementation.
- *Software for controlling the transmitter:* both *ULEFramer* and *EagleTx-Controller* (described in section 2.4) software are provided in order for the students to have an implementation that works out of the box in the Zybo board once they flash their VHDL code on the FPGA.

5 Documentation

- GHDL user guide :: Open-source VHDL simulator and compiler.
- GTKWave documentation :: GTK+ based wave viewer.

- Introduction to Zybo HDL

- Introduction to VHDL :: *Free Range VHDL* book :: included in folder. Further reading:
  - Peter J. Ashenden, "The Designer's Guide to VHDL", Third Edition, Morgan Kaufmann 2008. ISBN: 978-0120887859
  - Pong P. Chu, "FPGA Prototyping by VHDL Examples", First Edition, Wiley-Interscience 2008. ISBN: 978-0470185315

- DVB-S2 :: Full standard documentation :: included in folder.

- VHDL blocks and testbenches documentation :: Short introduction about the VHDL blocks provided to the students and automated testbenches :: included in folder.

- CPU software documentation :: Short introduction about the software provided :: included in folder.

- Datasheets :: included in folder.