ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Telecommunications Circuits Laboratory

EPFL, Lausanne

**Student Project at the Dept. of Electrical Engineering**

**Spring 2017**

**for**

**Thomas Verelst**

# Implementation of a packet encoder/decoder pair for the GNU Radio framework

| | |
|---|---|
| Advisor: | Dr Pascal Giard, EPFL-STI-IEL-TCL |
| Co-Advisor: | Orion Afisiadis, EPFL-STI-IEL-TCL |

| | |
|---|---|
| Handout: | 16.12.2016 |
| Due: | 20.02.2017 |

## 1 Introduction

In this student project, a wireless communication system will be implemented in the GNU Radio framework. The GNU Radio framework is the most popular open-source toolkit that provides signal processing blocks to implement software-defined radios. An example of a wireless communication chain is given in Figure 1.

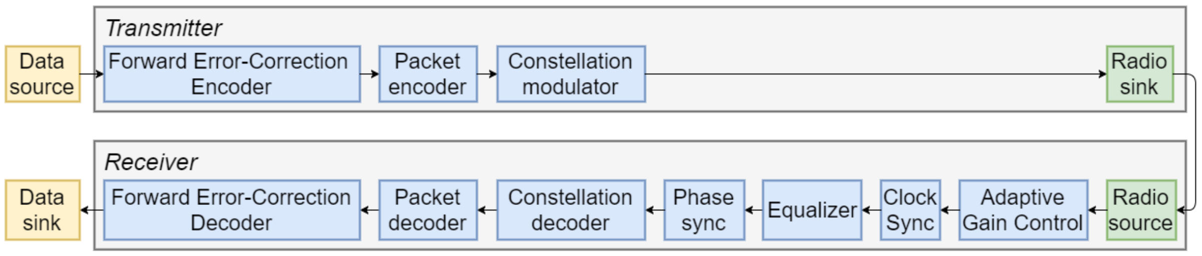

Figure 1: Communication chain

The current packet encoder/decoder implementations will be made more flexible. Specifically, the new packet encoder/decoder should make it possible to use soft-decision error correction and binary phase-shift keying (BPSK) modulation for the preamble, while using higher-order modulations for the payload data.

## 2 Task Description

### 2.1 Objectives

The main objective of this project is to familiarize with the GNU Radio framework, mostly written in Python and C++.

A more flexible packet encoder and decoder block will be implemented. A packet encoder encapsulates the incoming stream into packets, consisting of a preamble, header (containing packet length), payload and checksum. In particular, support for soft channel outputs should be added to enable soft-input error-correction, leading to more reliable wireless communications.

Also, better support for modulating the preamble with BPSK will be added, since this is difficult with the current available blocks in the GNU Radio framework. The synchronization will be modulated with BPSK, while a higher-order modulation is used for other packet data to improve data throughput in good channel conditions.

The blocks should be documented and of sufficient quality to be contributed back into the free software framework. Ideally, the core of the blocks should be sufficiently decoupled from GNU Radio so that they could easily be integrated into another framework (e.g. NI Labview) at the cost of writing another wrapper.

## 2.2 Specific Tasks

1. Review the existing packet encoder/decoder blocks.

2. Adapt the existing blocks or write new ones to support soft-input error correction.

3. Add support for distinct modulations for the preamble and payload e.g. BPSK for the preamble and some higher-order modulation for the rest of the packet.

4. If time permits, the current GNU Radio blocks can be optimized or extra functionality can be implemented. This can consist of:

   - Improve the performance of the packet encoder/decoder (or another processing block that might be a bottleneck) by using advanced C++ techniques, such as using SIMD instructions, assembly code or multithreaded processing. The optimization should be benchmarked to quantify the improvement.
   - Implement new blocks, e.g. blocks for live video transmission.

5. Iteratively setup a system to demonstrate the implemented blocks. In the first iteration, the transmitter/receiver is simulated. In the second iteration, two Universal Software Radio Peripherals (USRPs) are used but transmission is done over a wire with an attenuator. In the third and final iteration, transmission is done over the air.

6. Give a 20 minutes presentation in front of an audience, covering the project results and demonstrating the implemented project.

7. Write a report describing the results of the project.

8. Contribute the implemented blocks back into the free software framework.

9. In case the project is of high quality, summarize the key results in a 4 to 6 pages article to be submitted at a scientific conference.

### 2.3 Expected Results

Significantly better GNU Radio packet encoder/decoder blocks allowing the use of soft-input error-correction, and that supports the use of one modulation for synchronization (preamble) along with a different one for the remainder of the packet. For the same operating conditions, the former reduces the error rate while the latter improves synchronization stability at greater throughput.

## 3 Administrative Considerations and Recommendations

### 3.1 Administrative Considerations

- Regular meetings with your advisors to evaluate the status of work and to plan the next steps are a necessary precondition for successful completion of your student project.
- You will be assigned a desk and a workstation for carrying out the work.
- About the report:
  - The report comprises a detailed description of the architecture of your final design and of the tradeoffs you encountered during your work.
  - Include block diagrams and flow-charts whenever they help to clarify your architecture or the design flow.
  - Write about decisions you had to make and the corresponding reasoning. Evaluate what others have done and justify your architecture(s) and implementation(s).
  - Include a short data-sheet for your RTL implementation(s). This datasheet should briefly summarize the functionality and should provide all the necessary information to use your RTL code as a building block for a larger design (e.g., clearly describe interfaces and configuration information).
  - Emphasize the achievements of your work.
  - Pay attention to the structure of your report.
  - One of the bonus goals of the project is to publish your work to a conference, thus be careful and detailed while writing your report.
- Prepare a compressed archive that contains all relevant files and data used and generated during the project.

- Your student project has to be presented in front of the laboratory members once you have finished your work.

## 3.2 General Recommendations

- Keep it simple. If any portion of your code or your architecture looks complicated, have another look.
- Plan enough time for verification and, when applicable, testing.
- Plan enough time for documentation.
- The tools (MATLAB, Synopsys Design Compiler, Cadence Encounter, ...) that you are going to use are extensively used in both academia and industry. Thus, several tutorials are available on the web. The knowledge that you will aqcuire will certainly be useful in the near future.
  - VLSI-related tutorials can be found at: `http://www.vtvt.ece.vt.edu/vlsidesign/index.php`.
  - Tutorials on ASIC design software as well as on the design environment at EPFL are available at: `http://edadk.epfl.ch`.
- Try to get used to compilation scripts (and using command line options for the tools). They may be a little more time consuming the first time around, but they will make life easier during the numerous iterations needed to reach the final design.
- If you are looking for references, like conference or journal paper, use...
  - IEEEXplore: `http://ieeexplore.ieee.org/search/advsearch.jsp`
  - Google Scholar: `http://scholar.google.com`