Software Requirements Specification
Real-time C++ Implementation of OFDM

Perpared by: Kamil Rog
Spproved by:

June 23, 2021

Revision History

Still under editing for first release

Contents

1 Introduction

1.1 Purpose

This specification is a part of the document set inspired by IEEE software life cycle, specifically software requirements specification(SRS) IEEE 29148 which purpose in this project is to identify key requirements leading up and for the first major release (v1.0) of orthogonal frequency-division multiplexing(OFDM) library, ofdmlib and exemplary use case, an audio demo application that is also used to increase robustness of the library through highlighting practical implementation issues.

1.2 Document Conventions

In this specification, higher-level requirements are assumed to be inherited by detailed requirements

1.3 Intended Audience and Reading Suggestions

This document has been written as guide to identify key functional and performance requirements for and leading up to the first major release (v1.0) of OFDM library. OFDM modulation scheme is complex thus it is recommended to have a fundamental understanding of the concept prior to reading this document to fully understand it's contents. A highly recommended introduction material to the topic can be found at (youtube link for Dr.Porr Lectures)
In addition this document specifies purpose of the library, it's intended scope and limitations. Future developers, testers and project managers do not have to read this document in it's entirety, the attention of these groups can be focused only on sections 2.2-2.6 which give a list of the main features, the intended audience and limitations. Sections 3.3-3.4 provide a detailed account of the interfacing and 4 briefly describes each feature and specifies the requirements.
For those who are considering using ofdmlib please confirm the licences of ofdmlib and its dependencies for your the intended use (section 3.3).

1.4 Product Scope

OFDM library aims to provide configurable, robust and efficient real-time C++ coders for systems operating on xxxx platform. This library is an attempt to encourage developers to use the library in their projects and hopefully(but not required to) share the developed code in public domain or provide feedback and feature requests for future releases.

1.5 References

Full project documentation can be found at: https://github.com/krogk/ofdmlib/docs

2 Overall Description

2.1 Product Perspective

Currently no efficient OFDM implementation is available in public domain, this software package is intended to address this issue by providing a C++ library for such use. The implementation hopes to standardise the ofdm concept across products.

2.2 Product Functions

Encoder (Tx Chain):

- Energy Dispersal
- QAM Modulator
- Pilot Tones Addition
- Inverse Fast Fourier Transform
- Band-Pass Quadrature Modulator
- Cyclic-Prefix

Decoder (Rx Chain):

- Cross-correlator
- Band-Pass Quadrature Demodulator
- Fast Fourier Transform
- Pilot Tone Detector
- Channel Estimation
- Equalisation
- QAM Demodulator
- Reverse of Energy Dispersal

2.3 User Classes and Characteristics

Library is intended for C++ developers who are timed-constrained or lack the full technical expertise and would like to implement OFDM scheme in their projects. The library aims to be professional.

2.4 Operating Environment

Currently ofdmlib is being developed on Linux operating system, Ubuntu Focal.

2.5 Design and Implementation Constraints

The first major release is aimed to be delivered only for Linux operating system. Open Source

2.6 Documentation

The following documentation is going to be delivered along with the software.

- SRS – Software requirements specification - IEEE 29148
- SDD – Software design description - IEEE 1016
- STD – Software test documentation - IEEE 29119
- SUD – Software user documentation - IEEE 24748
- Doxgyen - UML Generation From Code

2.7 Assumptions and Dependencies

The core of OFDM is the Fourier and inverse Fourier transforms, there are numerous libraries in public domain which are highly optimised therefore the ofdmlib is going to utilize such libraries, this has legal/financial implications for potential developers but as per required delivery date for first major release this is the preferable choice, in the future the feasibility of developing own FFT and IFFT algorithms can be assessed.

3 External Interface Requirements

3.1 User Interfaces

OFDM is library not a stand-alone application, therefore no GUI is required, the working demo delivered alongside the library is intended to be minimalistic and operated from command line hence no GUI requirements specification has been produced for this project.
Error message standards

3.2 Audio Demo Interfaces

Hardware Interfaces

The development on PC running Ubuntu 20 means there are kernel audio driver which can be used. Which is based on I2S communication Protocol.
The developers

Software Interfaces

Supported Operating Systems:

- Ubuntu

Dependencies:

- FFTW3

Development Tools and Technologies:

- C++ 11
- Cmake 3.20.2
- BOOST

3.3 Communications Interfaces

ofdmlib does not contain any security or encryption features.

4 System Features

The description of the features is brief as a detailed account of each feature purpose has been given in the section x.x theory.

Priority of the feature scales from low(1) to high(5). For the release 1.0 and all releases leading up to it all features are crucial therefore the priority has not been given

4.1 System

Data

REQ-1: The underlying FFT and IFFT needs to work on one data type, double float

REQ-2: The ofdm coders must handle most common data types at the input.

REQ-3:

Coder object

Description and Priority

The coder object wraps all the objects related to ofdm modulation and provides easy to use interface.

Functional Requirements:

REQ-1: The encoder and decoder settings need to be common and implemented as one object

REQ-2: Codec must have a setting that specifies its role; encoder or decoder, configure and use appropriate functions

REQ-3: Setting variables must be private/protected.

REQ-4: The codec must contain a set of functions to get and set each setting.

4.2 OFDM Functional Objects (Tx & Rx Chain)

Energy Dispersal

Description

The coders must have the ability to randomise the incoming data to make the signal as random as possible.

Priority: 5

Stimulus/Response Sequences:

Functional Requirements:

REQ-1: The encoder must have the ability to pseudo-randomise incoming data. The decoder must obtain the original data from the pseudo randomised data

REQ-2: The seed must be set as a part of the coder initialization process.

REQ-3: The seed must be read and modifiable through set and get functions.

REQ-4: The energy dispersal must operate on the same data type as the data to be encoded.

REQ-5: The energy dispersal must take the pointer to the data and randomise it.

Note: For performance the Energy Dispersal and QAM De/Modulator can be integrated

Digital Quadrature De/Modulator

Description

The encoder needs to provide a flexible quadrature amplitude modulator

Priority: 5

Stimulus/Response Sequences:

Functional Requirements:

REQ-1: Must create a binary stream from the incoming data and encoded it using specified scheme.

REQ-2: The modulation scheme must be set as a part of the coder initialization process.

REQ-3: The modulation scheme must be read and modifiable through set and get functions.

REQ-4: The modulation scheme must be 4 QAM.

Note: 16,64,256 QAM could be developed if there is some time left before deadline.

Pilot Tones

Description: The coders need to inject real valued tones at specified locations for the

Stimulus/Response Sequences:

When QAM encoded data is to be inverse Fourier transformed into time domain a pilot tones need to be injected into an array of data at specified indices.

Functional Requirements:

REQ-1: Pilot tones cannot cause erasures or errors. Data needs to be manipulated appropriately to prevent this.

REQ-2: The Pilot tone locations can be specified as a number of samples between next tones with an offset or as an array of indices.

REQ-3: The pilot tone configuration, amplitude and locations, must be set as a part of the coder initialization process.

REQ-4: The pilot tone configuration must be read and modifiable through set and get functions.

REQ-5: Ideally the pilot tones should be set only once, upon the initialization of the IFFT input buffer or on configuration change.

REQ-6: Decoder must sum all imaginary parts of the FFT.

Fourier and Inverse Fourier Transform

Description

Discrete fast Fourier Transforms and are the key aspects of the OFDM

Priority: 5

Stimulus/Response Sequences

On the encoding side, the inverse Fourier transform occurs after the QAM encoding and

Functional Requirements

REQ-1: FFT buffers and methods must be encapsulated in one object.

REQ-2: The OFDM FFT object must manage memory appropriately to the specified settings.

REQ-3: The OFDM FFT object must configure the direction of the transform.

REQ-4: The choice between real(mirrored spectrum) and complex time series.

REQ-5: FFT object configuration must be set as part of the coder initialization process.

REQ-6: FFT object configuration must be read and modifiable through set and get functions.

Band-Pass Modulator

Description

Band-pass modulator upsamples the signal by interleaving real and complex samples.

Priority: 5

Functional Requirements

REQ-1: Upsample the signal at lowest rate

Note: For performance purposes the Band-Pass De/Modulator can be combined with cyclic prefix.

Cyclic-Prefix

Description

The Cyclic-prefix is added at the front of each symbol this occurs after up-

sampling by band-pass modulator and this is essentially the last step in the modulation scheme.

Priority: 5

Stimulus/Response Sequences

Functional Requirements:

REQ-1: The cyclic-prefix must be added in encoding process and used in the decoder as a part of symbol start detection process.

REQ-2: The cyclic-prefix length must be encapsulated in settings object.

Channel Estimation

Description

Priority: 5

Stimulus/Response Sequences

Functional Requirements:

REQ-1:

REQ-2:

Equalisation

Description and Priority

Priority: 5

Stimulus/Response Sequences

Functional Requirements:

REQ-1:

REQ-2:

5 Non-functional Requirements

Architecture

REQ-1: The architecture of the software should allow for easy multi-threaded implementation for the engineer.

5.1 Performance Requirements

FFTW3 Uses its own format for complex numbers, the (I)FFT transform outputs in this format in later releases these functions could be modified to provided already upsampled output.

Filters

REQ-1: Filters need to be implemented in efficient fashion without copying the data sets.

5.2 Software Quality Attributes

Test Framework

The approach to this project is test driven development. A sophisticated Test framework must be developed from the start of the implementation of the software package to provide confidence that the functionality discussed in the section 4.1 has been successfully implemented.

REQ-1: Each feature must have a unit test.

REQ-2: Each feature must be implemented into an integration test.

REQ-3: Each unit test and integration test must display a measure of performance, execution time or clock cycles.