Flexible GFDM PHY

December 12, 2017

Author Ana B. Martinez
ana-belen.martinez@ifn.et.tu-dresden.de

Contributors
Martin Danneberg
Shahab Ehsanfar
Zhitao Lin
Maximilian Matthe
Ahmad Nimr
Dan Zhang

Table of Contents

1 Introduction

This document provides basic information about how to get started with the *Flexible-GFDM-PHY* project. The user is expected to have previous knowlegde about the LabVIEW environment and programming.

2 System Requirements

The *Flexible-GFDM-PHY* project has been implemented on a National Instruments (NI) Software Defined Radio (SDR) platform. For the development, NI USRP-RIOs 2953R with 40 MHz bandwidth and LabVIEW Communications Design Suite 2.0 have been used [1] [2].
The connection between the USRP-RIO devices and the control desktop computers has been done via an NI PCIe-8371 card. The project has been tested via cable with a 30 dB attenuator as well as over the air with antennas VERT2450 [3].
Ensure that you are in compliance with all local laws before running this project with antennas to avoid a possible violation of local laws.
Compatibility with software and hardware versions different from the ones named in this document cannot be guaranteed.

3 URSP-RIO Setup

The project can run on one single USRP-RIO with 40 MHz or 120 MHz bandwidth or on two different USRP-RIO devices. In both cases the transmission can be realized either via a cable or over the air with antennas.

3.1 Setup with one USRP-RIO

The communication always takes place between one TX port and one RX port. In this setup, we will choose transmitter and receiver on different RF channels of the USRP-RIO. Therefore, the two possible configurations are:

1. from RF0/TX1 to RF1/RX2
2. from RF1/TX1 to RF0/RX2

For the communication via cable, an additional 30 dB attenuator has to be connected between one end of the cable and one of the ports. In case of transmission over the air, antennas have to be located at the corresponding ports.

3.2 Setup with two USRP-RIOs

In this setup, each USRP-RIO is controlled by a separate desktop computer. Multiple configurations are possible, considering that the communication can only occur from one TX port of one USRP-RIO to one RX port of the other USRP-RIO.

4 Overview

The compressed file *Flexibe-GFDM-PHY-Master.zip* with the LabVIEW implementation of the GFDM transceiver can be downloaded from the website: "https://github.com/ewine-project/Flexible-GFDM-PHY".
Once this file is copied and extracted to the hard disk drive, it can be opened with NI Labview Communications Design Suite 2.0.

4.1 Folder structure

The **GFDM PHY.lvproject** consists of different VIs:

1. **GFDM Host.gvi** - This host VI allows the configuration of the transceiver and interfaces with the bitfile **GFDM FPGA TopLevel for 40MHz USRP.lvbitx**.
2. **GFDM FPGA Toplevel for 40MHz USRP.gvi** - Top-level FPGA VI.
3. **UDP data generator and receiver.gvi** - Implements the communication via UDP.

and folders:

1. **Builds** - Contains precompiled bitfiles.
2. **FPGA** - Organized into subfolders with FPGA VIs, addressing different functionalities of the transceiver.
3. **Host** - Comprises Host VIs and subfolders for different functions.
4. **NI VIs** - Contains VIs developed by NI.
5. **Variables** - Includes resources and variables.

5 Running Flexible GFDM PHY Project

1. Lunch LabVIEW Communications System Design Suite.
2. Open the project **GFDM PHY.lvproject**.
3. In the Files pane, open the host VI **GFDM Host.gvi**.
4. Configure the parameters for the proper operation of the transceiver as well as the video stream application in case of UDP communication. This part will be explained in the following sections.
5. Runk the LabVIEW host VI by clicking Run (the green button).

6 Configuration of parameters in GFDM Host.gvi

Figure 1 shows the transceiver user interface. The tabs on the right part (TX, RX and Data) are dedicated to the configuration of the parameters. Results corresponding to different modules of the transceiver are displayed on the tabs of the left part. Although not shown here, outside the user interface part, this VI displays important parameters with the configuration of the FPGA, which add information about the internal operation of the transceiver.

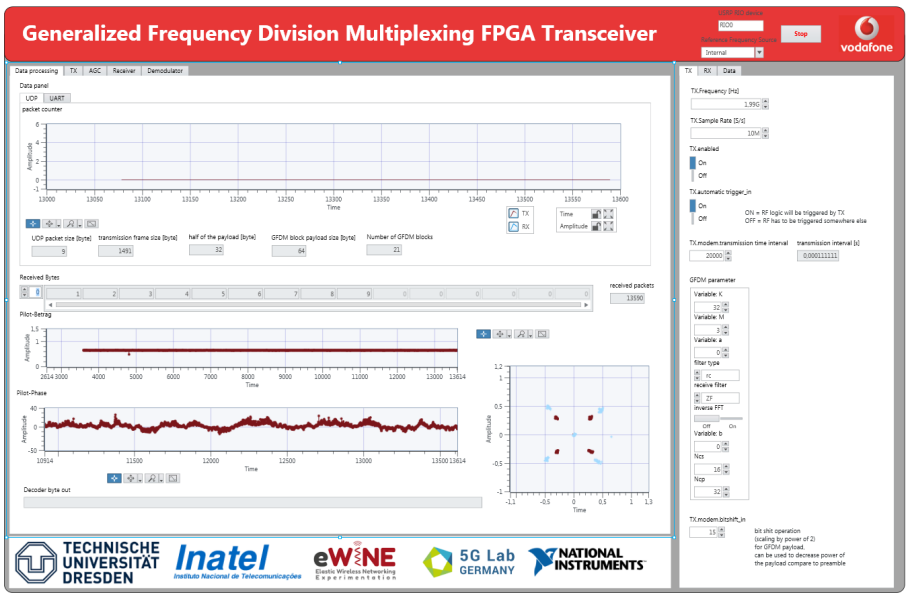

Figure 1: GFDM Host transceiver user interface. Data processing results (left) and parameter configuration (right).

The first step to take is the configuration of the USRP-RIO in the upper right part of the screen. The NI Measurement & Automation Explorer (MAX) provides the RIO identifier of the device. In addition, the Reference Frequency Source has to be selected among:

1. **Internal** - Internal reference clock.
2. **REF IN** - Reference is taken from the the REF IN port.
3. **GPS** - The GPS module provides the clock reference.

The following subsections describe the parameters of the configuration tabs. Fields with white backgroud are configuration fields, while fields with grey background are indicators, i.e., they value cannot be changed. Some of the parameters of these tabs are used for internal operations and can be left to their

default values. This document focuses on those parameters that need to be adapted for proper operation of the project.

6.1 TX

1. **TX.Frequency [Hz]** - Carrier frequency in Hz.
2. **TX.Sample Rate [S/s]** - Sampling frequency in samples per second.
3. **TX.enabled** - Enables/disables the transmitter.
4. **TX.automatic trigger_in** - If it is set to On, the transmitter triggers the RF logic. Otherwise another event has to activate it.
5. **TX.modem.tranmission time interval** - Sets the number of cycles the transmitter has to wait until it can start generating a new packet.
6. **tranmission time interval [s]** - Previous value expressed in seconds.
7. **GFDM parameter**
   (a) **Variable: K** - Number of subcarriers.
   (b) **Variable: M** - Number of subsymbols.
   (c) **Variable: a** - Roll-off factor of the prototype filter (between 0 and 1).
   (d) **filter type** - The prototype filter can be chosen between raised cosine (rc) or rectangular in time domain (rect).
   (e) **receive filter** - Zero Forcing (ZF) or Matched Filter (MF).
   (f) **inverse FFT** - Set to default value Off for common operation (FFT is performed in modulator and inverse FFT in detector).
   (g) **Variable: b** - Roll-off factor of the time domain window.
   (h) **Ncs** - Length of cyclic suffix in samples.
   (i) **Ncp** - Length of cyclic prefix in samples.
8. **TX.modem.bitshift_in** - Shift (in bits) to scale the GFDM payload. This option can be useful, e.g., in case the preamble is sent with higher power than the payload.

6.2 RX

1. **RX.Frequency [Hz]** - Carrier frequency in Hz.
2. **RX.Sample Rate [S/s]** - Sampling frequency in samples per second.
3. **RX.enabled** - Enables/disables the receiver.

4. **Gain-related parameters** - The receiver can work with a fixed or an adaptive gain. In case of an adaptive gain, different parameters are necessary for internal calculations. These fields can be left to their default values.
   (a) **Fixed Gain Setting** - Fixed (On) or adaptive (Off) gain.
   (b) **gain in** - Fixed value of gain to work with in case **Fixed Gain Setting** is set to On.
   (c) **AGC.milliseconds to wait** - Time to wait (in milliseconds) to adapt the gain.
   (d) **expected signal range upper bound** - Parameter needed to adjust the gain depending on the previous value of gain and the measured power. Default value: -18.
   (e) **expected signal range lower bound** - Parameter needed to adjust the gain depending on the previous value of gain and the measured power. Default value: -22.
   (f) **RX.Sync.rssi.time constant** - Time constant used during internal power calculation (after synchronization). Default value: 14.
   (g) **RX.rssi.time constant** - Time constant used during internal power calculation (before synchronization). Default value: 14.
5. **Channel Length** - Allows to adjust the length of the estimated Channel Impulse Response (CIR), i.e. the number of coefficients that will be used to calculate the channel frequency response with $N = K \cdot M$ samples.
6. **RX.modem.bitshift_in** - This parameter indicates the number of bits used internally to make a conversion from 64 to 32 bits in the core modem.
7. **RX.modem.fft.bitshift_in** - As in the previous case, a proper scaling is necessary to convert a 64-bit signal to a 32-bit signal at the output of the fft block.
8. **RX.switch channels** - Indicates which RF port is used for the reception. Select On to use the RX port on RF0 and Off for RF1.
9. **Sync Configuration Cluster**
   (a) **Metric peak threshold mode** - Fixed (Off) or adaptive (On) threshold.
   (b) **Metric peak threshold** - Threshold to use for synchronization peaks detection if **Metric peak threshold mode** is set to Off.
   (c) **min_threshold** - Minimum threshold used to detect synchronization peaks. A low value higher than zero has to be chosen.
   (d) **shift_threshold** - Scales threshold calculated to detect synchronization peaks.

(e) **CFO removal** - Indicates if CFO has to be compensated.

(f) **ffoPiControlCte** - Constant to control the speed of fractional frequency offset compensation.

(g) **autocorrelation_energythreshold** - Avoids division by zero when calculating the normalized autocorrelation. Set this parameter to a low value higher than zero.

(h) **addToMultipath** - Internal parameter.

(i) **ThresholdWindow** - Internal parameter.

(j) **Sync_scale Threshold** - Internal parameter.

(k) **halfPreambleVectorIn** - Repeated part of the preamble used for synchronization and channel estimation. The value of this parameter is read from a file, therefore, this field can be left by default.

6.3 Data

1. **PayloadSize [Byte]** - Payload size in bytes.
2. **Data Routing** - Determines the type of communication among:
   (a) **Data from Host**
   (b) **Data from UART**
   (c) **UART Loopback**
3. **K_set** - Displays the set of allocated subcarriers.
4. **M_set** - Displays the set of allocated subsymbols.
5. **UART**-related parameters are used in case **Data Routing** is not set to **Data from Host**. These parameters can be divided into control and status ones.
6. **Add frame counter** - Activates the frame counter.
7. **Read UDP?** - If it is set On, UDP data is used. Otherwise a ramp signal with **PayloadSize** length is generated. UDP data can be chosen when using an external video streaming application, e.g. VLC, as in this case. The transmitter receives the source data from the video stream player and the receiver sends the data to the player to be displayed.
   Configuration of VLC:
   (a) Transmitter
      i. Start cmd.exe and change to the VLC installation directory
      ii. Identify the path and name of the video file that will be used for streaming "video_file"

iii. Start the VLC application as a streaming server with the following command:
vlc "video_file"
:sout=#std{access=udp{ttl=1},mux=ts,dst=127.0.0.0:"vlc_tx_port"}
where "vlc_tx_port" corresponds to **UDP receive port** in the **Data** configuration tab.

(b) Receiver

i. From the command line window, start the VLC application as a streaming client with the command:
vlc udp://@:"vlc_rx_port"
where "vlc_rx_port" corresponds to **UDP send port** in the **Data** configuration tab.

8. **RM:No DC** - Leaves unused the subcarrier corresponding to DC.

9. **Use Phase Tracking** -Applies Common Phase Error Compensation.

10. **UDP receive port** - Number of UDP port used to receive data from the video stream player.

11. **UDP send port** - Number of UDP port used to send data to the video stream player.

7 GFDM FPGA TopLevel for 40MHz USRP

This section describes briefly the functionality of the Clock Driven Logic (CDL) loops of the top-level FPGA VI and adds references to relevant literature for further information.

1. **Data Routing** - Different FIFOs are configured in this CDL depending on the type of data chosen in the field **Data Routing**.

2. **Encoder** - Performs convolutional encoding and QPSK mapping.

3. **Resource Mapper - Demapper** - The resource mapper mapps data and control signals before the modulation, whereas the demapper demapps the data coming from the detector and applies QPSK demapping prior to the decoding.

4. **Decoder** - The decoder takes as input the already demapped data from the previous CDL and applies Viterbi decoding.

5. **Detector** - This CDL contains the channel equalizer and the core modem acting as demodulator.

6. **Modulator** - Performs Generalized Frequency Division Multiplexing (GFDM) modulation [4] using the core modem. Afterwards Cyclic Prefix (CP) and Cyclic Suffix (CS) insertion as well as windowing take place. Finally, a

preamble, which is used for synchronization and channel estimation purposes, is prepended to the modulated data.

7. **Synchronizer** - Receives the data from the Analog Digital Converter (ADC) and performs synchronization according to the algorithm proposed in [5].

8. **Register Bus and Required Controls/Indicators** - In this CDL registers needed for the appropriate operation of the transceiver are defined and initialized.

9. **Streaming Transceiver Engine** - Contains original code from NI with the definition of the streaming engine used as a baseline for the transceiver. This CDL receives from the modulator the signal to be transmitted and provides as output the received signal after the ADC operation.

References

[1] National Instruments. *NI USRP RIO. Data Sheet*. URL: http://http://www.ni.com/datasheet/pdf/en/ds-538.

[2] National Instruments. *Getting Started Guide. USRP 2950/2952/2953/2954/2955*. URL: http://www.ni.com/pdf/manuals/376355c.pdf.

[3] A National Instruments Company. Ettus Research. *VERT2450 Antenna*. URL: https://www.ettus.com/product/details/VERT2450.

[4] Martin Danneberg et al. "Flexible GFDM Implementation in FPGA with Support to Run-Time Reconfiguration". In: *VTC Fall*. IEEE, 2015.

[5] Ivan S. Gaspar et al. "A synchronization technique for generalized frequency division multiplexing". In: *EURASIP Journal on Advances in Signal Processing*. 2014.