An optimized GFDM software implementation for future Cloud-RAN and field tests

Johannes Demel DEMEL@ANT.UNI-BREMEN.DE
Carsten Bockelmann BOCKELMANN@ANT.UNI-BREMEN.DE
Armin Dekorsy DEKORSY@ANT.UNI-BREMEN.DE
Department of Communications Engineering, University of Bremen, Bremen, Germany

Andrej Rode ANDREJ.RODE@STUDENT.KIT.EDU
Sebastian Koslowski SEBASTIAN.KOSLOWSKI@KIT.EDU
Friedrich K. Jondral FRIEDRICH.JONDRAL@KIT.EDU
Communications Engineering Lab, Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract

5th Generation (5G) and Industry 4.0 (I4.0) cellular systems have many different use-cases which impose a diverse set of requirements on a candidate Multi-Carrier System (MCS). These requirements demand high flexibility and reconfigurability from any candidate waveform. In this paper we present *gr-gfdm* a Software-Defined Radio (SDR) implementation of the waveform candidate Generalized Frequency Division Multiplexing (GFDM) in GNU Radio, discuss the details of our implementation and show how the SDR flexibility can be employed for use in simulations and field trials. With a careful design the same implementation can be used in testing, simulation, field trials and interfaces to other programming languages are easily added. Furthermore we facilitate an over-the-air transmission with our implementation in order to demonstrate its capabilities, where we could achieve sample rates of up to $25\,\mathrm{MS/s}$.

1. Introduction

Current implementations of waveform candidates are focused on investigations of their performance in an academic setup or they are implemented in hardware with Field Programmable Gate Arrays (FPGAs) (Danneberg et al., 2015). This results in rather high performance losses in the software implementation or missing flexibility in the hardware implementation. For further analysis of new waveforms a performant SDR implementation which can be used in both, simulation and field trials, is desirable since it retains its flexibility without suffering much performance losses. Additionally, an SDR implementation is crucial for Cloud Radio Access Network (Cloud RAN) in order to move the computing away from the Remote Radio Heads (RRHs). Thus, we present the GFDM sofware implementation *gr-gfdm* that offers the required performance and flexibility (Rode, 2017).

1.1. Waveform

For upcoming 5G and I4.0 applications new MCS candidates are proposed. These waveforms try to overcome Orthogonal Frequency Division Multiplexing (OFDM) shortcomings, such as high Out-Of-Band (OOB) emissions while retaining its advantages.

One of the goals of the upcoming 5G standard is to provide a common communication standard for all types of modern communication. This includes data driven applications for end users where high data rates are required but also Machine-type-Communication (MTC) for industrial applications where latency and reliability are crucial. Each type of communication imposes its own requirements on the physical layer, but the waveform does not have to meet requirements for all communication types at the same time. This is an opportunity to introduce a flexible waveform implemented in software which can be reconfigured on the fly for different types of communication.

For I4.0 closed loop control the waveform can be parameterized to meet low latency and robustness requirements and for high data throughput the parameters can be changed to achieve maximal spectrum efficiency and throughput. For the envisioned cooperative environment new wave-

Proceedings of the 7^{th} GNU Radio Conference, Copyright 2017 by the author(s).

forms require low OOB emissions in order to coexist in a heterogeneous environment.

Current 4th Generation (4G) systems rely on OFDM which is a simple and effective MCS. However, several shortcomings with regards to OFDM were identified (Schaich & Wild, 2014). As the spectrum becomes more and more crowded while bandwidth demands increase, the coexistence properties of new waveforms need to be improved too. Aside of poor OOB emission properties, strict synchronization requirements and poor spectral efficiency are further OFDM shortcomings.

Several MCS candidate waveforms exist which offer different approaches to overcome OFDM shortcomings (Sahin et al., 2014). Filter-Bank Multi-Carrier (FBMC) minimizes OOB emissions by filtering each subcarrier but introduces large filter delays (Schaich & Wild, 2014). Universal Filterbank Multi-Carrier (UFMC) groups multiple subcarriers and then filters each group jointly in order to decrease filter delays, though it still only considers timeslots individually (Vakilian et al., 2013).

GFDM goes beyond symbol-based modulation by modulating entire frames (Michailow et al., 2014). Generally, GFDM is a highly flexible non-orthogonal waveform. Circular filters retain the option to use a Cyclic Prefix (CP) and a Cyclic Suffix (CS) for whole frames. Furthermore, these filters avoid large delays and can minimize OOB emissions. The parametrization can then be adjusted for either low latency requirements or high data throughput and spectral effiency.

1.2. Software Defined Radio

Straight forward MCS implementations often suffer from very heavy computational demands which makes their implementation on General Purpose Processor (GPP) intractable for SDR applications. Efficient software implementations for simulations and field tests are crucial to verify the performance of these new MCS. This enables parameterization of the waveform during run-time and bay be performed in a computation center in the Cloud RAN. This further stresses the need for an optimized software implementation of the used waveform.

In (Danneberg et al., 2015) a GFDM implementation in LabView targeting the Universal Software Radio Peripheral (USRP) X310 is presented. All relevant parts of the signal processing chain are implemented for this specific hardware and run on an FPGA. This contradicts the target to use the same codebase for simulations and field trials as well as its portability. An SDR implementation for simulations and field tests requires more portability and flexibility in order to adapt the communication system for its current purpose.

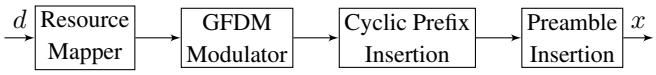

Figure 1. Transmitter processing steps

1.3. Main Contribution

The main contribution of this paper is a presentation of the open source SDR GFDM implementation *gr-gfdm* in GNU Radio (Rode, 2017). The implementation aims to fill the gap between slow software simulation implementations and inflexible hardware implementations. We discuss the theoretical background as well as the implementation with its design concepts. The paper is concluded with a presentation of over-the-air transmission capabilites of the implementation.

2. GFDM system description

This Section introduces the concepts of Generalized Frequency Division Multiplexing (GFDM). It is split into a transmitter and receiver subsections, depicted in Fig. 1 and 2, which detail the specifics of the system. A stream of complex symbols $d \in \mathbb{C}$ goes into the transmitter and a stream of estimates for the transmitted symbols $\hat{d}$ is produced by the receiver. In contrast to OFDM, GFDM focuses on whole frames instead of individual timeslots in order to optimize its transmission properties. This approach introduces more flexibility and thus, GFDM can be better matched to individual use-cases. Also, latencies may be minimized by designing a system accordingly.

2.1. Transmitter

The transmitter portion of a GFDM system is composed of three stages depicted in Fig. 1. In this paper we do not consider channel coding but we expect it to be part of a complete system. Multiple complex symbols d are grouped into a frame and assigned to their point on the time-frequency plane, or lattice (Sahin et al., 2014). The modulator transforms this frame into a complex baseband signal vector. A CP is added in order to obtain the cyclic channel properties at the receiver and thus enable simple one-tap Frequency-Domain Equalization (FDE).

MCS waveforms modulate symbols such that each symbol in a frame with M timeslots and K subcarriers is located at its unique point on the lattice. Thus, a maximum of $N = KM$ points may be transmitted. The lattice for a GFDM frame is represented by the matrix $\boldsymbol{D} \in \mathbb{C}^{M \times K}$ where each element $d_{m,k}$ corresponds to a symbol in the mth timeslot on the kth subcarrier (Michailow et al., 2012). Often, some subcarriers are not used for transmission but only $K_{\text{on}} \leq K$ are used.

In case $K_{\text{on}} < K$, subcarriers which are unused correspond to columns in $\boldsymbol{D}$ which are filled with zeros and consequently the occupied bandwidth is reduced. Thus, the resource mapper groups MK_{on} complex symbols together with $M(K - K_{\text{on}})$ zeros into $\boldsymbol{D}$. Stacking $\boldsymbol{D}$'s columns $\boldsymbol{d}_k$ according to

$$\boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_0^T & \boldsymbol{d}_1^T & \dots & \boldsymbol{d}_k^T & \dots & \boldsymbol{d}_{K-2}^T & \boldsymbol{d}_{K-1}^T \end{bmatrix}^T \quad (1)$$

returns a symbol vector $\boldsymbol{d} \in \mathbb{C}^{N\times 1}$ containing all symbols of a GFDM frame.

GFDM modulation is a linear operation which can be denoted in matrix notation

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{d} \quad (2)$$

where $\boldsymbol{x}$ is the transmit vector and $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ is a modulation matrix containing the filter coefficients for one symbol in each column (Michailow et al., 2014). This modulation matrix $\boldsymbol{A}$ can be written as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{g}_{0,0} & \boldsymbol{g}_{0,1} & \dots & \boldsymbol{g}_{1,0} & \dots & \boldsymbol{g}_{K-1,M-1} \end{bmatrix} \quad (3)$$

where the columns represent filters derived from a prototype filter $\boldsymbol{g} \in \mathbb{C}^{N\times 1}$. The n-th element of the derived filter for the k-th subcarrier in the m-th timeslot is obtained by modulating and circularly shifting the prototype filter

$$g_{k,m}[n] = g[(n - mK) \mod N] \cdot e^{j2\pi n \frac{k}{K}}. \quad (4)$$

From (2) it can be observed that GFDM is a linear, frame-based multicarrier modulation scheme. The desired spectral properties can be matched by means of the chosen prototype filter $\boldsymbol{g}$, e.g. Root-Raised-Cosine (RRC) or Gaussian filters. The chosen prototype filter may constitute orthogonal or non-orthogonal modulation and thus controls how much self-interference is present in a specific GFDM system. Appropriate prototype filter design may be performed with the aid of ambiguity functions (Matthias Woltering et al., 2015; Du, 2008). In general, non-orthogonal modulation must be assumed and self-interference must be considered.

Due to the cyclic filter shift in (4), a GFDM frame is cyclic. Thus, a CP can be employed to obtain a cyclic channel at the receiver and one-tap FDE is feasible. In contrast to OFDM, only one CP per frame is necessary which can be exploited to shorten frames and thus in turn reduce latency.

Matrix multiplication is an expensive operation, especially when N tends to be large. Frequency domain modulation and thus demodulation promises to drastically reduce complexity (Gaspar et al., 2013). Therefore, (2) can be rewritten to

$$\boldsymbol{x} = \boldsymbol{\mathcal{F}}_N^{-1} \sum_{k=0}^{K-1} \boldsymbol{P}_{N\times ML}^{(k)} \boldsymbol{G}_{ML\times ML} \boldsymbol{R}_{ML\times M} \boldsymbol{\mathcal{F}}_M \boldsymbol{d}_k \quad (5)$$

where $\boldsymbol{d}_k$ denotes the complex symbols modulated onto one subcarrier. First, these symbols are transformed to frequency domain with an M-point Fourier transform $\boldsymbol{\mathcal{F}}_M$. Next, upsampling in frequency domain is performed by means of a repetition matrix $\boldsymbol{R}_{ML\times M}$, where $L \leq K$ is the overlap factor. $\boldsymbol{G}_{ML\times ML}$ is a diagonal filter matrix with the ML prototype filter taps on its diagonal. $\boldsymbol{P}_{N\times ML}^{(k)}$ performs subcarrier modulation by shifting the samples into a vector of size N at the corresponding position of the kth subcarrier. For $K = L$, (5) is an alternative representation of (2). If the prototype filter is chosen such that its OOB leakage decays outside its subcarrier bandwidth L can become smaller than K. In case RRC filters are used, only adjacent subcarriers overlap. Typically $L = 2$ in this case and it becomes clear that L controls the modulators computational complexity.

From (5), it becomes apparent that M and K should both be a power of two in order to exploit the efficient Cooley-Tukey Fast Fourier Transform (FFT) algorithm. However, M is chosen such that it is an odd number because GFDM systems exhibit bad performance if both M and K are even numbers because Zero-Forcing (ZF) receivers do not exist (Matthe et al., 2014).

The next transmitter stage performs CP insertion. This is possible on a per frame basis because of the chosen cyclic shift in filter taps. Also, a CS may be inserted. Having only one CP and CS, of size N_{CP} and N_{CS} respectively, per frame improves spectral efficiency while still enabling simple one-tap FDE at the receiver.

Apart from CP insertion, frame windowing is performed in this stage in order to reduce OOB emissions (Michailow et al., 2014). Here frame windowing is applied to whole frames similar to (IEEE, 2012) where it is employed on a per-symbol basis. In accordance with (IEEE, 2012), usually a Raised-Cosine (RC) filter is applied. For the frame window N_{w} samples at the beginning and end of each frame are considered.

Before a GFDM frame is transmitted, a preamble is prepended as depicted in Fig. 1. This preamble is used for synchronization in the receiver but may also be used for an initial channel estimate.

2.2. Synchronization

A GFDM receiver must first locate a received frame $\boldsymbol{y}$ in the received sample stream before it can be demodulated. In (Demel et al., 2015), it has been shown that frame synchronization is a computationally expensive operation. Thus, we consider multiple synchronization stages in order to bound complexity.

The receiver processing chain, depicted in Fig. 2, is split into two stages. In the initial energy-based synchronization

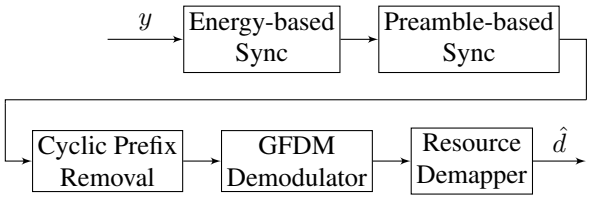

Figure 2. Receiver processing steps

stage, a coarse frame start is detected by detecting a rise in energy. The energy-based synchronization stage may be skipped if coarse synchronization is already achieved. Afterwards a high precision synchronization stage follows which also facilitates tracking. This preamble-based synchronization stage only searches in a window around the detected coarse frame start for the known preamble.

This fine synchronization is performed with an improved Schmidl&Cox algorithm (Awoyesila et al., 2008) which is adopted for GFDM (Gaspar et al., 2014). The synchronization algorithm requires the preamble to consist of two identical parts which are transmitted consecutively. In (Gaspar et al., 2014) a short GFDM frame with $M = 2$ timeslots is proposed with identical pseudo-random symbols in the first and second timeslot. The algorithm first performs a fixed-lag autocorrelation of length subcarriers K which yields a frame timing estimation with moderate accuracy. Furthermore, the fixed-lag autocorrelation is used to estimate a Carrier-Frequency-Offset (CFO).

In a window around this moderately accurate timing estimation a crosscorrelation with the received samples and the known preamble is performed. In this window, element-wise multiplication of the fixed-lag autocorrelation and crosscorrelation values results in a high precision timing synchronization. Eventually, synchronization yields a received frame vector including its CP and CS.

2.3. Receiver

A GFDM receiver must perform equalization and demodulate a received frame. Since GFDM is a non-orthogonal modulation scheme, Interference-Cancellation (IC) might be performed in order to remove self-interference.

Considering Fig. 2 after the synchronization stages, the modulated received frame $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}$ is obtained by removing the CP and CS with $\boldsymbol{H}$ being a cyclic block fading channel matrix and $\boldsymbol{n}$ being Additive White Gaussian Noise (AWGN). A ZF receiver with $\boldsymbol{G}_{\text{ZF}} = \boldsymbol{A}^{-1}$ would remove all self-interference, introduced by the non-orthogonal waveform, at the expense of noise-enhancement. In order to maximize Signal-to-Noise-Ratio (SNR) a Matched-Filter (MF) receiver with $\boldsymbol{G}_{\text{MF}} = \boldsymbol{A}^{H}$ is used but does not remove self-interference.

Unlike OFDM, GFDM enables to control self-interference by means of filter design. In case of RRC filters, only adjacent subcarriers need to be considered.

Similar to (5), demodulation is performed in frequency domain with $\boldsymbol{G}_{ML\times ML} = \boldsymbol{G}_{\text{MF}}$. First, the frame $\boldsymbol{y}$ is partially demodulated per subcarrier k

$$\boldsymbol{y}^0_{\text{RX},k} = (\boldsymbol{R}_{ML\times M})^T \boldsymbol{G}_{ML\times ML} \left(\boldsymbol{P}^{(k)}_{N\times ML}\right)^T \boldsymbol{H}^H_{\text{F}} \boldsymbol{\mathcal{F}}_N \boldsymbol{y} \tag{6}$$

where the diagonalized frequency domain channel matrix $\boldsymbol{H}_{\text{F}} = \boldsymbol{\mathcal{F}}\boldsymbol{H}$ is employed for one-tap channel equalization. $\boldsymbol{y}^0_{\text{RX},k}$ suffers from interference from adjacent subcarriers. This interference is combated with J IC iterations where $j \in \{0, \ldots, J-1\}$ (Gaspar et al., 2013). In each iteration j

$$\bar{\boldsymbol{d}}^j_k = q\{\boldsymbol{\mathcal{F}}^{-1}_M \boldsymbol{y}^j_{\text{RX},k}\} \tag{7}$$

is performed to obtain hard symbol decisions. Then, the interference to adjacent subcarriers

$$\boldsymbol{y}^j_{\text{I},k} = \boldsymbol{G}_{\text{I}} \boldsymbol{\mathcal{F}}_M (\bar{\boldsymbol{d}}^j_{(k+1) \mod K} + \bar{\boldsymbol{d}}^j_{(k-1) \mod K}) \tag{8}$$

is calculated where $\boldsymbol{G}_{\text{I}}$ accounts for the weighted interference derived from the used filters. Eventually, the next IC iteration $j+1$ is performed with updated receive vectors

$$\boldsymbol{y}^{j+1}_{\text{RX},k} = \boldsymbol{y}^0_{\text{RX},k} - \boldsymbol{y}^j_{\text{I},k}. \tag{9}$$

This process from (7) onwards is repeated J-times in order to minimize self-interference. After J iterations the demodulator returns interference-reduced soft decisions $\hat{\boldsymbol{d}}_k = \boldsymbol{\mathcal{F}}^{-1}_M \boldsymbol{y}^{J-1}_{\text{RX},k}$ for all subcarriers k.

3. Software Defined Radio

SDR is a term coined by Joseph Mitola (Mitola, 1992). The concept describes how formerly fixed hardware for signal processing is moved to the software domain. GNU Radio provides a powerful framework for SDR implementations (GNU Radio, 2017). It enables myriads of new applications and use-cases and offers unprecedented flexibility.

Research and development may be accelerated with software development techniques, e.g. rapid prototyping, introduced with the SDR concept (Otterbach et al., 2013). Field tests and simulations which share a common code base drastically improve technology verification and enable deeper investigation of the system of interest. Also, software development techniques help to improve code quality. Preferably, the switch from simulation to field test is performed by only switching out drivers as opposed to moving to a different implementation. A SDR implementation offers the advantage to combine these features and it can reveal feasibility in real world scenarios in regard to diverse requirements formulated by future 5G and I4.0 applications. Wireless Networks in-the-Loop (WiNeLo) is an

implementation that enables this easy switch from simulations to field tests.

Additionally, future Cloud RANs will benefit from a software implementation (Rost et al., 2015) which will enable more efficient use of available hardware. This could enable moving the signal processing away from the Radio Access Network (RAN) to centralized and virtualized computing resources. This also helps to capture challenges arising during research and development. Thus, development of new waveforms can be accelerated. Additional abstraction layers could also motivate borrowing techniques like heterogeneous computing from other industries with high computing requirements.

In this paper we focus on SDRs which are implemented in software and target GPP hardware but consider radio hardware constraints. Moreover, programming hardware logic is excluded from SDR because it contradicts with the target to virtualize signal processing for Cloud RAN.

Cooperative development of new waveforms is fostered with open-source tools, frameworks and waveforms. This is underlined for example in (Bloessl et al., 2013). While multiple implementations for WiFi exist, open-source implementations offer the possibility to investigate the proposed algorithms and collaboratively extend the basic system. This stands in contrast to proprietary software which requires extended efforts in terms of collaboration. Again, the presented approach to the mentioned WiFi implementation emphasizes the importance of a software development cycle approach to waveform development.

4. Implementation

In this Section we introduce the current implementation of *gr-gfdm*. The project was started in 2015 and heavily optimized since then. An analysis of expected implementation latencies for the underlying algorithms can be found in (Demel et al., 2017).

The implementation follows the division into logical segments of operation described in Sec. 2. Most of *gr-gfdm* is implemented in C++ for optimized code where function kernels separate optimized GFDM C++ functions from the GNU Radio interface as shown in Fig. 3. Apart from abstracting implementation details from the GNU Radio interface they enable reuse of the same implementation for other signal processing frameworks such as Python with NumPy and SciPy. Also, Python is used for implementation validation and initialization during start-up. All blocks in the transmitter and receiver can be parameterized with total number of subcarriers and timeslots. Most blocks are designed to be flexible and thus can be customized with more parameters depending on the blocks role.

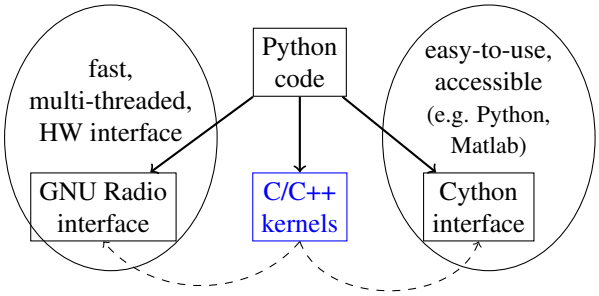

Figure 3. function kernel design concept

Using GNU Radio with the function kernels abstraction provides necessary flexibility to prototype and verify code in GNU Radio and Python in a timely manner. Additionally, the implementation is easily extendable with interfaces for other programming languages.

Licensing GNU Radio and Vector-Optimized Library of Kernels (VOLK) are freely available under the terms of the GPLv3 and Fastest Fourier Transform in The West (FFTW) is freely available under the terms of the GPLv2. The usage of libraries available under the terms of a GPL license has a huge advantage above propietary libraries. Performance impact of used algorithms can be examined and improved by modifying the source code. Reproducability of the implementation is ensured by checking used libraries for side effects.

4.1. Libraries and frameworks

This Subsection introduces the most important libraries and frameworks which are used in *gr-gfdm* (Rode, 2017).

GNU Radio A modular, multi-threaded framework for rapid-prototyping of SDR applications (GNU Radio, 2017). It offers a variety of basic tools for signal processing, visualization and infrastructure in order to develop new waveforms. A developer may focus on the actual algorithms at hand while GNU Radio provides a framework to deal with software design implications. This is particularly interesting when dealing with multi-threading because GNU Radio manages threads and data and allows writing thread-safe applications by default. GNU Radio provides a C++ API for developing performance critical parts of the algorithm. To speed up development prototyping GNU Radio also provides a Python API and the resulting flow graph representing the signal flow between signal processing blocks is generated using Python.

VOLK A library of math functions on vectors which are typically used in signal processing (VOLK, 2017).

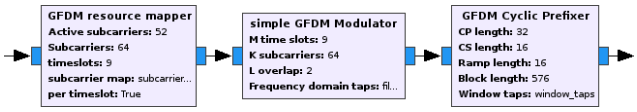

Figure 4. Transmitter in GNU Radio

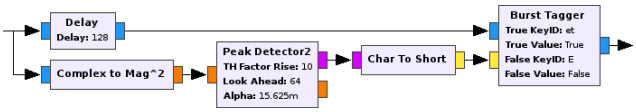

Figure 5. Energy-based synchronization

It makes use of Single-Instruction-Multiple-Data (SIMD) extensions which are present in many modern GPP hardware architectures such as Streaming SIMD Extensions (SSE), Advanced Vector Extensions (AVX) or NEON. It abstracts individual implementations for specific hardware and provides a canonical interface to all of them. VOLK enables developing a platform independent software which enables increased performance through usage of SIMD without being limited to a single hardware architecture.

FFTW One of the fastest known software implementations for Fourier transforms (Frigo & Johnson, 2005). It is a de facto standard for many software projects, both commercial and open-source.

4.2. Transmitter

This subsection introduces the transmitter signal processing blocks. It is assumed that data is provided in form of complex symbols. The GFDM transmitter first maps these symbols to a frame lattice, then performs GFDM modulation and finally adds a CP and or CS together with windowing. This is split into three steps illustrated in Fig. 4. Finally, a preamble is inserted before each modulated frame.

The Resource Mapper takes in blocks of MK_{on} samples and maps them onto a block with MK samples. The output vector contains the samples grouped subcarrier-wise for modulation. Unused subcarriers and unused lattice points are filled with $0 + j0$. Unused subcarriers can be determined by setting the number of active subcarriers and a subcarrier map during initialization of the block. The subcarrier map specifies the used subcarriers in each frame.

The Modulator expects vectors with MK samples which are then modulated. This block consumes the most transmitter resources and was optimized using the VOLK and FFTW libraries. The algorithms are implemented according to (5). To make sure the implementation works as expected tests compare the implemented algorithm with (2). These tests may be run after each compilation to ensure conformity. Customization of this block is allowed by specifying the overlap between subcarriers and the taps of the prototype subcarrier FIR filter in the frequency domain. With this implementation different subcarrier filters can be switched out very easily. This allows easy testing and simulation of the waveform.

The CP block adds a CP and CS and performs a windowing operation. It expects input vectors of size $N = MK$ and puts out vectors of size $N + N_{CP} + N_{CS}$. Afterwards frames are ready to be transmitted over-the-air. Parameterization of this block is allowed by specifying the CP and CS length. The window ramp length N_{ramp} and window taps can be specified as well.

4.3. Synchronization

Synchronization is typically a computationally heavy operation as shown in (Demel et al., 2015). We decided to match synchronization specifically to our system. Since we expect to work in a high SNR region, we start off with a coarse energy-detection-based synchronization stage. Fine synchronization is then performed with a simplified algorithm introduced in Section 2.2.

The energy-based synchronization stage is comprised of standard GNU Radio blocks as shown in Fig. 5. A rising edge over a certain threshold in the sample energy content is detected within a window. This synchronization stage has low complexity and provides a rough frame detection. The number of samples which are expected in a frame can be extracted and passed on to the next synchronization stage based on a detected peak.

The preamble is expected to be located within a window N_{pw} at the beginning of a frame. This narrows the search region for the preamble to a fixed window. In order to meet a high throughput, this synchronization stage is optimized with VOLK and FFTW. The fine synchronization stage emits a vector containing exactly one synchronized frame. Furthermore, CFO compensation as well as Automatic-Gain-Control (AGC) may be performed with only a small overhead.

4.4. Demodulation

GFDM is generally a non-orthogonal waveform and thus self-interference must be expected. For demodulation a simple demodulator without IC, ignoring self-interference, and an advanced demodulator with IC are implemented. The implemented receiver flowgraph is shown in Fig. 6. The demodulation algorithms are implemented according to (6) which uses the optimized operation in the frequency domain. Unit tests ensure the implementation works as expected.

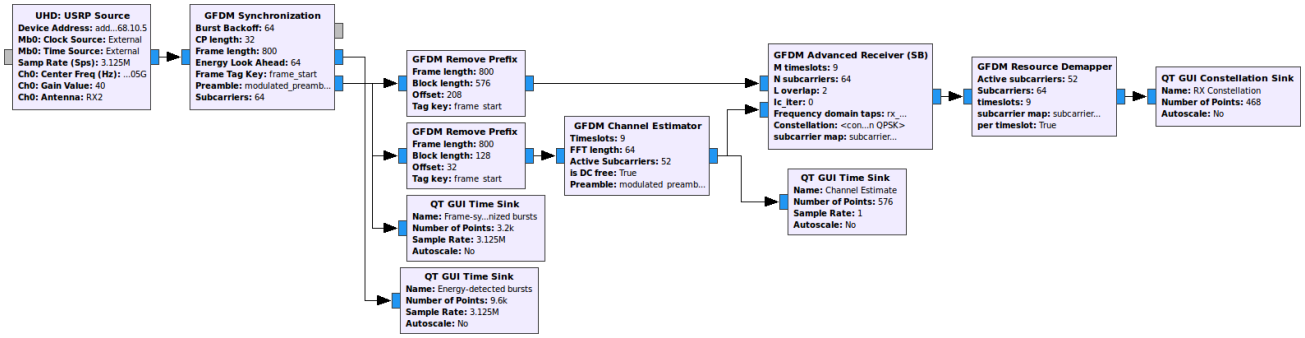

Figure 6. RX flowgraph in GNU Radio

The IC implementation is particularly necessary if higher order modulation is desired for the transmission and thus self-interference degrades performance more significantly. Using properties of the self interference, it is possible to significantly improve the quality of received symbols.

The demodulator may be provided with an initial channel estimate for equalization. This is especially necessary for field trials and simulations with channels other than AWGN, otherwise IC may degrade the performance of the system. A preamble based channel estimation algorithm is implemented in order to provide an initial channel estimate.

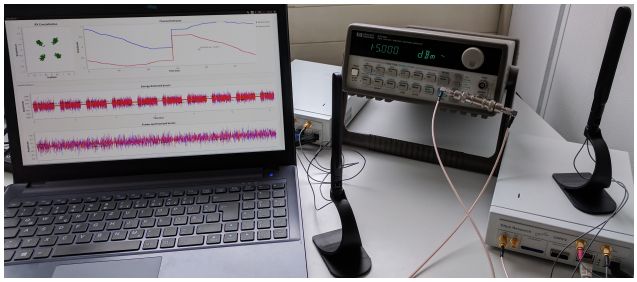

Figure 7. Hardware setup with Laptop, USRPs and signal generator

5. Field Trial

A major advantage of GNU Radio is its ability to serve as both, a simulation environment as well as a SDR. In this section we present the results of our efforts to use the implementation for over-the-air transmissions.

The hardware setup for this field trial is shown in Fig. 7 and summarized in Tab. 1. It consists of a laptop for transmit and receive signal processing in GNU Radio, USRPs as hardware frontends and a signal generator for frequency synchronization between the USRPs via their $10\,\mathrm{MHz}$ reference clock input. The GFDM system is parameterized as described in Tab. 2. The subcarrier filters have Root-Raised-Cosine shape, while the windowing function uses a

Software	Ubuntu 16.04 GNU Radio 3.7.11
Hardware	Intel Core i7-6700HQ with $16\,\mathrm{Gb}$ RAM
Frontend	2×USRP2 with RFX2400 daughterboards HP 33120A signal generator

Table 1. Software and hardware for field trial

Timeslots M	9
Subcarriers K	64
Active subcarriers K_{on}	52
Cyclic Prefix length N_{CP}	32
Cyclic Suffix length N_{CS}	16
Window ramp length N_{CP}	16
Number of IC iterations J	2

Table 2. GFDM system parameters

Raised-Cosine. Packets are transmitted every $1\,\mathrm{ms}$ with a sample rate of $3.125\,\mathrm{MS/s}$. This results in a packet length of $256\,\mu\mathrm{s}$ including a preamble. The corresponding receiver flowgraph is shown in Fig. 6.

The receiver chain is tapped during multiple stages in order to monitor the current state of the flowgraph with a GUI as shown in Fig. 8. A receiver symbol constellation shows the demodulated symbols, Quadrature Phase Shift Keying (QPSK) in this case. The channel estimate with its real and imaginary component is shown right next to it. Furthermore, the synchronization is facilitated in multiple stages with the corresponding output shown in the "Synchronization" part of the GUI. An energy-based burst detection is shown first and afterwards the time-synchronized bursts. The GNU Radio stream tag feature is used to indicate the start of a burst as detected by the receiver.

The presented field trial runs without underflows and overflows. Its resource requirements are measured with *htop* and indicate a load of approximately 90%, thus only one Central Processing Unit (CPU) core is occupied. Higher

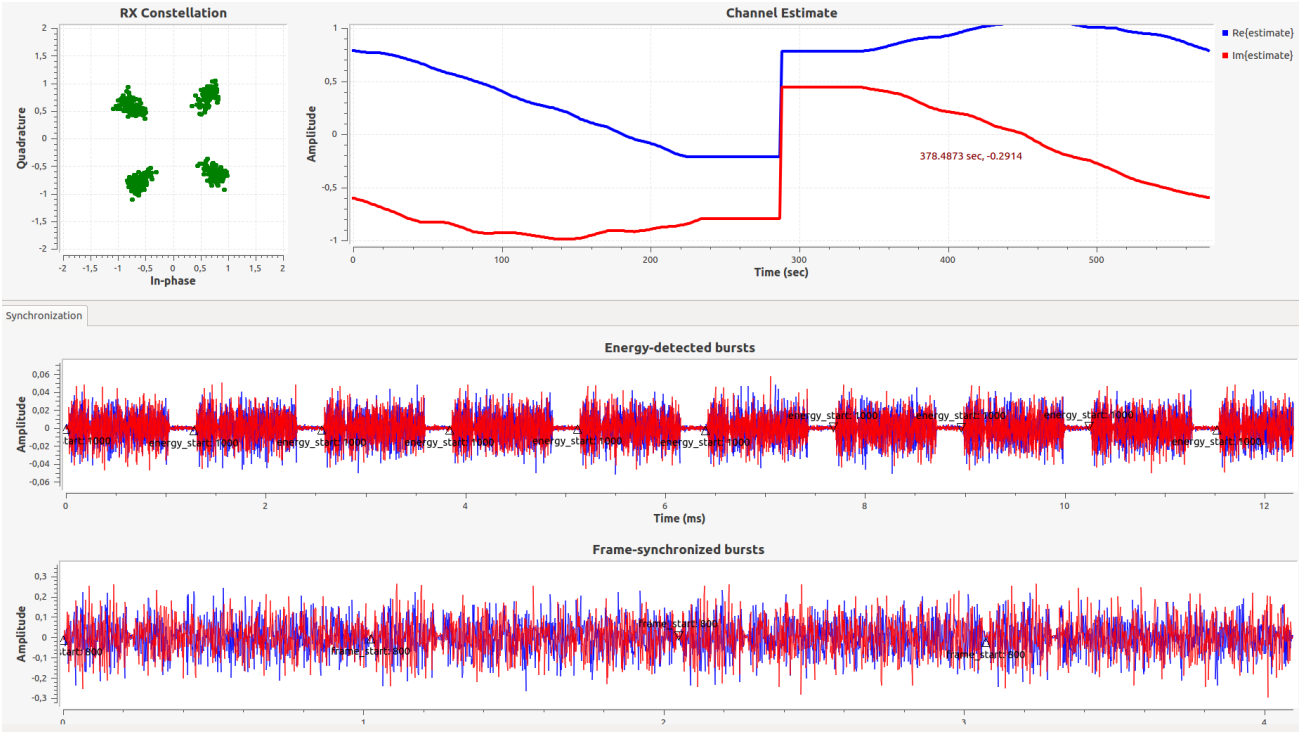

Figure 8. RX flowgraph GUI, showing a constellation, channel estimate and time samples for different synchronization stages

bandwidth may only affect the synchronization portion of the flowgraph. Thus, even at a sampling rate of 25 MS/s the load increases to approximately 250% while USRP source and sink consume more than 50% load.

6. Conclusion

In this paper we presented *gr-gfdm*, a GFDM implementation in GNU Radio. We made heavy use of GNU Radio's features in order to simplify the implementation efforts. It includes VOLK usage for massive speed-ups. This is, to the best of our knowledge, the first open source GFDM implementation in GNU Radio. The *gr-gfdm* can be used for simulations as well as field trials and its capabilities were demonstrated in a field trial. We showed the overall implementation is ready to be used for field trials and may run in real-time even at a sampling rate of 25 MS/s. We believe open source SDR implementations of new waveforms contribute to faster exploration of their capabilities. Thus they foster the adoption of new multicarrier waveforms because interested parties are free to use and contribute to the code.

Acknowledgment

This work was partly funded by the German ministry of education and research (BMBF) under grant 16KIS0263K (HiFlecs).

References

Awoyesila, Adegbenga B., Kasparis, Christos, and Evans, Barry G. Improved preamble-aided timing estimation for OFDM systems. *IEEE Communications Letters*, 2008. ISSN 10897798. doi: 10.1109/LCOMM.2008.081054.

Bloessl, Bastian, Segata, Michele, Sommer, Christoph, and Dressler, Falko. Towards an Open Source IEEE 802.11p stack: A full SDR-based transceiver in GNU Radio. In *IEEE Vehicular Networking Conference, VNC*, 2013. ISBN 9781479926879. doi: 10.1109/VNC.2013.6737601.

Danneberg, Martin, Michailow, Nicola, Gaspar, Ivan, Matthé, Maximilian, Zhang, Dan, Mendes, Luciano Leonel, and Fettweis, Gerhard. Implementation of a 2 by 2 MIMO-GFDM Transceiver for Robust 5G Networks. In *International Symposium on Wireless Communication Systems (ISWCS)*, Brussels, 2015.

Demel, Johannes, Koslowski, Sebastian, and Jondral, Friedrich K. A LTE receiver framework using GNU Radio. *Journal of Signal Processing Systems*, 78(3), 2015. ISSN 19398115 19398018. doi: 10.1007/s11265-014-0959-z.

Demel, Johannes, Bockelmann, Carsten, and Dekorsy, Armin. Evaluation of a Software Defined GFDM Implementation for Industry 4.0 Applications. In *Proceedings*

of the 2017 IEEE International Conference on Industrial Technology, Toronto, Canada, 2017.

Du, Jinfeng. *Pulse Shape Adaptation and Channel Estimation in Generalised Frequency Division Multiplexing Systems*. KTH, Stockholm, 2008. ISBN 978-91-7415-187-9.

Frigo, Matteo and Johnson, Steven G. The design and implementation of FFTW3. In *Proceedings of the IEEE*, 2005. ISBN 0018-9219. doi: 10.1109/JPROC.2004.840301.

Gaspar, Ivan, Michailow, Nicola, Navarro, Ainoa, Ohlmer, Eckhard, Krone, Stefan, and Fettweis, Gerhard. Low complexity GFDM receiver based on sparse frequency domain processing. In *IEEE Vehicular Technology Conference*, 2013. ISBN 9781467363372. doi: 10.1109/VTCSpring.2013.6692619.

Gaspar, Ivan S, Mendes, Luciano L, Michailow, Nicola, and Fettweis, Gerhard. A synchronization technique for generalized frequency division multiplexing. *EURASIP Journal on Advances in Signal Processing*, 2014. URL http://asp.eurasipjournals.com/content/2014/1/67.

GNU Radio. GNU Radio website. http://gnuradio.org/, 2017. Accessed: 2017-06-28.

IEEE. IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical report, IEEE, 2012. URL http://ieeexplore.ieee.org/servlet/opac?punumber=6178209.

Matthe, Maximilian, Mendes, Luciano Leonel, and Fettweis, Gerhard. Generalized frequency division multiplexing in a gabor transform setting. *IEEE Communications Letters*, 2014. ISSN 10897798. doi: 10.1109/LCOMM.2014.2332155.

Matthias Woltering, Dirk Wübben, Armin Dekorsy, Stephan Schedler, and Volker Kühn. Physical Layer Network Coding Using Gaussian Waveforms: A Link Level Performance Analysis. In *10th International ITG Conference on Systems, Communications and Coding (SCC 2015)*, Hamburg, 2015. URL http://www.scc2015.net/.

Michailow, Nicola, Krone, Stefan, Lentmaier, Michael, and Fettweis, Gerhard. Bit Error Rate Performance of Generalized Frequency Division Multiplexing. In *IEEE Vehicular Technology Conference (VTC Fall)*, Quebec City, 2012.

Michailow, Nicola, Matthe, Maximilian, Gaspar, Ivan Simoes, Caldevilla, Ainoa Navarro, Mendes, Luciano Leonel, Festag, Andreas, and Fettweis, Gerhard. Generalized frequency division multiplexing for 5th generation cellular networks. *IEEE Transactions on Communications*, 2014. ISSN 00906778. doi: 10.1109/TCOMM.2014.2345566.

Mitola, J. Software Radios Survey, Critical Evaluation and Future Directions. 1992. doi: 10.1109/NTC.1992.267870.

Otterbach, Nico, Braun, Martin, and Jondral, Friedrich K. Wireless Networks In-the-Loop: Creating an SDR Development Environment. In *IEEE International Symposium on Wireless Communication Systems (ISWCS)*, Ilmenau, 2013.

Rode, Andrej. gr-gfdm. https://github.com/kit-cel/gr-gfdm, 2017. Accessed: 2017-06-28.

Rost, Peter, Berberana, Ignacio, Maeder, Andreas, Paul, Henning, Suryaprakash, Vinay, Valenti, Matthew, Wübben, Dirk, Dekorsy, Armin, and Fettweis, Gerhard. Benefits and challenges of virtualization in 5G radio access networks. In *IEEE Communications Magazine*, 2015. doi: 10.1109/MCOM.2015.7355588.

Sahin, Alphan, Guvenc, Ismail, and Arslan, Huseyin. A survey on multicarrier communications: Prototype filters, lattice structures, and implementation aspects. *IEEE Communications Surveys and Tutorials*, 2014. ISSN 1553877X. doi: 10.1109/SURV.2013.121213.00263.

Schaich, Frank and Wild, Thorsten. Waveform contenders for 5G - OFDM vs. FBMC vs. UFMC. In *ISCCSP 2014 - 2014 6th International Symposium on Communications, Control and Signal Processing, Proceedings*, 2014. ISBN 9781479928903. doi: 10.1109/ISCCSP.2014.6877912.

Vakilian, Vida, Wild, Thorsten, Schaich, Frank, Ten Brink, Stephan, and Frigon, Jean Francois. Universal-filtered multi-carrier technique for wireless systems beyond LTE. In *2013 IEEE Globecom Workshops, GC Wkshps 2013*, 2013. ISBN 9781479928514. doi: 10.1109/GLOCOMW.2013.6824990.

VOLK. VOLK website. http://libvolk.org/, 2017. Accessed: 2017-06-28.