Frequency Domain Channelizer: A computationally efficient method for non-uniform channelization

Gereon A. Such GEREON.SUCH@UNIBW.DE
Matthias G. Schraml MATTHIAS.SCHRAML@UNIBW.DE
Andreas Knopp ANDREAS.KNOPP@UNIBW.DE
Bundeswehr University Munich, Chair of Signal Processing, Werner-Heisenberg-Weg 39, 85579 Neubiberg, Germany

Abstract

We present both, generalized theory and our implementation of a frequency domain channelizer suited for multicarrier Software Defined Radio receivers in terms of performance and applicability. Frequency domain channelizers fill the gap between multiple FFT/FIR-based channelizers and the polyphase channelizer considering dissimilar bandwidth carriers in a signal. Our open source implementation *gr-FDC* outperforms a combination of *Frequency Xlating FFT* filters at about ten channels and is introduced as an implementation example.

1. Introduction

The demand for services in digital radio communication rises in every part of the society. Prominent examples are multimedia streaming direct-to-home, the internet of things and connected cars for autonomous driving. Certain frequency bands are typically licensed to a certain type of communication, e.g. for television broadcasting (ITU, 2016).

However, the services often differs dramatically in the demand of bandwidth. Using a large bandwidth channel for a narrow-band service wastes the expensive frequency spectrum resource. The scarcity in the radio frequency (RF) spectrum leads to developments in dynamic spectrum allocation. Transmitting in currently unused frequencies can maximize the overall throughput of radio communication services. This is often done with Frequency Division Multiple Access (FDMA) schemes. Hence, the spectrum can be populated with many distinct carriers. These carriers are often non-uniform in their center frequencies and symbol rates. Applying a software defined radio (SDR) receiver with a high-bandwidth radio front-end at the base stations to such a scenario may be more advantageous than one radio front-end per carrier concerning flexibility or cost efficiency.

The extraction of many channels from a large bandwidth is a computationally expensive task. The polyphase channelizer is an efficient method to extract equally spaced carriers (Harris et al., 2003). For non-uniform carrier frequencies with arbitrary bandwidths, the current approach is to apply a rotator and a filter per carrier. For a few carriers this method is absolutely reasonable. For many carriers, however, the frequency domain channelizer (FDC) can be a very efficient option.

In this paper the theory of FDC is presented in Section 2 alongside of our GNU Radio implementation *gr-FDC* in Section 3. Moreover it is compared to a *Frequency Xlating FFT* filter per carrier in Section 4.

2. Frequency Domain Channelization

2.1. General Algorithm

An FDC generally transforms the complete received signal to the frequency domain (hence the name), extracts the corresponding frequency bins of the desired spectral part and transforms only this part back to the time domain. Since the time signal at the output should be continuous, several parameters must be considered as stated in Borgerding, 2006. The most important parameters are

$$\text{Block length} \quad N \in \mathbb{N}, \tag{1}$$

$$\text{Overlap length} \quad L \in \mathbb{N} \quad \text{with } L < N, \tag{2}$$

$$\text{Decimation factor} \quad d \in \mathbb{Q}, \tag{3}$$

where $\mathbb{N}$ are positive integer values and $\mathbb{Q}$ is any fraction of two positive integers. These parameters are well known from the frequency domain filter method called overlap-save (sometimes also called overlap-discard). Please refer to a signal processing textbook like Harris, 1987 for information about this method. Since the FDC is supposed to decimate the signal, $d \geq 1$ is an additional requirement. The decimated channel block length N_d and the decimated overlap length L_d must have an integer length, i.e.

Proceedings of the 8^{th} GNU Radio Conference, Copyright 2018 by the author(s).

$N_d = N/d \in \mathbb{N}$ and $L_d = L/d \in \mathbb{N}$. A more practical consideration of the parameters is given in section 2.3.

Fig. 1 is the graphical presentation of the general FDC algorithm. The first step of an FDC is to divide the vector of a digital complex baseband signal $\mathbf{r}$ with a sample rate f_s in consecutive blocks of length N. Each block starts with an overlap of L samples from the last block, while the first block is padded with L zeros. These blocks $\mathbf{r}_k$ are transformed to the frequency domain by an N-point Discrete-Fourier-Transformation (DFT) to

$$\mathbf{R}_k = \mathrm{DFT}\left(\mathbf{r}_k\right) \tag{4}$$

in the second step, where k is the index of subsequent blocks.

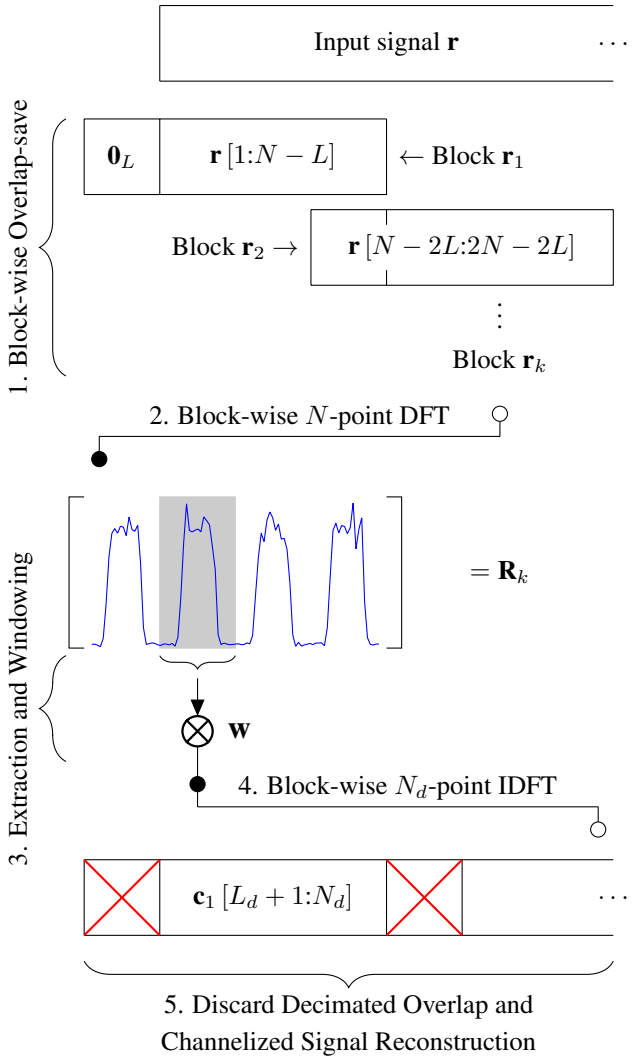

Figure 1. FDC structure.

The mixing and filtering operations and the decimation of the channelizer happens in the frequency domain in the third step. Mixing the signal with a negative frequency in the time domain is equivalent to the circular rotation of the DFT bins to the left. The rotation is given by a zero-based offset o, i.e. with $o = 0$ no rotation is performed. Hence, the baseband frequency is lowered by $f_o = of_s/N$. However, the possible center frequencies of the FDC are directly related to the DFT block length N, so a larger block length gives a finer granularity for the selection of the mixing frequencies.

The filtering of the FDC, like in the overlap-save method, can use a maximum of $L + 1$ lowpass finite impulse response filter coefficients. These coefficients are zero-padded and transformed into the frequency domain with an N-point DFT. Since we assume that the coefficients form a suitable antialiasing filter for the selected decimation factor, there are only N_d relevant bins in the DFT representation of the lowpass filter, which we call windowing vector $\mathbf{w}$.

Hence, instead of rotating all the bins and then use only the first N_d bins for further processing, index based selection of the bins of interest can be performed. So it is sufficient with good accuracy to extract a contiguous section of $\mathbf{R}_k$ with length N_d and discard the other bins. The extracted set of DFT bins is weighted with the N_d relevant bins of the windowing vector $\mathbf{w}$. However, if the used lowpass filter is not a good antialiasing filter, serious distortions can happen and the output of the FDC is different to the output of a channelizer based on a rotator and a filter working in the time domain. In the FDC, the window $\mathbf{w}$ is typically complex-valued which is explained in Section 2.2. The resulting output signal in the frequency domain is

$$\mathbf{C}_k = \mathbf{w} \cdot \mathbf{R}_k\left[o + 1 : o + N_d\right] \tag{5}$$

with $(\cdot)$ being the Hadamard product (element-wise multiplication).

In the last steps, the time domain channelized and decimated signal $\mathbf{c}$ is obtained by the N_d-point inverse DFT (IDFT) of all channelized blocks $\mathbf{C}_k$, discarding the overlap, i.e. removing the first L_d samples of each $\mathbf{c}_k$, and reconstructing the full vector by the concatenation of all blocks:

$$\mathbf{c}_k = \mathrm{IDFT}\left(\mathbf{C}_k\right)\left[L_d + 1 : N_d\right] \tag{6}$$

$$\mathbf{c} = \left[\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_{\frac{N-L}{d}}\right] \tag{7}$$

2.2. Subsequent Phase Shifts and its Compensation

Extracting a certain range of DFT bins in the third step in Fig. 1 is equivalent to baseband mixing. So the channelized

signal in the time domain corresponds to

$$\mathbf{c}(n) = \mathbf{r}(n) \exp\left(-j2\pi \frac{f_o n}{f_s}\right), \tag{8}$$

where $f_o = o f_s / N$ is the offset frequency selected by the offset o.

Due to the overlap-save procedure some samples are repeated in the blocks. However, the signal must be phase continuous even with these L samples which are repeated and discarded afterwards. Hence the phase at the beginning of the second block ($n = N - L + 1$) must be equal to the phase at the beginning of the first block ($n = 1$). The necessary requirement for the phase difference is given by

$$\exp\left(-j2\pi \frac{o(N-L)}{N}\right) = 1. \tag{9}$$

Therefore, the argument of the phase difference must be an integer multiple of 2π. With respect to the brevity of further equations we define the inverse relative overlap factor Ψ to be

$$\Psi = \frac{N}{L}. \tag{10}$$

The phase difference can be rewritten to

$$\exp\left(-j2\pi\left(o - \frac{o}{\Psi}\right)\right) = \exp\left(j2\pi \frac{o}{\Psi}\right), \tag{11}$$

since the offset o is an integer value. The requirement of (9) is fulfilled if the offset o is 0 or an integer multiple of the inverse relative overlap Ψ (Borgerding, 2006). Otherwise phase shifts occur on block transitions.

The first option to prevent the phase shifts is to adjust the offset o accordingly to be an integer multiple of $N/\mathrm{GCD}(N, L)$, where $\mathrm{GCD}(\cdot)$ is the greatest common divisor of both arguments. However, this can dramatically reduce the possible center frequencies the FDC can process, especially in cases of fractional values for Ψ or small block lengths N.

Another option is to compensate the block-wise phase difference by replacing the windowing vector $\mathbf{w}$ in the third step in Fig. 1 with

$$\mathbf{w}_k = \mathbf{w} \exp\left(-j2\pi \frac{o}{\Psi}(k-1)\right). \tag{12}$$

The windowing vector $\mathbf{w}_k$ can be implemented by a look-up table, since the phase values repeat after a certain number of frames. In general $N/\mathrm{GCD}(N, oL)$ vectors must be saved, where for the worst-case the offset is set to $o = 1$. In modern digital signal processing units or software defined radio environments, the additional memory requirement is most likely irrelevant but has to be considered for every application.

2.3. FDC Parameter Considerations

We further analyze the parameters from Section 2.1, as they crucially define the possible channel carrier frequencies and channel bandwidths, since these cannot be chosen continuously for an FDC but are limited by the overlap-save method and possible software or algorithmic limitations for DFT block lengths.

In every case, N, N_d, L and L_d need to be integers, since half samples do not exist and an interpolation is not considered for performance reasons. Hence the maximum decimation factor d_{max} is the $\mathrm{GCD}(N, L)$. Therefore, all possible decimation factors are given by

$$d = \frac{d_{max}}{m}, \tag{13}$$

where $m \in \mathbb{N}$ and $m \leq d_{max}$. All possible bandwidths of the channelized signals are $f_{s,d} = f_s / d$.

To assure a comfortable set of possible decimation factors matching to several bandwidths alongside an easy computation of the (I)DFTs, we can limit some parameters to certain values. If we choose N and L to be power-of-2 values and the possible decimation rates to all power-of-2 values up to L, the maximum of Ψ different windowing vector sets have to be saved to prevent phase shifts on block transitions. This parameter limitation is considerable for the Radix-2 Fast-Fourier-Transform (FFT) algorithm and its inverse to increase the performance. Even though the common FFTW3 library (Frigo & Johnson, 2003) allows non-power-of-2 transformation lengths on general purpose processors, performance gains can be acchieved by power-of-2 block lengths. Our FDC implementation takes advantage of this performance gain and restricts block lengths and bandwidths to power-of-2 values.

3. Presenting gr-FDC

Our multi-purpose implementation of an FDC is the GNU Radio *gr-FDC* module developed and shared on GitHub (Such, 2018). It contains three core signal processing blocks utilized by a wrapping Python hierarchical block, a custom waterfall block to visualize extracted channels and additional auxiliary blocks. The three core blocks represent different operational modes to operate on a frequency domain vector, extract and process a specific bandwidth and reconstruct the channelized time signal.

- The first operational mode is the *Throughput* mode. The output is a constant stream of complex samples representing the time signal of a channel with constant carrier frequency and constant bandwidth. The *Throughput* mode is meant to operate on constant stream channels.

- The second block is the *FixedCarrier* mode, which also operates on a fixed carrier frequency and bandwidth, but it detects power differences in the channel, buffers and finally emits signal via message passing as PDU or stores each buffered signal in a separate file. The *FixedCarrier* mode is designed to operate on Time Division Multiple Access (TDMA) channels. It is possible to periodically emit the buffered channel signal after a certain amount of blocks processed, to be able to handle channels with long activation time, i.e. speech in close to realtime.

- The third block called *Detection* is not defined by a carrier frequency and a fixed bandwidth but a start and stop frequency, which define a segment. In this segment, carriers are determined by rising and falling edges of the power spectral density and a temporary channel is tailored to it adjusting carrier frequency and bandwidth according to detected parameters. The extracted channels are as well as in *FixedCarrier* mode emitted as a PDU via message passing or stored in separate files. Like in *FixedCarrier* mode, the preemptive emission of the channel buffer is possible.

  To reduce the necessary numerical complexity and avoid noise influence on determination of the carriers, especially if a carrier is still active, the number of samples for the power spectral density computation is reduced by averaging. The number of samples determined by the parameter *Minimum Channel Distance* specified by the developer. On the other hand, this averaging factor reduces the resolution to determine the channel's start and stop frequencies. This is why in Fig. 2 the *Detection* channels suffer from an offset. Nevertheless, a bandwidth puffer can be chosen to securely acquire all channels.

- The hierarchical block *Frequency Domain Channelizer* is a wrapping convenience block which takes care of parameter checks and connections. We strongly recommend to use this block instead of connecting the blocks yourself to avoid false parametrization. It is possible to input a stream of complex or real samples representing the time signal, or, if you plan to implement a high performance system, input the overlapped and Fourier transformed vector to be able to implement the Fourier transformation on another machine or hardware.

- The visualizing block *WaterfallMsgTagging* is a PyQt block displaying a waterfall diagramm and visualizing the emitted channels based on the PDU meta data. It is tailored to gr-FDC in terms of displaying and not discarding any blocks, which is why we did not derive it from QT GUI Waterfall Sink. The Throughput channels are not visualized, since this mode does not emit any PDUs. Fig. 2 illustrates the graphical output. Currently, only the frames on detected channels are drawn, no further meta information is provided.

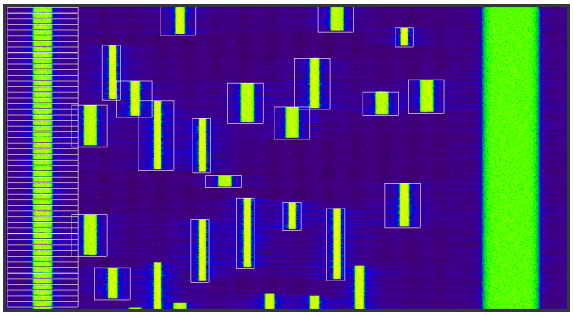

Figure 2. gr-FDC WaterfallMsgTagging Output

The wrapping channelizer and the waterfall block are written in Python and the signal processing blocks are written in C++ utilizing C++ STL in terms of data structures, primarily `std::vector` and `std::deque`, and in block threading in terms of `std::thread`. For numerical operations, both the Vector Optimized Library of Kernels (FSF, 2015) and FFTW3 (Frigo & Johnson, 2003) via GNU Radio wrapper are used making *gr-FDC* efficient for live systems.

All channelizing blocks operate on vector lengths of power-of-2. The main reasons for this are performance evaluation of the FFT and IFFT, avoidance of errors by design and algorithmic convenience to find a matching decimation factor representing an integer extraction range according to a floating point bandwidth. The result is a slight carrier frequency deviation from the desired frequency and the bandwidth according to the used integer decimation factor. Actual carrier frequency and bandwidth information are provided in the meta data of each PDU or are printed on initialization by the *Throughput* channels. For Throughput channels, this is not a satisfying solution but the only reasonable one we see right now, since the actual extraction bandwidth and carrier frequency is determined on initialization. You can use these parameters to adjust subsequent blocks if they depend on the actual sample rate.

Since the carrier frequency and carrier phase usually have to be synchronized in subsequent steps after the channelization, these deviations might be compensated in your application anyways, but to be fair compared to *Frequency Xlating FFT* blocks, a rotator might be applied on the channelized data. If the parameters are set properly, the upper bound for carrier frequency deviation f_Δ is the equivalent of one sample of the frequency domain vector of block length N, and thus $f_\Delta \leq f_s/N$ with sample rate f_s.

Currently, all applied windows in the frequency domain are lowpass filters, and a matched filter has to be applied

on the channelized signal. As stated in Section 5, for the *Throughput* and *FixedCarrier* modes, this is planned but not implemented yet. Furthermore, the window on the time signal before transforming the block-wise signal is rectangular to assure a reversible Fourier Transformation, which leads to a leakage effect visible in the Waterfall diagram. Currently, we accept this since we do not want to sacrifice performance, but consider several solutions to resolve this, for instance by performing a convolution of a minimal Blackman-Harris window in the frequency domain.

4. Performance Comparison

Compared to a parallel *Frequency Xlating FFT Filter* combination, *gr-FDC* yields a significant performance gain, where performance means a reduction of computational complexity without a loss of signal quality.

4.1. Computational Efficiency

For evaluation of the computational performance of *gr-FDC* compared to a *Frequency Xlating FFT Filter* channelizing combination, we used a sample file with sample rate $f_s = 8$ MHz and a duration of $T = 40$ s, resulting in a length of $3.2 \cdot 10^8$ samples. This file was channelized with a parallel filter combination of K *Frequency Xlating FFT Filters*, each with a unique center frequency but always with a lowpass of bandwidth $B = 0.05$ including a transition width of 0.01 relative to the sample rate and decimation factor of $d = 40$. The FDC operates on the same carrier frequencies with the same bandwidth but with a block length of $N = 4096$ samples and a relative inverse overlap of $\Psi = 4$. Internally, the decimation factor is lowered to $d = 32$ due to *gr-FDC* design limitations giving *gr-FDC* a slight penalty in terms of *samples-to-be-processed*. The simulations where performed throttle-less and one after another measuring the necessary execution time. All processing times below 40 s (the duration of the sample file) indicate a realtime channelization capability. Nonetheless, considering signal processing of each channel, a sufficient performance buffer should always be considered.

The simulation was performed on a desktop system with an Intel Core i5-2500K Quad Core processor, with 16 GB RAM and Ubuntu 18.04 operating system with GNU Radio 3.7.12.0 built from GitHub source. Fig. 3 shows the results of the simulation. In good approximation the processing time is linear dependent of the number of channelizers with a break-even point at around 10 channelizers, where the *gr-FDC* outperforms the *Frequency Xlating FFT* channelizers. With the used hardware, the maximum realtime capability for an 8 Mega-Samples per second signal would be in case of a *Frequency Xlating FFT* filter combination 60 channelizers and for the *gr-FDC* 120 channelizers.

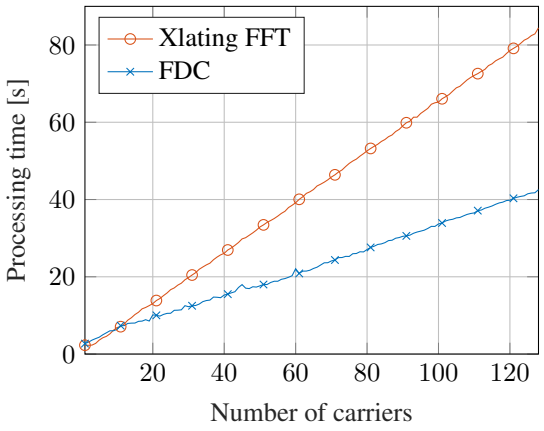

Figure 3. gr-FDC and multiple *Frequency Xlating FFT* performance comparison per carrier

Keep in mind that this is one specific case and for different parameters, i.e. bandwidth, block length, relative inverse overlap and necessary post-channelization frequency correction, this gain might be reduced. Furthermore, the maximum realtime capability in terms of used channelizers is rather a criteria for the computational capability than the FDC performance.

4.2. Signal Quality Validation

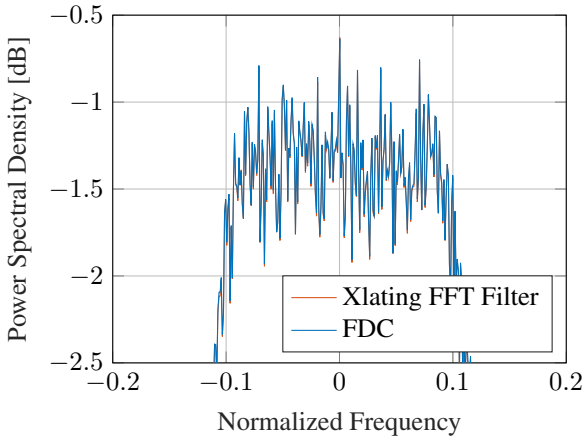

Figure 4. Power Spectral Density of *gr-FDC* and *Frequency Xlating FFT Filter*

The blockwise fast convolution with integrated decimation in the frequency domain as proposed in Section 2 and implemented by *gr-FDC* is not equal to a straight forward convolution of the whole signal with an equivalent FIR filter of the used window. This is due to multiplication of the window in the frequency domain representing a circular convolution on each block in the time domain, which is equal to interference of samples from the beginning to the

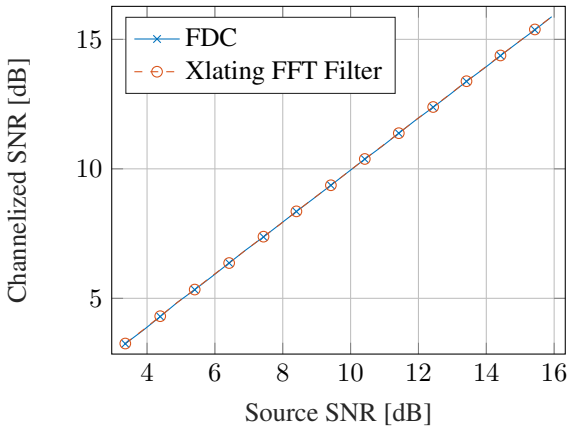

Figure 5. Signal-to-noise ratio comparison of *gr-FDC* and *Frequency Xlating FFT Filter*

end of the block in the time domain and vice versa. This differs from the straight forward convolution in the time domain as implemented by an FIR filter. Though this effect should be canceled by the appropriate choice of an overlap, the quality of the resulting signal has to be verified against conventional channelizing methods.

To confirm the similarity of *gr-FDC* and *Frequency Xlating FFT Filter* channelization concerning the signal-to-noise ratio (SNR), a single QPSK carrier is channelized, synchronized and the SNR is measured and compared.

For Figure 4, the QPSK carrier is channelized with an output rate of 4 samples per symbol. Also the complete bandwidth of the carrier is within the passband of the lowpass filter. Since the magnitudes of the power spectral densities of the common *Frequency Xlating FFT Filter* and our *gr-FDC* are identical, the expected behavior of distortion-free channelization of the *gr-FDC* is confirmed.

Figure 5 illustrates the resulting SNRs for channelization with *gr-FDC* in *Throughput*-mode with block length of $N = 2048$ and a *Frequency Xlating FFT Filter*, where each SNR is plotted against the theoretical SNR. Obviously, the measured SNRs do not differ besides numerical variations, which further indicates the low-loss properties of *gr-FDC*.

5. Conclusion and Future Development

Frequency domain channelizing and our GNU Radio implementation *gr-FDC* offer a performant channelization method for different use cases in live software defined radio systems for broadband radio monitoring, Demand Assigned or Frequency Division Multiple Access system carrier extraction. Main advantages include the variable channel bandwidths and frequencies when multiple carriers are channelized and its performance boost, since the forward DFT only needs to be performed once independently of the numbers of carriers. The main disadvantages are the limitation of the extraction bandwidth and carrier frequency to integer steps and cannot be chosen continuously.

Future planned and considered features for *gr-FDC* include:

- Root Raised Cosine Matched Filtering option instead of lowpass windowing for *FixedCarrier* or *Throughput* channels
- Further GUI interaction and meta data visualization from the waterfall display to monitor and control the channelizer
- Code clean-up
- Further documentation besides GRC-doc string

We encourage you to implement the FDC concept for a channelizer customized to your needs, for example a channel activation controlled by a systems broadcast and signaling channel, which we do not plan to implement in *gr-FDC*.

If *gr-FDC* suffices to your application, feel free to use, contribute, fork or customize it accordingly. *gr-FDC* is licensed under GPL v3 and hosted on GitHub (Such, 2018).

References

Borgerding, Mark. Turning Overlap-Save into a Multiband Mixing, Downsampling Filter Bank. *IEEE Signal Processing Magazine*, 2006.

Frigo, Matteo and Johnson, Steven. *FFTW3*, 2003. URL http://www.fftw.org.

FSF. *Vector Optimized Library of Kernels (Volk)*. Free Software Foundation, 2015. URL http://libvolk.org.

Harris, F. J., Dick, C., and Rice, M. Digital receivers and transmitters using polyphase filter banks for wireless communications. *IEEE Transactions on Microwave Theory and Techniques*, 51(4):1395–1412, April 2003.

Harris, Frederic J. Chapter 8 - Time Domain Signal Processing with the DFT. In *Handbook of Digital Signal Processing*, pp. 633 – 699. Academic Press, San Diego, 1987. ISBN 978-0-08-050780-4.

ITU. *Radio Regulations*. International Telecommunications Union, 2016. URL http://www.itu.int/pub/R-REG-RR-2016.

Such, Gereon. *gr-FDC*, 2018. URL https://github.com/gereonsuch/gr-FDC.