Evaluation of a Software Defined GFDM Implementation for Industry 4.0 Applications

Johannes Demel, Carsten Bockelmann, Armin Dekorsy
Department of Communications Engineering
University of Bremen
Bremen, Germany
Email: {demel,bockelmann,dekorsy}@ant.uni-bremen.de

***Abstract*—Industry 4.0 (I4.0) closed-loop control applications require ultra low-latency and ultra high reliability. Generalized Frequency Division Multiplexing (GFDM) promises to meet these requirements in conjunction with low Out-Of-Band (OOB) emissions. We present latency measurements of an open-source, modular, portable GFDM software implementation using the GNU Radio framework. The used implementation and its components are investigated regarding their theoretical function. We investigate the usability of the GFDM implementation for Software-Defined Radio (SDR). This is facilitated by benchmarking the various functions with respect to different parameter sets. Thus, we focus on latency measurements which provide reliable figures for SDR system design. Furthermore, we show that low latency broadband SDR systems for I4.0 applications are feasible.**

I. Introduction

Current 4th Generation (4G) and WiFi mobile radio technologies are designed for high data rate human-centered applications. These applications include video streaming and large file transfers. As a consequence of this design target 4G systems focus on large packets. Furthermore, moderate latency requirements are sufficient and occasional packet losses are expected which is traded in for higher overall throughput. Next generation communication systems are expected to enable new applications which exhibit Machine-type-Communication (MTC) behavior. In the realm of I4.0, closed-loop control applications are envisioned which require ultra low-latencies and ultra high reliability. Moreover, a vast range of applications require that the waveform is flexible and can be adopted to different use-cases.

A. Motivation

New I4.0 applications are envisioned to maximize flexibility and create cooperative distributed production lines. In contrast to current 4G and WiFi communication systems I4.0 applications impose differing requirements on a I4.0 communication system [1]. Current systems are human-centered whereas MTC in I4.0 systems is expected to enable closed-loop control applications. Especially, industrial automation closed-loop control applications require ultra high reliability and ultra low-latency for short packet communication systems. In these systems communication is deterministically organized in cycles which are expected to have a duration of less than $1\,\mathrm{ms}$. In this timeframe all clients must send and receive their packets [1]. Additionally, these systems require reliable and deterministic real-time data delivery. Exceeding the overall latency budget leads to data loss and may bring an entire production line to a halt. Here we focus on physical layer processing latencies.

Current 4G systems rely on Orthogonal Frequency Division Multiplexing (OFDM) which is a simple and effective Multi-Carrier System (MCS). However, several shortcomings with regards to OFDM were identified [2]. These shortcomings include poor OOB emission properties, strict synchronization requirements and poor spectral efficiency. I4.0 communication systems are expected to operate in harsh, densely populated environments such as production floors with multiple communication systems in close proximity. Thus, I4.0 systems must coexist with each other and with legacy systems. Moreover, these systems must use the spectrum more efficiently by reducing their OOB emissions in order to enable them to meet the I4.0 requirements.

Several MCS candidate waveforms for I4.0 exist which offer different approaches to overcome OFDM shortcomings [3]. Filter-Bank Multi-Carrier (FBMC) minimizes OOB emissions by filtering each subcarrier but introduces large filter delays [2]. 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 [4].

GFDM goes beyond symbol-based modulation by modulating entire frames [5]. 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. For I4.0 low-latency communication systems short filter delays are an important advantage because latency reduction is a critical design criteria which GFDM can deliver.

In addition to over-the-air latencies, signal processing adds significant delays to a communication system. The introduced latencies need to be known in order to be able to design a reliable low-latency I4.0 system. In [6] a GFDM implementation in LabView targeting the Universal Software Radio Peripheral (USRP) X310 is presented but its latency properties are unknown. There are no other latency measurements of GFDM systems known to the authors. All relevant parts of the signal processing chain are implemented for this specific hardware. I4.0 applications represent a diverse set of use cases.

This requires more portability and flexibility in order to adapt the communication system to these use-cases.

SDR is a term coined by Joseph Mitola [18]. The concept describes how formerly fixed hardware for signal processing is moved to the software domain. It enables myriads of new applications and use-cases and offers unprecedented flexibility. Furthermore, research and development may be accelerated with software development techniques, e.g. rapid prototyping, introduced with the SDR concept [7]. Field tests and simulations which share a common code base drastically improve technology verification. A SDR implementation offers the advantage to combine these features and it can reveal first figures for latencies in a GFDM system for I4.0. Additionally, future Cloud Radio Access Networks (Cloud RANs) will benefit from a software implementation [8] which will enable more efficient use of available hardware.

B. Main Contribution

The main contribution of this paper is an investigation of the latencies introduced by the various processing steps of a GFDM SDR software implementation, including transmission, reception and synchronization. For these applications we run a set of benchmarks which outline the practicability of the implementation in the context of I4.0 low-latency applications. We present benchmark results for all functions introduced in theory and point out how different parameters impact latency. The results show that the investigated SDR implementation is suitable for low-latency I4.0 systems.

II. 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 different individual use-cases. Also, latencies may be minimized by designing a system accordingly.

A. 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 [3]. 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 [9]. Often, some subcarriers are not used for transmission but only $K_{\text{on}} \leq K$ are used.

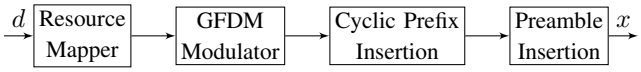

Fig. 1. Transmitter processing steps

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 [5]. 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 [10], [11]. 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 [12]. Therefore, (2) can be rewritten to

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

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 [13].

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 [5]. Here frame windowing is applied to whole frames similar to [14] where it is employed on a per-symbol basis. In accordance with [14], 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.

B. Synchronization

A GFDM receiver must first locate a received frame $\boldsymbol{y}$ in the received sample stream before it can be demodulated. In [15], 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 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.

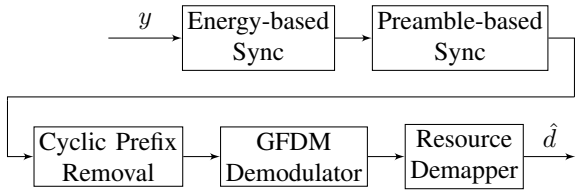

Fig. 2. Receiver processing steps

This fine synchronization is performed with an improved Schmidl&Cox algorithm [16] which is adopted for GFDM [17]. The synchronization algorithm requires the preamble to consist of two identical parts which are transmitted consecutively. In [17] 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.

C. 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

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

where the diagonalized frequency domain channel matrix $\boldsymbol{H}_{\mathrm{F}} = \boldsymbol{\mathcal{F}}\boldsymbol{H}$ is employed for one-tap channel equalization. $\boldsymbol{y}^0_{\mathrm{RX},k}$ suffers from interference from adjacent subcarriers. This interference is combated with J IC iterations where $j \in \{0, \ldots, J-1\}$ [12]. In each iteration j

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

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

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

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

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

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 $\hat{\boldsymbol{d}}_k = \boldsymbol{\mathcal{F}}^{-1}_M \boldsymbol{y}^{J-1}_{\mathrm{RX},k}$ for all subcarriers k.

III. Implementation

In this Section we introduce the implementation used for our benchmarks. The SDR implementation *gr-gfdm* is freely available under the terms of the GPLv3 [19]. Details of the implementation may be explored in source code at [19]. Here, only the foundations are layed out.

The implementation *gr-gfdm* is originally targeted to be a GNU Radio Out-Of-Tree (OOT) module which was established by Andrej Rode in 2015. We introduced function kernels, which represent the operations presented in Sec. II, in order to separate optimized GFDM C++ functions from the GNU Radio interface. These kernels, which are benchmarked in Sec. IV, represent the different stages for transmitting and receiving GFDM signals.

A. Libraries and frameworks

This Subsection introduces the most important libraries and frameworks which are used in *gr-gfdm*.

a) GNU Radio: A modular, multi-threaded framework for SDR applications [20]. It offers a lot of standard capabilities 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 deals with all the software design implications. This is particularly interesting when dealing with multi-threading because GNU Radio manages threads and data. It is freely available under the terms of the GPLv3.

b) Vector-Optimized Library of Kernels (VOLK): A library of math functions which are typically used in signal processing [21]. It makes use of Single-Instruction-Multiple-Data (SIMD) extensions which are present in many modern General Purpose Processor (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.

c) Fastest Fourier Transform in The West (FFTW): One of the fastest known software implementations for Fourier transforms [22]. It is a de facto standard for many software projects, both commercial and open-source.

IV. Benchmarks

This section explores how the system under test performs under various parameters. The primary target is to measure processing latencies for the different stages in the processing chain.

The idea of the simulations is to identify the cost of individual operations for GFDM transmitters and receivers as depicted in Fig. 1 and 2. The benchmarked kernels present a minimum latency cost for the investigated GFDM transceiver system. Additional latencies due to multi-threading, peripheral access and such are not considered. The simulation benchmarks help identify suitable parameter sets for the task at hand. Also, they help at identifying possible targets for future optimizations.

A. System setup

We want to motivate the chosen parameter set by outlining system requirements for I4.0 closed-loop control systems which exhibit a deterministic cyclic transmission pattern. All clients exchange packets at regular intervals. These intervals are typically below $1\,\mathrm{ms}$ which motivates the low latency requirement.

The expected packet sizes vary approximately in the range $48\,\mathrm{bit}$ to $1024\,\mathrm{bit}$ [1]. Thus, individual packets are very short in terms of over-the-air transmission time. In order to use the spectrum resource efficiently, a Time-Division-Duplex (TDD) is the duplex scheme of choice.

The individual benchmarks are facilitated as follows: First, initialization of the kernels is performed. Second, each kernels operation is performed on random data multiple times in order to flush out any initialization related effects. Third, each kernels operation is performed 10000 times in a single thread with random data. We perform latency benchmarks without channel influences such as fading or noise. The reliability of GFDM highly depends on the channel model and other physical layer functions such as modulation and channel coding but these are beyond the scope of this analysis.

Each benchmark run is measured with Boost.Chrono [23]. The hardware used for the benchmarks is an Intel Core i7-4790 with $16\,\mathrm{Gb}$ RAM running Kubuntu 14.04 with Linux kernel 3.13. The software used for the benchmarks is GCC 4.8.4, VOLK 1.30 and FFTW 3.3.3.

B. Results

Fig. 3 shows the latency over block size $N = MK$ results for the resource mapper and demapper where K is fixed and M is varied in a discrete range. As the number of symbols per block increases, so does latency. With fewer active subcarriers, latency increases more slowly. Mapper and demapper latencies are negligible in comparison to other kernels.

At the heart of the transmitter chain is the GFDM modulator. As can be observed in Fig. 4, it causes most of the processing

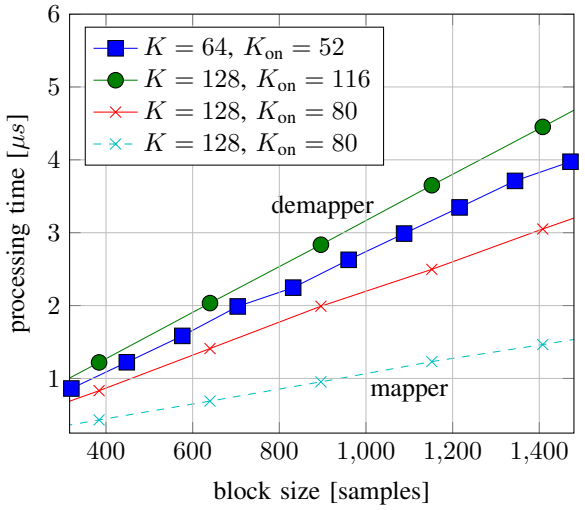

Fig. 3. Latency measurement for resource mapping [dashed] and demapping [solid] with different numbers of subcarriers K and active subcarriers K_{on}.

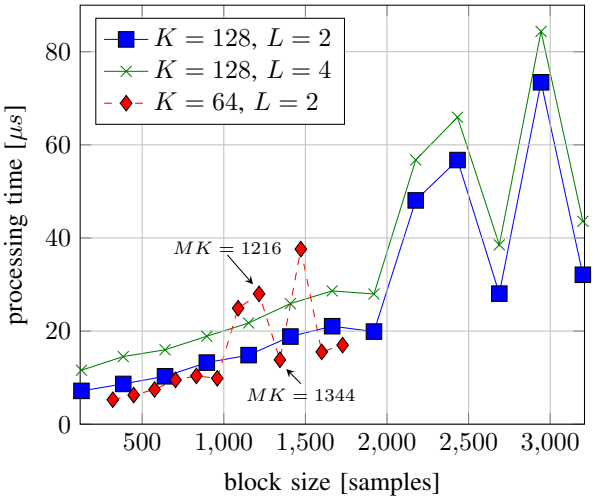

Fig. 4. Latency measurement for GFDM modulation with increasing block size.

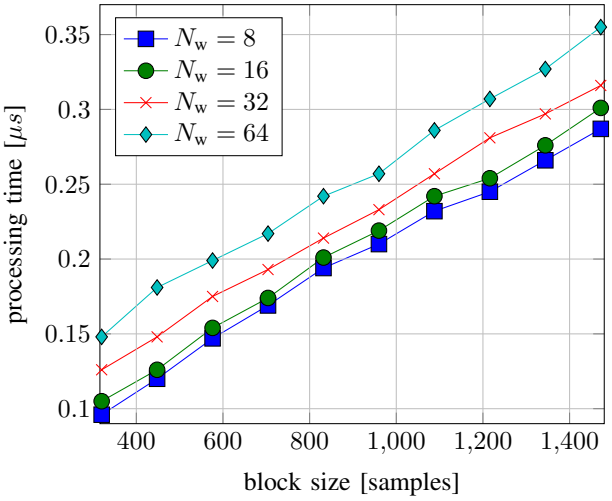

Fig. 5. Latency measurement for CP addition and windowing with different window roll-off sizes.

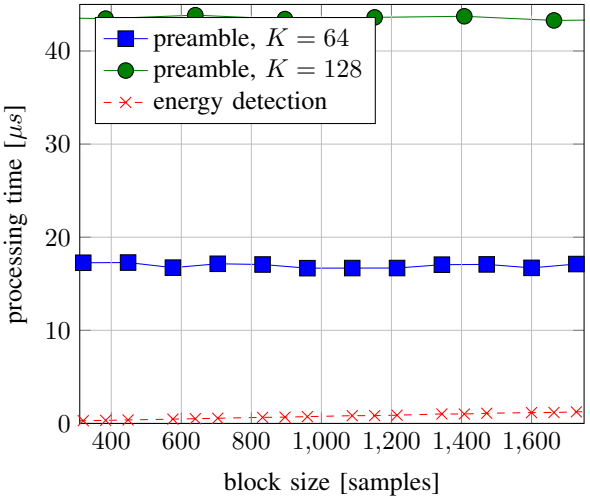

Fig. 6. Latency measurement for synchronization stages with different number of subcarriers K.

latency spent at the transmitter. In contrast to the other processing steps, it does not exhibit a linear behavior with regards to block size. The employed Fourier transforms vary in size and are not a power of 2. Though, the FFTW authors point out that the algorithms work best for powers of 2. Furthermore, small prime factors for the FFT-size improve latency. This may be exemplified by considering a FFT-size of $K = 64$ and comparing the benchmark results for $MK = 1216$ and $MK = 1344$. Though the block size is increased, for $MK = 1344$ the latency is cut in half. This is explained by looking at the prime factors, being $2, 3, 7$ for $MK = 1344$ and $2, 19$ for $MK = 1216$.

The results for combined CP addition and frame windowing are shown in Fig. 5. The benchmark reveals two major sources for latencies. First, the copy operation dominates latency. Second, a larger transition window at the beginning and the end of a block increases latency.

Synchronization is a two stage process. First, the presence of a frame is detected by means of an energy detector. In Fig. 6 energy detection is performed on vectors of block size N. Second, a fine synchronization is performed with the help of a preamble detector within a certain frame. The search window for preamble detection is fixed to $N_{\text{CP}} + 3K + 160$ samples with $N_{\text{CP}} = \frac{K}{2}$. The resulting processing times, shown in Fig. 6, outline that energy detection does not introduce significant latency and preamble detection introduces constant latency with respect to block size because the system must only search for the preamble within a window.

The receiver is available in two different flavors. A simple receiver demodulates GFDM frames by performing the inverse operations outlined for the modulator. However, GFDM is

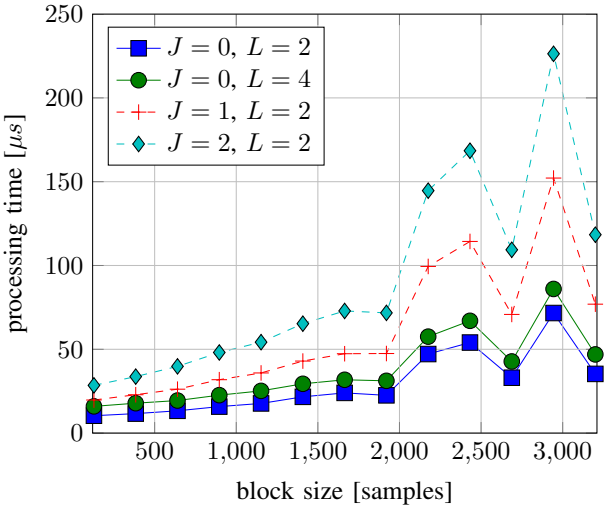

Fig. 7. Latency measurement for GFDM demodulation with increasing block size and different number of IC iterations with $K = 128, K_{\text{on}} = 80$

not an orthogonal modulation scheme and thus introduces self-interference. This interference is targeted in the advanced receiver which performs IC where more iterations generally improve the Symbol-Error-Rate (SER) performance. With $J = 2$ iterations, all interference is canceled out if a noise free transmission is considered. Fig. 7 shows the resulting latency benchmarks. IC iterations dominate latency increase. Again, the receiver exhibits the same behavior as the modulator with respect to FFT-size.

We consider one example to outline the overall one-way latency with a specific parameter set with $K = 128, K_{\text{on}} = 80, M = 21, N_{\text{CP}}/N_{\text{CS}} = 64$ and $20\,\text{MSps}$ sample rate, the resulting frame duration is $140.8\,\mu\text{s}$. The corresponding processing latency with $L = 2$ and $J = 2$ IC iterations sums up to $187\,\mu\text{s}$. Thus, the overall worst case system latency is expected to be below $328\,\mu\text{s}$. This result is suitable for low-latency applications $< 1\,\text{ms}$ while other parameter sets may even deliver lower latencies for smaller frames.

V. Conclusion

We presented a GFDM transceiver which runs completely in software. It is capable of providing low latency signal processing for high data rate broadband SDR environments. The common code-base enables simulations as well as field tests for rapid prototyping. The performance indicates that GPP hardware is capable enough to be used in low-latency systems.

Acknowledgment

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

References

[1] A. Osseiran, J. F. Monserrat, and P. Marsch, *5G Mobile and Wireless Communications Technology*, 1st ed. New York, NY, USA: Cambridge University Press, 2016.

[2] F. Schaich and T. Wild, "Waveform contenders for 5G - OFDM vs. FBMC vs. UFMC," in *ISCCSP 2014 - 2014 6th International Symposium on Communications, Control and Signal Processing, Proceedings*, 2014.

[3] A. Sahin, I. Guvenc, and H. Arslan, "A survey on multicarrier communications: Prototype filters, lattice structures, and implementation aspects," *IEEE Communications Surveys and Tutorials*, 2014.

[4] V. Vakilian, T. Wild, F. Schaich, S. Ten Brink, and J. F. Frigon, "Universal-filtered multi-carrier technique for wireless systems beyond LTE," in *2013 IEEE Globecom Workshops, GC Wkshps 2013*, 2013.

[5] N. Michailow, M. Matthe, I. S. Gaspar, A. N. Caldevilla, L. L. Mendes, A. Festag, and G. Fettweis, "Generalized frequency division multiplexing for 5th generation cellular networks," *IEEE Transactions on Communications*, 2014.

[6] M. Danneberg, N. Michailow, I. Gaspar, M. Matthé, D. Zhang, L. L. Mendes, and G. Fettweis, "Implementation of a 2 by 2 MIMO-GFDM Transceiver for Robust 5G Networks," in *International Symposium on Wireless Communication Systems (ISWCS)*, Brussels, 2015.

[7] N. Otterbach, M. Braun, and F. K. Jondral, "Wireless Networks In-the-Loop: Creating an SDR Development Environment," in *IEEE International Symposium on Wireless Communication Systems (ISWCS)*, Ilmenau, 2013.

[8] P. Rost, I. Berberana, A. Maeder, H. Paul, V. Suryaprakash, M. Valenti, D. Wübben, A. Dekorsy, and G. Fettweis, "Benefits and challenges of virtualization in 5G radio access networks," in *IEEE Communications Magazine*, 2015.

[9] N. Michailow, S. Krone, M. Lentmaier, and G. Fettweis, "Bit Error Rate Performance of Generalized Frequency Division Multiplexing," in *IEEE Vehicular Technology Conference (VTC Fall)*, Quebec City, 2012.

[10] 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. [Online]. Available: http://www.scc2015.net/

[11] J. Du, *Pulse Shape Adaptation and Channel Estimation in Generalised Frequency Division Multiplexing Systems*. Stockholm: KTH, 2008.

[12] I. Gaspar, N. Michailow, A. Navarro, E. Ohlmer, S. Krone, and G. Fettweis, "Low complexity GFDM receiver based on sparse frequency domain processing," in *IEEE Vehicular Technology Conference*, 2013.

[13] M. Matthe, L. L. Mendes, and G. Fettweis, "Generalized frequency division multiplexing in a gabor transform setting," *IEEE Communications Letters*, 2014.

[14] 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," IEEE, Tech. Rep., 2012. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=6178209

[15] J. Demel, S. Koslowski, and F. K. Jondral, "A LTE receiver framework using GNU Radio," *Journal of Signal Processing Systems*, vol. 78, no. 3, 2015.

[16] A. B. Awoyesila, C. Kasparis, and B. G. Evans, "Improved preamble-aided timing estimation for OFDM systems," *IEEE Communications Letters*, 2008.

[17] I. S. Gaspar, L. L. Mendes, N. Michailow, and G. Fettweis, "A synchronization technique for generalized frequency division multiplexing," *EURASIP Journal on Advances in Signal Processing*, 2014. [Online]. Available: http://asp.eurasipjournals.com/content/2014/1/67

[18] J. Mitola, "Software Radios Survey, Critical Evaluation and Future Directions," 1992.

[19] A. Rode, "gr-gfdm," https://github.com/kit-cel/gr-gfdm, accessed: 2016-10-27.

[20] GNU Radio, "GNU Radio website," http://gnuradio.org/, accessed: 2016-10-27.

[21] VOLK, "VOLK website," http://libvolk.org/, accessed: 2016-11-03.

[22] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," in *Proceedings of the IEEE*, 2005.

[23] Boost, "Boost.Chrono website," http://www.boost.org/doc/libs/1_53_0/doc/html/chrono/users_guide.html, accessed: 2016-11-14.