Real-Time Generation of Standard-Compliant DVB-T Signals

Giuseppe BARUFFA, Luca RUGINI, Fabrizio FRESCURA, Paolo BANELLI

Dept. of Engineering, University of Perugia, Via G. Duranti 93, 06125 Perugia, Italy

{giuseppe.baruffa, luca.rugini, fabrizio.frescura, paolo.banelli}@unipg.it

Submitted July 27, 2017 / Accepted December 6, 2017

Abstract. *This paper proposes and discusses two software implementations of the DVB-T modulator, using C++ and MATLAB, respectively. All the key features of the DVB-T standard are included. The C++ DVB-T modulator, incorporated into the Iris framework developed by Trinity College of Dublin, works in real time on an Intel Core i7 2.4 GHz CPU with the Iris testbed. The MATLAB-based DVB-T modulator is coupled with a receiver implementation with channel estimation, equalization, soft-output demapping, and channel decoding. The validation step demonstrates that the proposed DVB-T software implementations generate standard-compliant DVB-T signals that are correctly received by commercially available TV sets and USB dongles. The software code for the Iris-based C++ modulator, and for the MATLAB-based modulator and receiver, has been made publicly available under the GNU license.*

Keywords

Digital Video Broadcasting-Terrestrial (DVB-T), OFDM, software-defined radio (SDR), C++, MATLAB

1. Introduction

Digital Video Broadcasting-Terrestrial (DVB-T) is a popular broadcasting standard used in Europe and worldwide [1], [2], [3]. Traditionally, DVB–T transmitters make use of hardware-based architectures, either customized or reprogrammable. However, software-defined radio (SDR) solutions are becoming attractive in order to incorporate many software-defined communication standards into a single device, which can be driven by a low-cost personal computer (PC) with multicore central processing units (CPUs) [4], [5], [6].

In the last fifteen years, SDR solutions for DVB-T systems have been investigated by many researchers, using different software languages, mainly C++ and MATLAB. For what concerns the C++ language, [7] proposed a mixed SDR digital signal processor (DSP) platform that uses a host PC and an MDS TM-13 IREF DSP board. A complete DVB-H transmitter and receiver chain that runs in real time on a multithreaded DSP was described in [8]. In 2008, [9] proposed a GNU Radio-based DVB-T modulator that integrates a hybrid C++/Python code with a universal software radio peripheral (USRP) front-end. A C++ receiver was then proposed in [10] to reduce the computational cost with respect to [9]. By means of accurate optimization strategies and a memory-based acceleration technique [11], real-time reception and decoding of TV signals can be achieved. A memory-efficient DVB-T/H receiver has been proposed in [12], using a DSP-based and accelerator-assisted SDR architecture together with multithreading and a parallelized version of the C language. A software-defined C receiver implementation for ISDB-T is described in [13]: this receiver can operate on a single frequency segment of the ISDB-T signal, thus allowing to receive a standard definition TV signal in real time. C++ software for DVB-T signal reception has been proposed in [14], [15], [16], [17], which demonstrate the real-time feasibility for the whole receiving chain. In [18], the most time-consuming processing stages (Viterbi decoder and FFT) have been implemented using a graphic processing unit, whereas the remaining stages still run on the CPU and are implemented in C; however, the Reed-Solomon (RS) decoder has not been considered in the reception chain. In 2014, we proposed an USRP-based C DVB-T transmitter that works in real time and makes use of parallelization strategies such as multithreading and vectorization [19]. The Brazilian version of ISDB-T has been implemented in [20] at the transmission side on a regular PC, using GNU radio and C++, achieving real-time processing.

MATLAB software for digital video broadcasting systems has been proposed as well. For instance, [21] develops DVB-T transmission and reception chains, with a subsequent SIMD-oriented pre-compilation step that provides optimized assembly code for specialized reprogrammable processors. Some portions of the DVB-T reception chain have been implemented in MATLAB in [22] for the purpose of localization of GNSS systems: due to the low complexity of the synchronization algorithm, real-time processing was achieved. A MATLAB-based implementation of DVB-T transmitter and receiver is proposed in [23] and simulated in [24], focusing on channel coding and decoding operations. A MATLAB DVB-T receiver is described in [19], including mode detection and time and frequency synchronization.

This paper presents and describes a new software implementation of the DVB-T modulator using C++. Differently from previous literature (e.g., [9] and [19]), the designed C++ code is integrated into the Iris SDR testbed developed by Trinity College of Dublin (TCD) [25]. Actually, the Iris testbed and the associated software enclose several features (e.g., runtime reconfiguration, network stack support, embedded systems support, component-based architecture [25]) and also provide implicit multithreading capabilities. This way, differently from [19], real-time transmission can be achieved without applying specific parallelization strategies to the designed DVB-T C++ code. For the purpose of validation of the designed DVB-T C++ software, we also include the implementation of a MATLAB-based DVB-T modulator: the validation procedure compares the output of the C++ DVB-T modulator with the output of the MATLAB DVB-T modulator. The correctness of the validation has been further confirmed by the correct reception, on regular TV sets and USB dongles, of the DVB-T signal transmitted on-air in a laboratory environment.

In addition to a standard-compliant DVB-T modulator, the designed MATLAB code also includes a DVB-T receiver with channel estimation, equalization, soft-output demapping, Viterbi decoding, deinterleaving, Reed-Solomon (RS) decoding, and descrambling. Differently from most of the previous SDR-based DVB-T systems proposed in the literature so far, an important feature of the designed C++ and MATLAB code is the public availability under the GNU license (at https://github.com/wishful-project/module_iris/tree/master/dvb-tx-iris), to enable reuse by other researchers and practitioners.

The remainder of this paper is organized as follows. Section 2 presents a brief overview of the DVB-T standard. Section 3 describes the implementation and the validation of the C++ DVB-T modulator, while Section 4 illustrates the MATLAB implementation of DVB-T. Section 5 discusses the impact of the proposed software, while Section 6 concludes the paper.

2. Overview of the DVB-T Standard

The DVB-T modulator can be summarized as in Fig. 1 [1]. The data to be transmitted are MPEG-2 Transport Stream (TS) packets, each with 188 data bytes containing compressed video and audio. Groups of eight TS packets are scrambled using a pseudo-random binary sequence that produces a scrambled sequence with 1503 bytes. The obtained sequence is then parsed and encoded using a nonbinary shortened RS code with 188 input bytes and 204 output bytes (the additional 51 zero bytes are not transmitted) [1]. These output bytes are convolutionally interleaved using $I = 12$ delay paths and memory cells with $M_I = 204/I = 17$ bytes. Then, the interleaved bits are convolutionally encoded using a rate $r_c = 1/2$ and a constraint length $L = 7$. The code rate can be increased up to $r_c \in \{2/3, 3/4, 5/6, 7/8\}$ by puncturing (nonequispaced bit decimation) [1]. Successively, bit interleaving and ν-bits symbol interleaving is performed using a RAM whose size depends on the DVB-T mode (6048 cells for 8K, 1512 cells for 2K). Afterward, the mapper uses Gray encoding and square quadrature-amplitude modulation (QAM) with $M = 2^{\nu}$. The obtained QAM data cells are inserted into a frame adapter that includes signaling information, such as boosted pilots for channel estimation and transmission parameter signaling (TPS). The whole frame is then converted in orthogonal frequency-division multiplexing (OFDM) symbols, with cyclic prefix (CP): one frame contains $N_F = 68$ OFDM symbols, and one superframe contains four frames. The OFDM modulator is detailed in Sec. 2.1. If necessary, the DVB-T modulator can include an interpolator that adjusts the sample rate and a spectrum emission mask (SEM) filter. Interpolation and SEM filtering is described in Sec. 2.2.

2.1 OFDM

Each OFDM symbol contains K active carriers and $N_{\mathrm{FFT}} - K$ virtual carriers as guard frequency bands. N_{FFT} is the FFT size. OFDM performs an inverse FFT that converts the assembled data from the frequency domain to the time domain. A CP (with N_G samples) is also prepended to each OFDM symbol. The OFDM baseband signal $\tilde{s}[n]$ can be expressed by

$$\tilde{s}[n] = \sum_{m=0}^{+\infty} \sum_{l=0}^{N_F-1} z_{m,l}[n] = \sum_{m=0}^{+\infty} \sum_{l=0}^{N_F-1} \sum_{k=0}^{K-1} c_{m,l,k} G_k \psi_{m,l,k}[n] \qquad (1)$$

where m is the frame index, l is the OFDM symbol index, k is the subcarrier index, $c_{m,l,k}$ is the QAM data cell, G_k is a weighting factor that pre-compensates linear distortions caused by following stages ($G_k = 1$ if no distortion), and $z_{m,l}[n]$ is the OFDM symbol in the time domain.

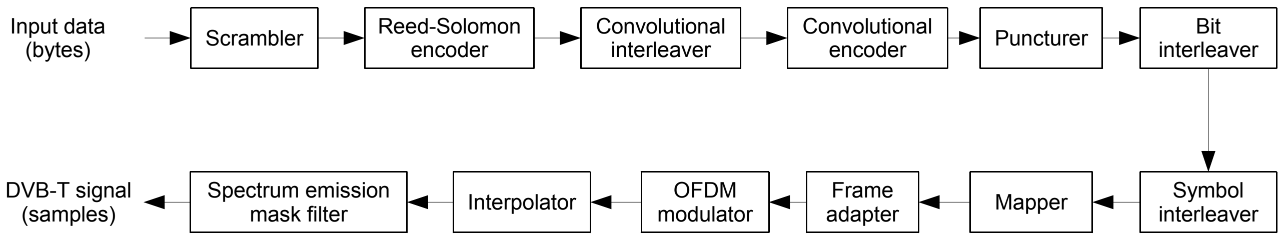


Fig. 1. Block diagram of the DVB-T modulator.

The orthogonal carriers $\psi_{m,l,k}[n]$ are expressed by

$$\psi_{m,l,k}[n] = \exp\left\{ j2\pi \frac{k-K_2}{N_{\mathrm{FFT}}} \left[n - N_{\mathrm{G}} - (l + mN_{\mathrm{F}})N_{\mathrm{S}}\right] \right\} \Pi_{N_{\mathrm{S}}}\left[n - (l + mN_{\mathrm{F}})N_{\mathrm{S}}\right] \quad (2)$$

where $K_2 = (K-1)/2$, $N_{\mathrm{S}} = N_{\mathrm{FFT}} + N_{\mathrm{G}}$, and $\Pi_{N_{\mathrm{S}}}[n]$ is equal to one for $n = 0, 1,\ldots, N_{\mathrm{S}}-1$ and to zero elsewhere. There are 3 possible modes for OFDM: $N_{\mathrm{FFT}} \in \{2048, 4096, 8192\}$ for 2K, 4K (in DVB-H only), and 8K, respectively, which correspond to $K \in \{1705, 3409, 6817\}$ active carriers. The CP ratio $N_{\mathrm{G}}/N_{\mathrm{FFT}}$ is selected in the set $\{1/32, 1/16, 1/8, 1/4\}$.

2.2 Interpolation and SEM Filter

The interpolator is not part of the DVB-T standard [1] and is required only when the digital-to-analog converter (DAC) sample rate $f_{\mathrm{s,DAC}}$ is different than the DVB-T sampling rate $f_{\mathrm{s,DVB\text{-}T}} = 64/7$ MHz. When $f_{\mathrm{s,DAC}} > f_{\mathrm{s,DVB\text{-}T}}$, the oversampling ratio is $R_{\mathrm{o}} = f_{\mathrm{s,DAC}}/f_{\mathrm{s,DVB\text{-}T}} = L_{\mathrm{o}}/M_{\mathrm{o}}$, where L_{o} and M_{o} are coprime. The sampling rate conversion can be done by an interpolating filter with impulse response $h_{\mathrm{I}}(t)$, producing as output [19]

$$\tilde{y}[n] = \sum_{i=P_1}^{P_2} \tilde{s}[p_n - i]\, h_{\mathrm{I}}\left((i + \mu_n) T_{\mathrm{s,DVB\text{-}T}}\right) \quad (3)$$

where $p_n = \lfloor nT_{\mathrm{s,DAC}} / T_{\mathrm{s,DVB\text{-}T}} \rfloor = \lfloor n/R_{\mathrm{o}} \rfloor$, $\mu_n = n/R_{\mathrm{o}} - p_n$, $T_{\mathrm{s,DVB\text{-}T}} = 1/f_{\mathrm{s,DVB\text{-}T}}$, and $T_{\mathrm{s,DAC}} = 1/f_{\mathrm{s,DAC}}$. The interpolator distortion can be pre-compensated by choosing $G_k = 1/H_{\mathrm{I}}((k-K_2) f_{\mathrm{s,DVB\text{-}T}}/N_{\mathrm{FFT}})$ in (1), where $H_{\mathrm{I}}(f)$, is the Fourier transform of $h_{\mathrm{I}}(t)$.

The SEM filter (not in the DVB-T standard [1]) is required only when the SEM has to be imposed at the baseband level. This filter also reduces the IF images caused by interpolation [19]. The number of taps depends on the required stopband attenuation and transition bandwidth, and should be low for real-time operation.

3. C++ DVB-T Modulator

The implemented C++ DVB-T modulator is structured with a modular design that makes the designed software easily embeddable into existing SDR frameworks.

3.1 Overview of the Iris Framework

The Iris SDR framework [25], developed by the Telecommunications Research Centre at TCD, is primarily employed to perform cognitive-radio experiments and enable dynamic spectrum access [26], so it is highly suitable for real-time signal processing operations. The Iris framework, written in C++, is intended to run on general-purpose processors and can be configured to perform many different radio structures. A radio structure is built on basic blocks called components, which perform a particular process of the radio chain. A data buffer is positioned between adjacent components by the framework, thus allowing data to be exchanged between the components. Metadata are passed among components to specify particular information data, such as a time stamp or a data rate. The Iris framework has also the task to perform data safety, that is, one component cannot use data while another component is still producing them, and vice versa. A useful feature of the Iris framework is its capability of parallelization of sequential tasks: as explained in Sec. 3.1, parallelization can be achieved by assigning group of components (whose tasks can proceed sequentially) to an *engine*, i.e., an Iris framework environment that has its own data buffers and can run in a thread separate from that of the other engines.

During a typical call to the Iris framework, the host application reads a configuration file and assembles a radio structure through the Iris application programmer interface (API). A component manager is accessed to select the requested components from a library of precompiled components that include the designed DVB-T software. All the selected components are then instantiated and executed by the radio engine, to perform the designed task.

3.2 Design of the C++ Code Structure

Each DVB-T basic building block has been assigned to a single serial task that runs inside a unique Iris component. This way, parallelization can be achieved by instantiating different components to separate engines. To achieve the highest degree of parallelization, we choose to fit every component inside its own engine: indeed, this design philosophy permits a pipeline-like execution of the instructions of consecutive engines, thereby minimizing the processing latencies. Hence, we discarded the alternative philosophy that fits all components inside a single engine: indeed, this way the DVB-T data flow would be executed serially and would introduce large latencies in the processing of consecutive data blocks, because the engine would have to wait for the end of the processing of the last component before reactivating the first component. To maximize the number of engines, we split the DVB-T modulator in 12 functional components, i.e., the 12 blocks of Fig. 1. Next, we separately designed each block, as explained in Sec. 3.3. As verified in Sec. 3.4, this design method enables real-time operation of the DVB-T modulator. Clearly, this highly parallel design increases the usage of computing resources, such as RAM and threads, with respect to a slower solution that uses a unique engine shared by all the components.

3.3 Design of the C++ Blocks

Every block is designed according to the Iris API in C++ language. Each block is instantiated as a single component in a derived base class of the Iris API specifications [25], thus inheriting all the methods needed for the proper management of that component. Component data processing methods are implemented as specified by the Iris API [25]. Thus, a generic component requests a data buffer to process, waiting in idle mode until input data are provided. The designed blocks are generally unable to request a particular number of input samples or data. All the blocks, except the scrambler and the SEM filter, have their own private internal data buffer, where multiple data input bursts can be stored, until a sufficient number of samples is collected in order to trigger the designed signal processing operation.

The C++ code of the first 10 blocks has been designed by translating the DVB-T standard specifications [1] into lines of code, avoiding the use of existing libraries: this way, we have maximum control of the specific operation of each block. The only exception to this general rule is the use of the FFTW library [27] inside the OFDM block, for maximum performance and portability. The interpolator and the SEM filter have been designed [19] assuming an Ettus USRP N210, whose sample frequency is $f_{\mathrm{s,DAC}} = 100/n_{\mathrm{DAC}}$ MHz (n_{DAC} is a positive integer). The designed SEM filter uses a Kaiser window with passband bandwidth $W = (K + 1)\, f_{\mathrm{s,DVB\text{-}T}} / N_{\mathrm{FFT}} \approx 7.6$ MHz, transition bandwidths of 700 kHz, and stopband attenuation of 25 dB.

3.4 Validation of the C++ Code

The code has been validated using the automated testing provided within the Iris framework building step. The command *cmake* with *ctest* [28] automatically performs several tests on both code sanity and expected outputs. Concerning the code sanity tests, we performed the following validation actions: (a) *smoke tests* to verify that the code compiles and links correctly into an executable; (b) *crash-proof tests* to verify that the produced executable runs without crashes; (c) *black-box tests* to verify that the number and data types of inputs and outputs match the expected ones, without knowledge of the implementation of the block; (d) *white-box tests* to verify that the number and data types of inputs and outputs match the expected ones, with knowledge of the implementation of the block.

Concerning the tests on the expected outputs of the DVB-T modulator, we have compared the output data of the C++ modulator to a reference output, for a predefined set of input data. To obtain the reference outputs, we have implemented the first 10 blocks of Fig. 1 in a high-level interpreted language, i.e., MATLAB. For those blocks operating on fixed point data (i.e., the first seven blocks of Fig. 1), we required that the output data samples of the two different implementations (C++ design and MATLAB reference) are bit-by-bit equal. Differently, for those blocks operating on floating-point data (i.e., mapper, framer, and OFDM), we required that the output data samples of the two implementations have a maximum difference below a preselected tolerance. This tolerance depends on the expected quantization step ΔQ, which in turn depends on the characteristics of the adopted DAC. For instance, assuming that the real and the imaginary parts of the signal are in the range $[-1,+1]$, a DAC with $N_{\mathrm{DAC}} = 14$ bits gives $\Delta Q = 2/2^{N_{\mathrm{DAC}}} \approx 1.2\times10^{-4}$. Thus, for validation purposes, two floating point values are considered equal when their difference is below a tolerance value ε, where ε is chosen such as $\varepsilon < \Delta Q$. In this case, the two floating point values will produce samples that either are equal or differ at maximum by one quantization level ΔQ only. In our tests, $\varepsilon = 10^{-4}$.

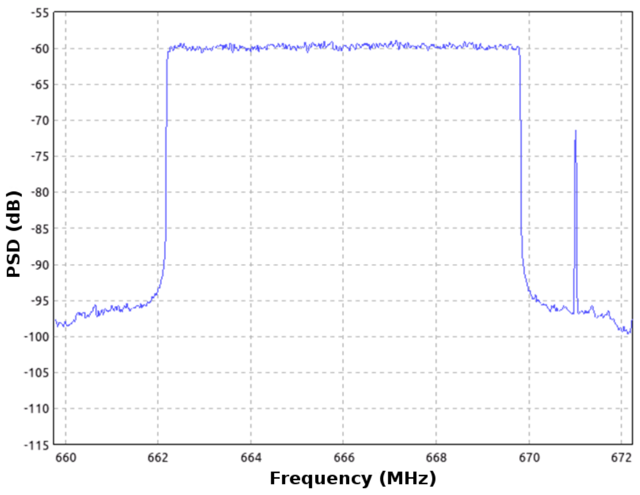


Fig. 2. PSD of the generated DVB-T signal.

Note that the last two optional blocks of the DVB-T modulator of Fig. 1 have not been implemented in MATLAB. Concerning the SEM filter output, Figure 2 shows the power spectrum density (PSD) of the generated DVB-T signal, at a carrier frequency $f_c = 666$ MHz, using two Ettus USRP N210 (one as transmitter and another one as spectrum analyzer). Note the flat PSD in the passband and the total attenuation of 35 dB in the stopband. The spike at 671 MHz is generated by the local oscillator carrier, using a frequency offset for near-zero IF configuration ($f_{\mathrm{IF}} = 5$ MHz): this spike can be filtered out with an analog filter during the final amplification stage. If a final analog filter is absent (such as in USRP devices), this spike can be moved into the transition band to avoid adjacent channel interference (e.g., by using $f_{\mathrm{IF}} = 3.9$ MHz).

Summarizing, all the different tests performed for every block of the DVB-T modulator chain have been completely successful, including all the code sanity tests and all the expected output tests. In addition, we have also verified that the generated DVB-T signal, when transmitted in real time by an Ettus USRP (such as N210, B205, B210) in a laboratory environment, is correctly received by both commercial TV sets and USB dongles.

3.5 Time Performance

To enable real-time operation of the designed DVB-T modulator, every block should be able to achieve a processing speed higher than the speed (specified in [1]) required to perform the block task. Iris contains dedicated steps for the benchmarking of single processing components, which in our case coincide with the DVB-T blocks.

This benchmarking suite provides the speed of the block, to be compared with the time requirement. For every block, the maximum input data rate $R_{\mathrm{st,max}}$ is constrained by [1] and by the input sample rate of the SDR device. For the USRP, we used $f_{\mathrm{s,DAC}} = 10$ MHz ($f_{\mathrm{s,DAC}} = 12.5$ MHz), which leads to $R_{\mathrm{o}} = 35/32$ ($R_{\mathrm{o}} = 175/128$) and to a SEM filter with 19 taps (23 taps). By comparing $R_{\mathrm{st,max}}$ with the actual data rate R_{act} measured by the suite, we obtain the speed-up factor $F_{\mathrm{su}} = R_{\mathrm{act}} / R_{\mathrm{st,max}}$ of each block. We assume that real-time operation is possible when, for all the blocks, $F_{\mathrm{su}} > 1.2$, where the 20% margin accounts for additional latencies due to the data exchange among blocks (operated by the engines) and overall thread execution scheduling.

Table 1 reports the achieved performance R_{act}, the required maximum data rate $R_{\mathrm{st,max}}$, and the speed-up factor F_{su} of all the blocks. The benchmarks are run on an Intel Core i7-3630QM CPU at 2.4 GHz with 8 GB DDR3 RAM and Ubuntu 16.04 OS, and the programs have been compiled with the GNU C++ compiler v. 5.4.0. Benchmarking is performed adopting input data of large size, thus minimizing the impact of the Iris engine overhead on data buffer management. All the speed-up factors are above the threshold value 1.2: in Tab. 1, the fastest blocks are the scrambler and the convolutional interleaver, while the slowest blocks are the SEM filter and the RS encoder.

Table 2 reports the achieved time performance for the whole chain. The different configurations of Tab. 2 consider the different DVB-T parameters (number of carriers, CP length, modulation, and code rate) and the absence or presence of the optional blocks in Fig. 1, i.e., the interpolator and the SEM filter. The output of the last block in the chain (either the OFDM modulator or the SEM filter) has been forwarded to a virtual dummy load (the */dev/null* device of the Ubuntu file system). In all cases, we obtained a typical speed-up factor of $F_{\mathrm{su}} = 2.5$ in the absence of interpolation and SEM filtering, and of $F_{\mathrm{su}} = 1.8$ in the presence of interpolation and SEM filtering. Since the obtained speed-up factors are greater than 1.2 in all cases, we assume that our C++ DVB-T software can work in real time without problems. This is confirmed by all the real-time experiments we have performed. Note that the speed-up factors of Tab. 2 for the aggregate system are significantly lower than the speed-up factors of the slowest blocks in Tab. 1. This means that the latencies due to the data exchange among blocks and the thread execution scheduling times cannot be neglected.

Block		R_{act}	$R_{\mathrm{st,max}}$	F_{su}
Scrambler		4158.5	31.7	131.3
RS encoder		211.8	31.7	6.7
Convolutional interleaver		4016.4	34.4	116.8
Convolutional encoder		409.0	34.4	11.9
Puncturer	1/2	552.0	68.7	8.0
	2/3	612.7	68.7	8.9
	3/4	627.0	68.7	9.1
	5/6	616.6	68.7	9.0
	7/8	604.7	68.7	8.8
Bit interleaver	4QAM	357.3	39.3	9.1
	16QAM	374.5	39.3	9.5
	64QAM	379.1	39.3	9.6
Symbol interleaver	2K	317.3	6.5	48.8
	4K	313.0	6.5	48.2
	8K	314.7	6.5	48.4
Mapper	4QAM	485.1	6.5	74.6
	16QAM	480.0	6.5	73.8
	64QAM	449.1	6.5	69.1
Frame Adapter	2K	197.5	6.5	30.4
	4K	191.2	6.5	29.4
	8K	178.4	6.5	27.4
OFDM	2K 1/32	122.0	7.4	16.5
	8K 1/4	96.4	6.1	15.8
Interpolator	35/32	120.8	9.1	13.2
	175/128	103.2	9.1	11.3
SEM filter	19 taps	52.6	10.0	5.3
	23 taps	45.7	12.5	3.7

Tab. 1. Input data rates and speed-up factor for the compiled C++ blocks. Data rates are in Mb/s for the first six blocks and in Msample/s for the last six blocks.

Configuration	SEM filter	R_{act}	$R_{\mathrm{st,max}}$	F_{su}
8K, 1/4, 4QAM, 1/2	No	12.85	4.98	2.6
8K, 1/4, 4QAM, 1/2	23 taps	9.38	4.98	1.9
8K, 1/4, 4QAM, 2/3	No	17.17	6.64	2.6
8K, 1/8, 4QAM, 5/6	No	23.63	9.22	2.6
8K, 1/32, 16QAM, 1/2	No	30.75	12.06	2.5
8K, 1/16, 16QAM, 2/3	No	39.48	15.61	2.5
8K, 1/16, 16QAM, 5/6	No	49.37	19.52	2.5
8K, 1/4, 64QAM, 3/4	No	57.15	22.39	2.6
8K, 1/4, 64QAM, 3/4	23 taps	40.72	22.39	1.8
8K, 1/8, 64QAM, 5/6	No	70.10	27.65	2.5
8K, 1/32, 64QAM, 7/8	No	80.03	31.67	2.5
8K, 1/32, 64QAM, 7/8	23 taps	57.58	31.67	1.8
2K, 1/32, 64QAM, 7/8	No	80.32	31.67	2.5

Tab. 2. Input data rates and speed-up factor for the whole DVB-T modulator. Data rates are expressed in Mb/s.

When the software runs in real time, typically the overall CPU load is about 16%. Therefore, the remaining computational resources are sufficient for performing other tasks, such as retrieving or building the transport stream from a local disk, or receiving the transport stream by a remote network connection. The used memory typically amounts to roughly 100 MB and is almost constant for all the configurations. Such computational loads could allow a single modern CPU to run several instances of the DVB-T modulator in parallel, each one allocated to a different TV channel, with the only bottleneck consisting in the limited bandwidth of the communication channel used to convey the generated waveform samples towards the SDR devices.

4. MATLAB DVB-T Software

4.1 Modulator

As explained in Sec. 3.4, the first 10 blocks of Fig. 1 have been implemented also in MATLAB, for the purpose of validation of the designed C++ blocks. These MATLAB scripts have been designed independently from the C++ blocks, starting from the DVB-T standard [1]. These scripts have been further validated by comparing their outputs with other MATLAB implementations, such as [23]. This comparison has revealed a minor typo in the RS encoder of [23] (the code generator polynomial in [1] is different from the MATLAB default one used in [23]).

4.2 Demodulation and Decoding

The block diagram of a DVB-T receiver is shown in Fig. 3. We assume that a previous subsystem has already detected the DVB-T operating parameters (OFDM mode, CP, and code rate) and has properly synchronized the received OFDM symbols, both in time and frequency. For instance, parameter detection and synchronization could be performed as in [19]. Thus, the MATLAB receiver removes the CP from the received signal, and subsequently performs the FFT. After the FFT processing, the noisy received signal $R[k]$ on the k-th carrier can be expressed by

$$R[k] = C[k]S[k] + W[k] \tag{4}$$

where $C[k]$ is the channel transfer function, $S[k]$ is the data (or pilot, or TPS) signal, and $W[k]$ is the additive white Gaussian noise (AWGN), on the k-th carrier, and $k = 0, \ldots, K-1$ represents the active carrier index. Then, the data carrier samples are separated from the pilot carrier samples: the information conveyed by the pilot carriers is instrumental to estimate the channel transfer function $C[k]$ on the k-th carrier, for equalization purposes. The channel estimator performs a least-squares interpolation based on the FFT, as in [2]. In practice, the estimated channel $\hat{C}[k]$ is a noisy copy of the real channel, as expressed by

$$\hat{C}[k] = C[k] + W_{\mathrm{C}}[k] \tag{5}$$

where $W_{\mathrm{C}}[k]$ is the channel estimation error, mainly due to receiver AWGN. The data carriers are then equalized using the zero-forcing criterion, as

$$Y[k] = \frac{R[k]}{\hat{C}[k]} \tag{6}$$

where $Y[k]$ is the noise-affected equalized constellation point and $R[k]$ is the received constellation point expressed by (4). The equalized data form the basis for the estimation of the log-likelihood ratios (LLRs), which enable the soft-input decoding of the convolutional code. These LLRs can be computed using different techniques: in our case, we have adopted a Log-max approximation [29], [30], to achieve a good trade-off between decoding accuracy and computational complexity. For the generic m-th bit carried by the k-th QAM symbol, the Log-max LLR $\Lambda(b_{k,m})$ is expressed as

$$\Lambda(b_{k,m}) = \frac{|\hat{C}[k]|^2}{4}\left[\min_{\alpha\in S_m^0}\left|Y[k]-\alpha\right|^2 - \min_{\alpha\in S_m^1}\left|Y[k]-\alpha\right|^2\right] \tag{7}$$

where S_m^0 is the subset of constellation points where the m-th bit is 0, and S_m^1 is the subset of constellation points where the m-th bit is 1.

The outputs of the LLR block are then forwarded to the symbol and bit deinterleavers, and then to the depuncturer. Depuncturing assigns a null value to the LLR of erased bits, such that the subsequent Viterbi decoder will not consider the contribution of erased bits in the computation of branch and path metrics. The Viterbi decoder uses soft inputs with block truncation, and should use a trace-back length l_{TB} several times larger than the code constraint length, in order to take into account the effects of punctured bits. The chosen Viterbi algorithm has a hard-decided output, thus the decoded bits are sent to the convolutional deinterleaver and to the RS decoder. The RS decoder pads with zeroes the input words and follows the usual decoding procedure (syndrome computation, error location, error evaluation, and error correction). The decoded message words are then descrambled and the original TS data are recovered.

4.3 Performance Results

Figure 4 shows the results obtained by simulations on the AWGN and P1 channel [1], represented in terms of bit error rate (BER) after soft Viterbi decoding versus the carrier-to-noise ratio (C/N). The C/N is defined as

$$\frac{C}{N} = \frac{P_{\mathrm{R}}}{\sigma_{\mathrm{W}}^2} = \frac{\sum_{k=0}^{K-1} E\{|C[k]S[k]|^2\}}{E\{|W[k]|^2\}\sum_{k=0}^{K-1} E\{|C[k]|^2\}} \tag{8}$$

where $P_{\mathrm{R}} = \sum_{k=0}^{K-1} E\{|C[k]S[k]|^2\} \big/ \sum_{k=0}^{K-1} E\{|C[k]|^2\}$ is the average power of the received signal, $\sigma_{\mathrm{W}}^2 = E\{|W[k]|^2\}$ is the AWGN variance, and $E\{\cdot\}$ is the expectation operator. The P1 multipath channel (static, non-line-of-sight) leads to

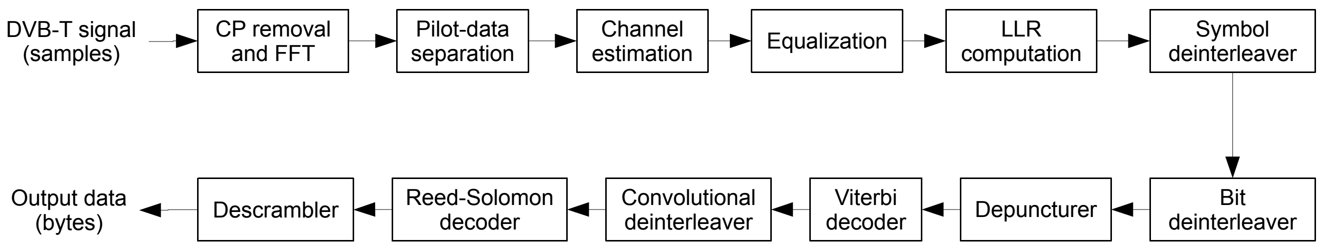
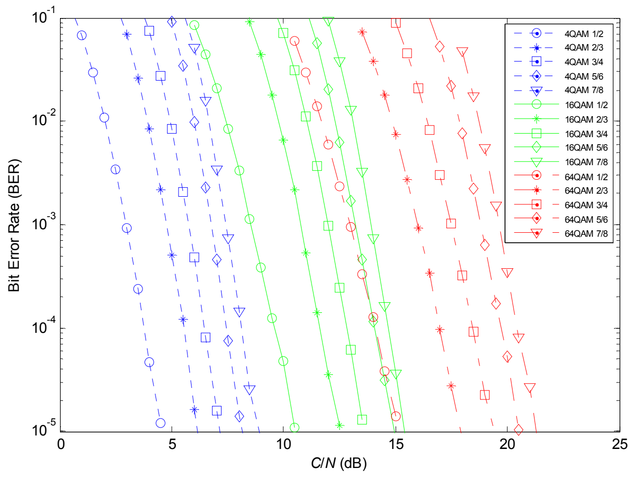
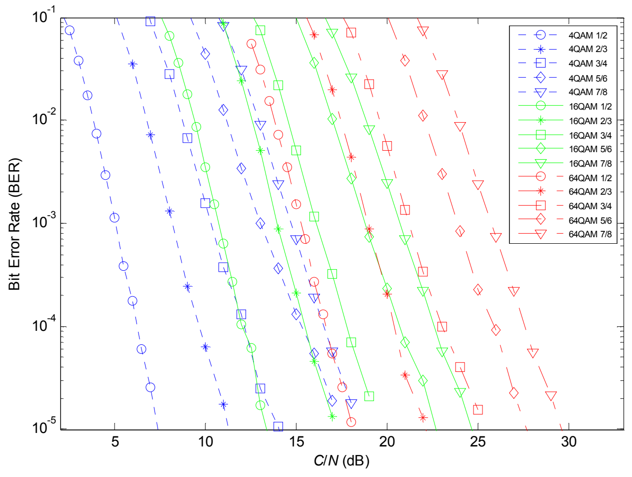


Fig. 3. Block diagram of the MATLAB DVB-T receiver.



(a)



(b)

Fig. 4. Simulated performance of DVB-T using the MATLAB code: (a) AWGN channel; (b) P1 channel.

M	CR	C/N (AWGN)		C/N (P1)	
		[1]	Sim.	[1]	Sim.
4	1/2	3.5	3.6	5.9	5.9
	2/3	5.3	5.3	9.6	9.2
	3/4	6.3	6.2	12.4	11.6
	5/6	7.3	7.2	15.6	14.6
	7/8	7.9	7.9	17.5	15.9
16	1/2	9.3	9.3	11.8	11.7
	2/3	11.4	11.4	15.3	15.0
	3/4	12.6	12.6	18.1	17.3
	5/6	13.8	13.8	21.3	20.1
	7/8	14.4	14.4	23.6	22.1
64	1/2	13.8	13.8	16.4	16.2
	2/3	16.7	16.7	20.3	20.0
	3/4	18.2	18.2	23.0	22.4
	5/6	19.4	19.4	26.2	25.2
	7/8	20.2	20.2	28.6	27.2

Tab. 3. Simulated QEF C/N (in dB) of the DVB-T standard.

M	CR	AWGN		F1 channel		P1 channel	
		QEF	BER	QEF	BER	QEF	BER
4	1/2	3.5	$2.0{\cdot}10^{-4}$	4.1	$1.5{\cdot}10^{-4}$	5.9	$2.0{\cdot}10^{-4}$
	2/3	5.3	$2.0{\cdot}10^{-4}$	6.1	$1.4{\cdot}10^{-4}$	9.6	$1.0{\cdot}10^{-4}$
	3/4	6.3	$1.9{\cdot}10^{-4}$	7.2	$1.9{\cdot}10^{-4}$	12.4	$7.3{\cdot}10^{-5}$
	5/6	7.3	$1.9{\cdot}10^{-4}$	8.5	$1.4{\cdot}10^{-4}$	15.6	$7.8{\cdot}10^{-5}$
	7/8	7.9	$1.9{\cdot}10^{-4}$	9.2	$2.3{\cdot}10^{-4}$	17.5	$3.6{\cdot}10^{-5}$
16	1/2	9.3	$1.8{\cdot}10^{-4}$	9.8	$1.7{\cdot}10^{-4}$	11.8	$1.5{\cdot}10^{-4}$
	2/3	11.4	$2.0{\cdot}10^{-4}$	12.1	$1.6{\cdot}10^{-4}$	15.3	$1.3{\cdot}10^{-4}$
	3/4	12.6	$1.9{\cdot}10^{-4}$	13.4	$2.0{\cdot}10^{-4}$	18.1	$7.1{\cdot}10^{-5}$
	5/6	13.8	$2.0{\cdot}10^{-4}$	14.8	$1.9{\cdot}10^{-4}$	21.3	$5.2{\cdot}10^{-5}$
	7/8	14.4	$2.1{\cdot}10^{-4}$	15.7	$1.6{\cdot}10^{-4}$	23.6	$3.6{\cdot}10^{-5}$
64	1/2	13.8	$1.9{\cdot}10^{-4}$	14.3	$1.7{\cdot}10^{-4}$	16.4	$1.4{\cdot}10^{-4}$
	2/3	16.7	$2.0{\cdot}10^{-4}$	17.3	$1.9{\cdot}10^{-4}$	20.3	$1.3{\cdot}10^{-4}$
	3/4	18.2	$1.9{\cdot}10^{-4}$	18.9	$2.0{\cdot}10^{-4}$	23.0	$1.1{\cdot}10^{-4}$
	5/6	19.4	$2.1{\cdot}10^{-4}$	20.4	$1.9{\cdot}10^{-4}$	26.2	$5.8{\cdot}10^{-5}$
	7/8	20.2	$2.1{\cdot}10^{-4}$	21.3	$2.4{\cdot}10^{-4}$	28.6	$3.4{\cdot}10^{-5}$

Tab. 4. Simulated BER of DVB-T at the QEF C/N.

Rayleigh fading. We assume perfect channel knowledge, and soft Viterbi decoding with unquantized inputs and a trace-back length of $l_{\mathrm{TB}} = 128$.

According to [1], the performance should be measured at the quasi error-free (QEF) condition (not more than a single uncorrected error per hour). QEF corresponds to a BER after the RS decoder of $1{\times}10^{-11}$, and to a BER after the Viterbi decoder of $2{\times}10^{-4}$. Table 3 reports the values of C/N (in dB) for QEF, for different modulation sizes M and coding rates (CRs). The C/N values obtained using the MATLAB modulator and receiver are in good agreement with the C/N values provided by the DVB-T specifications [1]. Note that perfect agreement is virtually impossible, since the DVB-T specifications do not disclose all the decoding parameters, such as the trace-back length. Table 4 shows the BER after the Viterbi decoder, simulated at the QEF C/N value, for different DVB-T configurations and for different channels, such as AWGN, F1, and P1 [1]. The F1 multipath channel (static, line-of-sight) leads to Rice fading. Table 4 explains that the simulated BER is either close to the QEF BER of $2{\times}10^{-4}$ or lower, thus confirming that the decoding parameters have been selected properly.

5. Discussion

We discuss the impact of the proposed SDR DVB-T modulator, and we compare the proposed DVB-T software with DVB-T software designs available in the literature.

5.1 Impact

First, we explain the advantages of SDR-based DVB-T transmitters with respect to conventional DVB-T transmitters, and successively we list and discuss some useful applications of the proposed DVB-T software.

In the upcoming future, some TV channels located in the upper UHF band will be switched off, since that spectrum will be reallocated to cellular networks for extending their 4G (or upcoming 5G) services [31], [32]. This spectrum reallocation pushes the broadcasting players (operators and equipment companies) to move their technologies towards SDR-based ones. The reason for this change is twofold. First, the profitability of hardware-based broadcasting transmitters will be reduced, because there will be less market opportunities to catch. However, SDR transmitters, due to their easier re-programmability and lower development costs (at least in their digital segment), will allow to accommodate the market uncertainties. Second, broadcasting operators could use many low-power small-coverage transmitters co-located with cellular network base stations, instead of using few high-power large-coverage transmitters. In this case, SDR systems can be shared between broadcasting and cellular technologies.

Our software can be used for different applications related to DVB-T, such as signal analysis, interference estimation, power resource allocation, coverage issues, and so on. For instance, the designed software permits a statistical evaluation of the peak-to-average power ratio of true DVB-T signal waveforms. As a second example, adjacent channel interference estimation can be performed too, to assess the coexistence with nearby DVB-T (or LTE) channels [33]. In addition, power allocation algorithms can be exploited without changing the DVB-T receiver. Another possibility is the investigation of the effect of impulsive noise [34], non-Gaussian channels, and quantization errors [35], on the DVB-T received signal. Furthermore, by means of our SDR-based design, the DVB-T signal can first be received by a custom DVB-T receiver and then retransmitted to a single destination by a regenerative relay, nearby to the receiver. Coverage issues may be investigated by incorporating the designed software into a transmitting device for trial experiments. The proposed software could be used also jointly with cognitive radio systems that try to exploit the TV white spaces: for instance, in [36], the authors introduce a spectrum-sensing-based system where HDTV signals received via a satellite link can be redistributed in home environments over unoccupied DVB-T channels, using a DVB-T transmitter. Obviously, this mentioned list of possible applications is only exemplificative and non-exhaustive.

5.2 Comparison with Other DVB-T Software

Herein we detail the main differences between the proposed DVB-T software and the existing literature. We focus on those DVB-T designs that are closely related to our proposed design: about C++, we consider the solutions proposed in [9] and [19]; with reference to MATLAB designs, we compare with [19] and [23].

With respect to the Soft-DVB in [9], there are 4 main differences. First, our C++ code is publicly available (https://github.com/wishful-project/module_iris/tree/master/dvb-tx-iris/components/gpp/phy). Second, [9] is based on GNU Radio, while our C++ code is based on the Iris framework, which requires less memory. Third, [9] employs a hybrid Python/C++ language, while our software uses the C++ language only. Fourth, [9] produces RF signals with 7 MHz channel bandwidth only (due to throughput limitations of the USB-based interface towards the USRP in [9]), while our software can employ different channel bandwidths, including the 8 MHz one. With respect to the C++ DVB-T in [19], there are two main differences. First, the C++ code of [19] is not publicly available. Second, the C++ code of [19] uses vectorization and multithreading, while the designed C++ code uses the intrinsic parallelization provided by the Iris framework. Therefore, although both transmitters work in real time, our non-parallelized C++ code is simpler and more easily understandable than [19]: this facilitates code modification and code reuse by other researchers.

With respect to the MATLAB DVB-T of [19], there are three main differences. First, the designed MATLAB code is publicly available (at https://github.com/wishful-project/module_iris/tree/master/dvb-tx-iris/scripts/dvbt/MATLAB). Second, the designed code also includes a validated DVB-T modulator and (F1 and P1) channel generation functions. Third, although [19] includes synchronization, the designed MATLAB code has some improved features not present in [19], such as LLR-based soft-output demapping and FFT-based channel estimation. With respect to the MATLAB DVB-T modulator of [23], there are two main differences. First, [23] does not include frame adaptation, and hence pilots and TPS are not included. Second, in [23], OFDM is not implemented.

6. Conclusion

We have proposed two software implementations of DVB-T, using C++ and MATLAB, publicly available under the GNU license. The generated DVB-T signal has been validated by the equivalence between C++ and MATLAB outputs and by the correct reception by TV sets and USB dongles. When incorporated into the Iris SDR framework, the C++ DVB-T modulator works in real time. The proposed code can be useful for many applications and can be easily modified. Future work could include DVB-T2 [37], DVB-T2-Lite [38], and DVB-C2 [39].

Acknowledgments

The research leading to these results has received funding from the European Horizon 2020 Programme under grant agreement no. 645274 (WiSHFUL project, Open Call 1, DVB-TX-IRIS Extension). The authors thank Nicholas Kaminski, Maicon Kist, Alextian Liberato, Diarmuid Collins, Pedro Alvarez, and Francisco Paisana of TCD for all the help in using the Iris software and testbed. The authors also thank Gianluca Reali and Mauro Femminella for the USRP hardware. We also thank the anonymous reviewers for their constructive comments.

References

[1] DIGITAL VIDEO BROADCASTING (DVB). Framing structure, channel coding and modulation for digital terrestrial television,

ETSI EN 300 744 V1.6.1 (2009-01). ETSI.

[2] POGGIONI, M., RUGINI, L., BANELLI, P. DVB-T/H and T-DMB: Physical layer performance comparison in fast mobile channels. *IEEE Transactions on Broadcasting*, 2009, vol. 55, no. 4, p. 719–730. DOI: 10.1109/TBC.2009.2034418

[3] REIMERS, U. H. DVB - The family of international standards for digital video broadcasting. *Proceedings of the IEEE*, 2006, vol. 94, no. 1, p. 173–182. DOI: 10.1109/JPROC.2005.861004

[4] ULVERSØY, T. Software defined radio: Challenges and opportunities. *IEEE Communications Surveys and Tutorials*, 2010, vol. 12, no. 4, p. 531–550. DOI: 10.1109/SURV.2010.032910.00019

[5] PALKOVIC, M., RAGHAYAN, P., LI, M., et al. Future software-defined radio platforms and mapping flows. *IEEE Signal Processing Magazine*, 2010, vol. 27, no. 2, p. 22–33. DOI: 10.1109/MSP.2009.935386

[6] BLAKE, G., DRESLINSKI, R. G., MUDGE, T. A survey of multicore processors. *IEEE Signal Processing Magazine*, 2009, vol. 26, no. 6, p. 26–37. DOI: 10.1109/MSP.2009.934110

[7] KRATOCHVÍL, T., SLANINA, M. The DVB channel coding application using the DSP development board MDS TM-13 IREF. *Radioengineering*, 2004, vol. 13, no. 4, p. 14–17.

[8] IANCU, D., YE, H., GLOSSNER, J., et al. Software-only implementation of DBV-H. In *Proceedings of SPIE Multimedia on Mobile Devices*, 2008, 8 p. DOI: 10.1117/12.763800

[9] PELLEGRINI, V., BACCI, G., LUISE, M. Soft-DVB: A fully-software GNU Radio-based ETSI DVB-T modulator. In *Proceedings of the 5th Karlsruhe Workshop on Software Radios*, 2008.

[10] PELLEGRINI, V., LUISE, M. Fully software OFDM modulation in vehicular, highly time-variant channels: An implemented technology and its results. In *Proceedings of IEEE International Symposium on Wireless Communication Systems.* Tuscany (Italy), 2009. DOI: 10.1109/ISWCS.2009.5285237

[11] PELLEGRINI, V., DI DIO, M., ROSE, L., LUISE, M. On the computation/memory trade-off in software defined radios. In *Proceedings of IEEE Global Telecommunication Conference (GLOBECOM)*. Miami (FL, USA), 2010. DOI: 10.1109/GLOCOM.2010.5683494

[12] JIANG, Y., XU, W., GRASSMANN, C. Implementing a DVB-T/H receiver on a software-defined radio platform. *International Journal of Digital Multimedia Broadcasting*, 2009, 7 p. DOI: 10.1155/2009/937848

[13] SUGANO, H., MIYAMOTO, R., OKADA, M. Fully software-based real-time ISDB-T 1 segment receiver. In *Proceedings of IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. Nuremberg (Germany), 2011, 5 p. DOI: 10.1109/BMSB.2011.5954956

[14] YU, J.-C., SHIH, J.-Z., HSU, Y.-T., TSENG, S.-M. Reed-Solomon decoder optimization for PC-based DVB-T software radio receiver. In *Proceedings of IEEE International Conference on Consumer Electronics (ICCE)*. Las Vegas (USA), 2011, 2 p. DOI: 10.1109/ICCE.2011.5722645

[15] TSENG, S.-M., HSU, Y.-T., LIN, H.-K. Iterative channel decoding for PC-based software radio DVB-T receiver. *Wireless Personal Communications*, 2013, vol. 69, no. 1, p. 403–411. DOI: 10.1007/s11277-012-0580-z

[16] TSENG, S.-M., HSU, Y.-T., CHANG, Y.-Y., LEE, T.-C. Software baseband optimization except channel decoding for PC-based DVB-T software radio receiver. *Lecture Notes in Electrical Engineering*, 2014, vol. 260, p. 319–327. DOI: 10.1007/978-94-007-7262-5_37

[17] TSENG, S.-M., CHANG, T.-K., HSU, Y.-T. A/D USB dongle implementation for NB/PC-based software radio DVB-T receiver. In *Proceedings of IEEE International Conference on Advanced Technologies for Communication (ATC)*. Hanoi (Vietnam), 2012, 5 p. DOI: 10.1109/ATC.2012.6404278

[18] LEE, K.-H., HEO, S. W. GPU based software DVB-T receiver design. In *Proceedings of IEEE International Conference on Consumer Electronics (ICCE)*. Las Vegas (USA), 2013, 5 p. DOI: 10.1109/ICCE.2013.6487027

[19] BARUFFA, G., RUGINI, L., BANELLI, P. Design and validation of a software defined radio testbed for DVB-T transmission. *Radioengineering*, 2014, vol. 23, no. 1, p. 387–398.

[20] MACIEL, Y. P., AKAMINE, C., BEDICKS, JR. G., LOPES, P. B. ISDB-T_B transmission in software-defined radio. In *Proceedings of the 7th IEEE Latin-American Conference on Communications (LATINCOM)*. Arequipa, (Peru), 2015, 6 p. DOI: 10.1109/LATINCOM.2015.7430122

[21] CICHON, G., FETTWEIS, G., MOUSE: A shortcut from Matlab source to SIMD DSP assembly code. In *Proceedings of International Workshop on Embedded Computer Systems*. Samos (Greece), 2004, p. 159–167. DOI: 10.1007/978-3-540-27776-7_17

[22] VEJRAŽKA, F., KOVAR, P., ESKA, M., PURICER, P. Software navigation receivers for GNSS and DVB. *TransNav International Journal on Marine Navigation and Safety of Sea Transportation*, 2007, vol. 1, no. 2, p. 137–141.

[23] HÜTTL, A., KRATOCHVÍL, T. DVB-T channel coding implementation in MATLAB. In *Proceedings of MATLAB Conference.* Prague (Czech Republic), 2009.

[24] POLÁK, L., KRATOCHVÍL, T. Simulation of the DVB-H channel coding and transmission in MATLAB. In *Proceedings of IEEE International Conference Radioelektronika*. Brno (Czech Republic), 2010, 4 p. DOI: 10.1109/RADIOELEK.2010.5478591

[25] SUTTON, P., LOTZE, J., LAHLOU, H., et al. Iris: an architecture for cognitive radio networking testbeds. *IEEE Communications Magazine*, 2010, vol. 48, no. 9, p. 114–122. DOI: 10.1109/MCOM.2010.5560595

[26] DOYLE, L. E., SUTTON, P. D., NOLAN, K E., et al. Experiences from the Iris testbed in dynamic spectrum access and cognitive radio experimentation. In *Proceedings of IEEE Symposium on New Frontiers in Dynamic Spectrum (DySPAN)*. Singapore, 2010, 8 p. DOI: 10.1109/DYSPAN.2010.5457835

[27] FRIGO, M., JOHNSON, S. G. FFTW: An adaptive software architecture for the FFT. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Seattle (USA), 1998, p. 1381–1384. DOI: 10.1109/ICASSP.1998.681704

[28] MARTIN, K., HOFFMAN, B. An open source approach to developing software in a small organization. *IEEE Software*, 2007, vol. 24, no. 1, p. 46–53. DOI: 10.1109/MS.2007.5

[29] TOSATO, F., BISAGLIA, P. Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2. In *Proceedings of IEEE International Conference on Communication (ICC)*. New York (USA), 2002. DOI: 10.1109/ICC.2002.996940

[30] BARUFFA, G., RUGINI, L. Soft-output demapper with approximated LLR for DVB-T2 systems. In *Proc. IEEE Global Communication Conference (GLOBECOM)*. San Diego (USA), 2015. DOI: 10.1109/GLOCOM.2015.7417395

[31] GOMEZ-BARQUERO, D., WINSTON CALDWELL, M. Broadcast television spectrum incentive auctions in the U.S.: Trends, challenges, and opportunities. *IEEE Communications Magazine*, 2015, vol. 53, no. 7, p. 50–56. DOI: 10.1109/MCOM.2015.7158265

[32] ALA-FOSSI, M., BONET, M. Clearing the skies: European spectrum policy and future challenges of DTT in Finland and Spain. *International Journal of Digital Television*, 2016, vol. 7, no. 3, p. 363–377. DOI: 10.1386/jdtv.7.3.363_1

[33] BARUFFA, G., FEMMINELLA, M., MARIANI, F., REALI, G. Protection ratio and antenna separation for DVB-T/LTE coexistence issues. *IEEE Communications Letters*, 2013, vol. 17, no. 8, p. 1588–1591. DOI: 10.1109/LCOMM.2013.070113.130887

[34] BANELLI, P. Bayesian estimation of a Gaussian source in Middleton's class-A impulsive noise. *IEEE Signal Processing Letters*, 2013, vol. 20, no. 10, p. 956–959. DOI: 10.1109/LSP.2013.2274774

[35] RUGINI, L., BANELLI, P. On the equivalence of maximum SNR and MMSE estimation: applications to additive non-Gaussian channels and quantized observations. *IEEE Transactions on Signal Processing*, 2016, vol. 64, no. 23, p. 6190–6199. DOI: 10.1109/TSP.2016.2607152

[36] FADDA, M., MURRONI, M., POPESCU, V. An unlicensed indoor HDTV multi-vision system in the DTT bands. *IEEE Transactions on Broadcasting*, 2012, vol. 58, no. 3, p. 338–346. DOI: 10.1109/TBC.2012.2201559

[37] GRÖNROOS, S., NYBOM, K., BJÖRKVIST, J. Complexity analysis of software defined DVB-T2 physical layer. *Analog Integrated Circuits and Signal Processing*, 2011, vol. 69, no. 2-3, p. 131–142. DOI: 10.1007/s10470-011-9724-4

[38] SAMO, D. A., SLIMANI, M., BARUFFA, G., RUGINI, L. A performance study of DVB-T2 and DVB-T2-Lite for mobile reception. *Digital Signal Processing*, 2015, vol. 37, p. 35–42. DOI: 10.1016/j.dsp.2014.11.002

[39] HASSE, P., ROBERT, J. A software-based real-time DVB-C2 receiver. In *Proceedings of IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. Nuremberg (Germany), 2011. DOI: 10.1109/BMSB.2011.5954935

About the Authors ...

Giuseppe BARUFFA (corresponding author) was born in Perugia, Italy, in 1970. He received the Laurea degree in Electronic Engineering and the Ph.D. degree in Telecommunications from the University of Perugia in 1996 and 2001, respectively. In 2005, he visited Swiss Federal Polytechnic of Lausanne (EPFL), Switzerland. Since 2005, he has been an Assistant Professor of Telecommunications with the Department of Engineering at the University of Perugia. He is the author of about 60 scientific papers. His main research interests include digital television broadcasting techniques, joint source/channel coding for video, analysis of intermodulations in nonlinear systems, and GNSS /Cellular hybrid positioning techniques.

Luca RUGINI was born in Perugia, Italy, in 1975. He received the Laurea degree (cum laude) in Electronic Engineering in 2000 and the Ph. D. degree in Telecommunications in 2003 from the University of Perugia, Perugia, Italy. In 2007, he visited Delft University of Technology, Delft, The Netherlands. He is currently an Assistant Professor with the Department of Engineering at the University of Perugia. His research interests lie in the area of signal processing for communications, with emphasis on multicarrier and cognitive radio techniques. He serves as an Associate Editor of the IEEE Signal Processing Letters, Digital Signal Processing, and the EURASIP Journal on Advances in Signal Processing.

Fabrizio FRESCURA is an Assistant Professor at the Department of Engineering of the University of Perugia since January 2005, where he is the lecturer of digital signal processing courses. He received the Laurea degree (cum laude) in Electronic Engineering from the University of Perugia in 1992, and the Ph.D. degree in Telecommunications in 1997. He was the advising professor of the winning team of the Texas Instruments 1997 DSP Solution Challenge. From 1999 to 2000 he has been working in Texas Instruments France. In 2000, he was co-founder of the university spin-off company Digilab2000. Since 2001 he coordinates the research activity of DSPLAB of University of Perugia. The main activities are related to DSP for Image/Video processing and transmission, and DSP implementation. In 2004, he was co-founder of the university spin-off company Mediatech. Since 2002 he joined the standardization committee ISO/IEC JTC 1/SC 29/WG 1. He has been co-editor and co-chair of the JPWL (JPEG 2000 Wireless) AHG (Ad Hoc Group) and co-editor of D-Cinema AHG. The scientific activity is proven by about 80 papers published on peer-reviewed international journals, proceedings of international conferences, and contributions to international organization standards.

Paolo BANELLI received the Laurea degree (cum laude) in Electronics Engineering and the Ph.D. degree in Telecommunications from the University of Perugia, Italy, in 1993 and 1998, respectively. In 2005, he was appointed as Associate Professor in the Department of Electronic and Information Engineering, University of Perugia, where he has been an Assistant Professor since 1998. In 2001, he joined the SpinComm group, as a Visiting Researcher, in the Electrical and Computer Engineering Department, University of Minnesota, Minneapolis. His research interests include signal processing for wireless communications, with emphasis on multicarrier transmissions, signal processing for biomedical applications, spectrum sensing for cognitive radio, waveform design for 5G communications, and graph signal processing. He was a Member (2011–2013) of the IEEE Signal Processing Society's Signal Processing for Communications and Networking Technical Committee. In 2009, he was a General Co-chair of the IEEE International Symposium on Signal Processing Advances for Wireless Communications. He currently serves as an Associate Editor of the IEEE Transactions on Signal Processing and the EURASIP Journal on Advances in Signal Processing.