A SOFTWARE DEFINED RADIO APPROACH FOR DIGITAL TELEVISION ISDB-T TRANSMITTERS

André L. G. Reis (University of Campinas, Campinas, São Paulo, Brazil; andre.lgr@gmail.com); André F. B. Selva (University of Campinas, Campinas, São Paulo, Brazil; andrefselva@gmail.com); Karlo G. Lenzi (CPqD, Campinas, São Paulo, Brazil; lenzi@decom.fee.unicamp.br); Luis G. P. Meloni (University of Campinas, Campinas, São Paulo, Brazil; meloni@decom.fee.unicamp.br); Sílvio E. Barbin (University of São Paulo, São Paulo, São Paulo, Brazil; barbin@usp.br)

ABSTRACT

Although most countries already have Digital Television (DTV) available, there are several others migrating their infra-structure from analog to digital. One of these countries is Brazil, which intends to accomplish a full switch up to 2016. This initiative is seen as of extreme importance by local authorities since Brazil will host the FIFA World Cup in 2014 and the Olympics Games in 2016, events that will be seen by millions of viewers around the globe. The Brazilian DTV is based on the Japanese ISDB-T standard for the modulation scheme, which is an OFDM system capable of supporting services from mobile to full-definition. Since the relevance of the subject, this paper presents a software defined radio approach to implement a full ISDB-T transmitter using GNU Radio and USRP, covering a complete study from modeling to implementation. This papers aims not only to discuss and clarify several misinterpretations of the ISDB-T standard, but also to present an efficient implementation on a SDR architecture, serving as reference to new developers and as a roadmap for manufactures.

1. INTRODUCTION

The Integrated Services Digital Broadcasting – Terrestrial (ISDB-T) [1] was developed to support different services from mobile to full-definition. Occupying a channel of 6 MHz, it is a versatile and robust system, which integrates several protections against errors that can occur in the transmission.

The most common implementation of the ISDB-T transmitter is based on specialized hardware to perform the steps required to modulate the signal. Our goal here, however, is to present a Software Defined Radio [5] approach to implement such transmitter. SDR platforms are already widely used in scientific field, allowing several researches to work on areas like cognitive radios [10], OFDM modulation [11], among others. A research group in Italy has already successfully implemented the DVB (Digital Video Broadcasting) transmitter using SDR [13].

The main reason to implement a ISDB-T transmitter using a SDR platform is the flexibility offered by the latter. Characteristics of the system can be easily changed and upgrades can be performed in short time, without hardware modification.

This paper describes the ISDB-T Standard, trying to clarify some of its aspects and particularities. Also, it briefly discusses the implementation of the transmitter using Matlab and GNU Radio.

2. ISDB-T STANDARD OVERVIEW

The ISDB-T is an OFDM (Orthogonal Frequency Division Multiplexing) system with 13 segments per frame organized in up to 3 hierarchical layers. It supports 3 operation modes, which defines how many data carriers will be used in each OFDM segment.

The input of the ISDB-T modulator is a BTS (Broadcast Transport Stream) file which contains the TSPs (Transport Stream Packet) used to transport the audio and video data, among other informations. Fig. 1 below shows the processing chain of the modulator.

First we have a shortened Reed-Solomon (204,188) encoder, which includes parity bytes in TSPs, followed by the hierarchical divider, which is responsible to distribute TSPs across each hierarchical layer processing chain. Each hierarchical layer is composed by a sequence of signal processing chain responsible to adapt the TSPs to transmission format. This includes an energy dispersal, bit and byte interleavers, convolutional encoder and mapper.

The final steps include hierarchical layers combiner, OFDM frame adaptation (allocation of data carriers, pilots and Transmission and Multiplexing Configuration Control), IFFT and guard-interval addition. When all these tasks are completed, the signal is ready to be transmitted. It is worth noticing that a multiplex frame (which comes from BTS) is

aligned with a OFDM frame. All these tasks will be detailed in following sections.

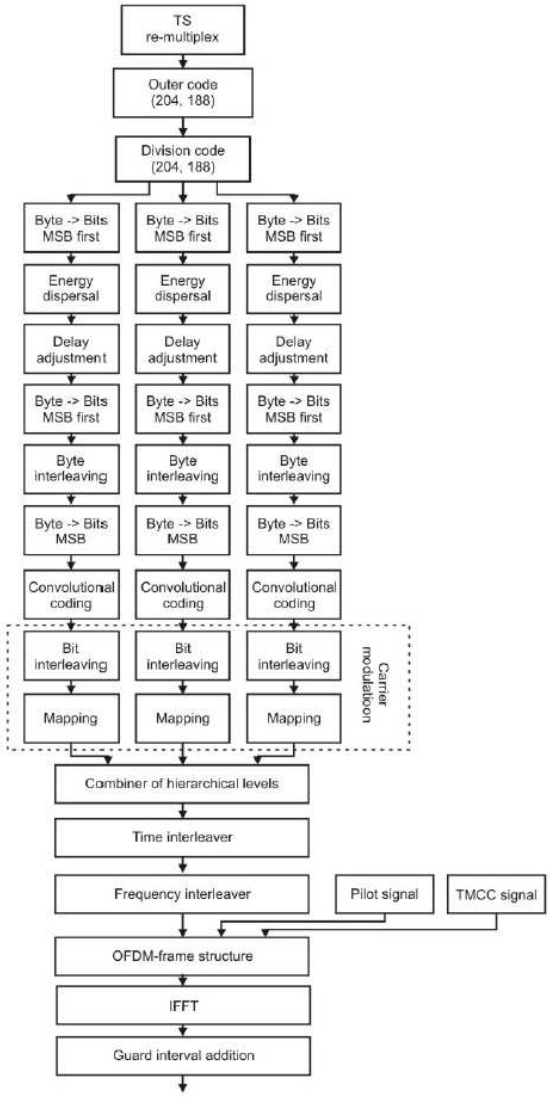


Fig. 1. Block diagram of channel coding [1].

The ISDB-T Standard defines several parameters that are crucial for good understanding of the system. Below we clarify some of those definitions presented in [1].

- 1 OFDM Frame = 204 OFDM Symbols
- 1 OFDM Symbol = 13 OFDM Segments
- 1 OFDM Segment = 1 Data Segment + Pilots + TMCC + AC
- 1 Data Segment = *n* Data Carriers (*n* depends on the operation mode, namely 2k, 4k and 8k)

To illustrate those parameters, in Fig. 2 we see in Frequency Axis a OFDM symbol composed by sub-carriers (Data Carriers + Pilots + TMCC + AC) and in Time Axis we see the propagation of OFDM symbols (204 of these will constitute a OFDM frame).

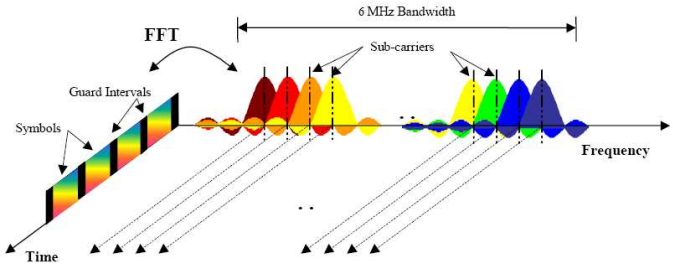


Fig. 2. OFDM Symbol in frequency and time domain [3].

2.1. Reed-Solomon

The Reed-Solomon algorithm is an error-corrector code developed by Irving S. Reed and Gustave Solomon in 1960 [2]. It is capable to correct random errors and burst noise. Due to its code properties, which makes it very robust to errors,, it is largely used in storage medias like DVD and CD, mobile communications, etc.

In the ISDB-T, a shortened version of RS encoder is used, namely RS(204,188). This is obtained by adding 51 null bytes in the beginning of the TSP and applying it to RS(255,239). After encoding, the inserted zeros are removed from the data stream. In the end of this process, the TSP will have 188 data bytes and 16 parity bytes, composing the 204 bytes BTS packet.

2.2. Hierarchical Divider

The BTS file contains all packets of hierarchical layers A, B and C. The Hierarchical Divider is responsible to redirect each TSP to its processing chain, to discard null packets generally used to maintaining a system's constant transmit rate, to store the IIP (ISDB-T Information Packet), which will configure the parameters used to modulate the signal, and to shift the sync byte (0x47) from beginning to end of the packets. More details of its operation are available in [1] and [12].

2.3. Energy Dispersal

This block randomizes the data within the TSP, in order to distribute its energy equally along the packet, reducing the inter symbol interference caused by transmitting repetitive information. Fig. 3 shows the circuit responsible to perform this task. It is a simple shift register with a XOR operation. This polynomial is also used in other OFDM transmission standards such as WiMAX.

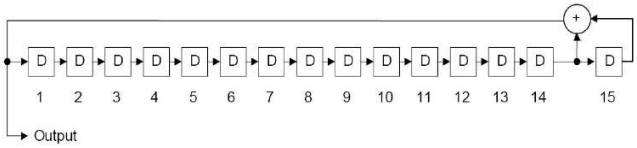


Fig. 3. PRSB-Generating Polynomial and Circuit [1].

It is worth emphasizing that the sync byte (0x47) should not be randomized along with the data bytes.

2.4. Byte Interleaving

This block is responsible of interleaving the data, so that if frequency selective errors occur in the channel, they do not corrupt the data block since it's scattered . The interleaver is composed by 12 paths with variable delays. Fig. 4 shows its scheme.

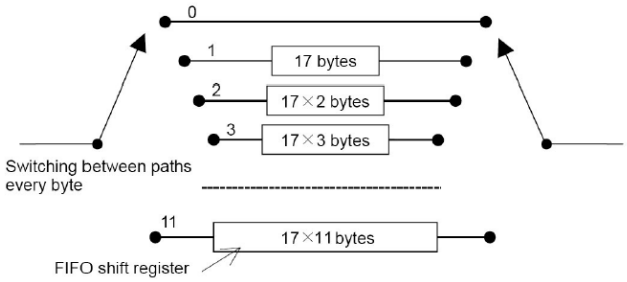


Fig. 4. Byte Interleaving circuit [1].

A delay adjustment is required here, in order to compensate the delay introduced by the circuit above. Details on the amount of delay adjustment required are available in [1].

2.5. Inner Code

This is a very common convolutional encoder with constraint length k = 7. This inner code allows multiple coding rates through bit puncturing patterns. Fig. 5 shows its circuit.

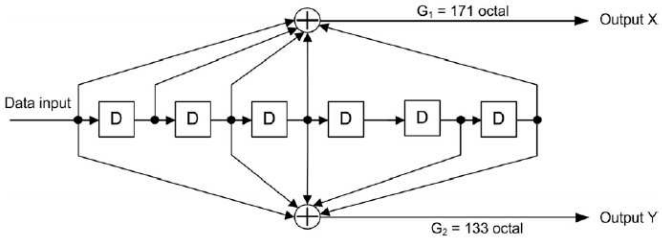


Fig. 5. Coding circuit of a convolutional code with constrain length *k* of 7 and a coding rate of ½ [1].

The puncturing pattern depends on the coding rate used. Table I shows puncturing pattern that should be performed to achieve the coding rates.

TABLE I. INNER-CODE CODING RATES AND TRANSMISSION-SIGNAL SEQUENCE [1]

Coding rate	Puncturing pattern	Transmission-signal sequence
1/2	X : 1 Y : 1	X1, Y1
2/3	X : 1 0 Y : 1 1	X1, Y1, Y2
3/4	X : 1 0 1 Y : 1 1 0	X1, Y1, Y2, X3
5/6	X : 1 0 1 0 1 Y : 1 1 0 1 0	X1, Y1, Y2, X3, Y4, X5
7/8	X : 1 0 0 0 1 0 1 Y : 1 1 1 1 0 1 0	X1, Y1, Y2, Y3, Y4, X5, Y6, X7

2.6. Bit Interleaving

The same concept used in Byte Interleaving is used here. The main difference is that now we should parallelize the data depending on how many bits each modulation scheme uses. For each modulation scheme we will have a different interleaver in terms of shift registers and paths. Fig. 6 shows the bit interleaving for 64QAM, which uses 6 bits per carrier.

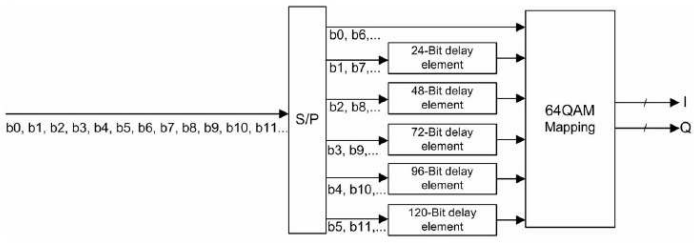


Fig. 6. 64QAM Modulation system diagram [1].

It is worth noticing that it is required a delay adjustment [1], as in Byte Interleaving.

2.7. Carrier Modulation

The ISDB-T supports DQPSK, QPSK, 16QAM and 64QAM. After the bit interleaving, the data is mapped in a point of the constellation. As an example, Fig. 7 shows the constellation of 16QAM modulation scheme.

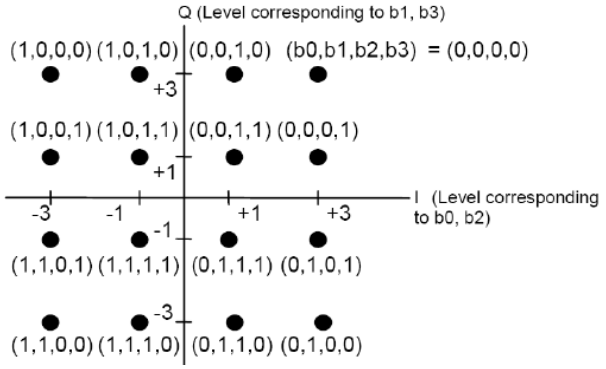


Fig. 7. Constellation of 16QAM [1].

After mapping the data in constellation, a modulation level normalization is required. Table II shows the normalization factor.

TABLE II. MODULATION LEVEL NORMALIZATION [1]

Carrier modulation scheme	Normalization factor
DQPSK and QPSK	$Z/\sqrt{2}$
16QAM	$Z/\sqrt{10}$
64QAM	$Z/\sqrt{42}$

2.8. Data segment configuration

This step just rearranges data into data segments to make easier the allocation of data carriers in OFDM symbols. Fig. 8 below shows the organization of data carriers.

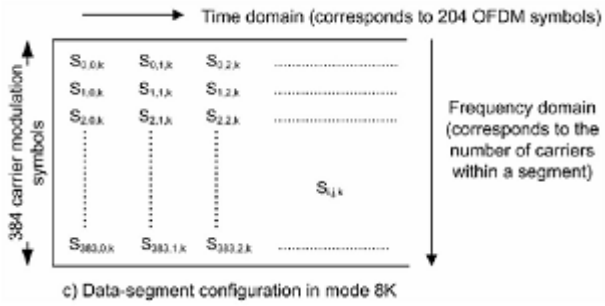


Fig. 8. Data-segment configuration [1].

Note that 96, 192 and 384 are the number of data carriers per OFDM segment used in modes 1, 2 and 3, respectively.

2.9. Hierarchical combiner

The Hierarchical combiner puts all 13 segments together in order to gather enough data to build a OFDM symbol. The sequence of segments is shown by Fig. 9.

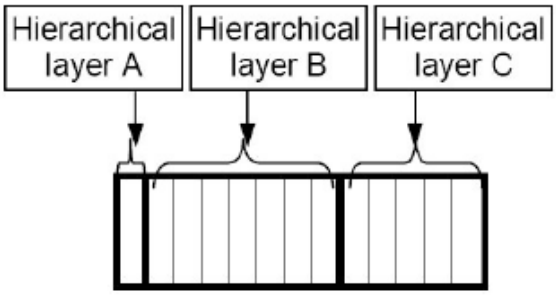


Fig. 9. Segments order [1].

2.10. Time Interleaving

The Time Interleaver operates separately on each of 13 data segments. Fig. 10 and Fig. 11 show the schematic of this block.

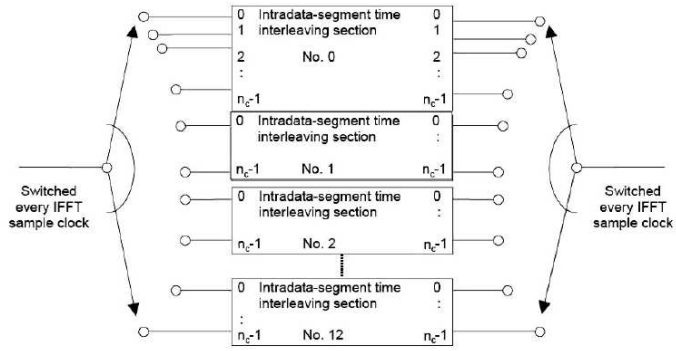


Fig. 10. Time Interleaver.

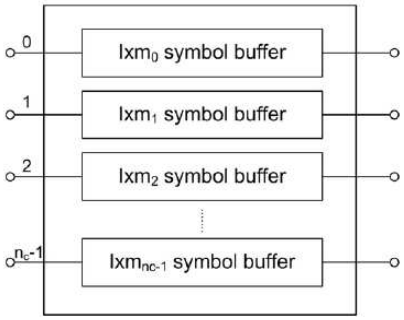


Fig. 11. Configuration of intradata-segment [1].

The parameter I corresponds to the interleave length and m_i is given by

$$m_i = (5i) \bmod 96, \qquad (1)$$

where i is the carrier number, that is, each carrier will have paths with different delays. Also, this step requires a delay adjustment [1], as Bit and Byte Interleaving.

2.11. Frequency Interleaving

This module is divided in three steps: inter-segment interleaving, intra-segment carrier rotation and intra-segment carrier randomizing.

2.11.1. Inter-segment interleaving

This step is performed only on hierarchical layers B and C, since layer A has only one segment. The ISDB-T Standard shows a confusing figure to describe this interleaving, but it is simply a matrix interleaver [9]. Suppose we have *k* segments and *n* data carriers per segment. The interleaving process consists in writing the segments column-wise in a *k* x *n* matrix and reading line-wise, so each line of the matrix is the interleaved segment. Fig. 12. illustrates this process.

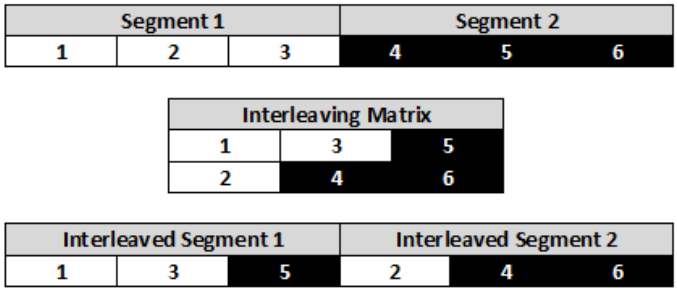
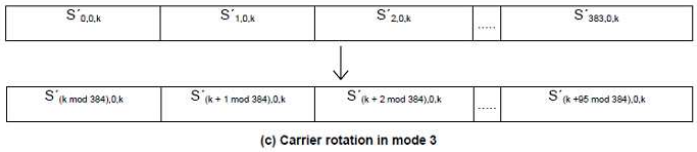


Fig. 12. Matrix Interleaver fictional example for 2 segments with 3 data carriers each.

2.11.2. Intra-segment carrier rotation

The carrier rotation, as its name says, rotates the carriers inside the segment according to the segment number. Fig. 13 shows the process for modes 1, 2 and 3.



NOTE: The symbol $S'_{I,j,k}$ represents the carrier symbol of the k_{th} segment following inter-segment interleaving.

Fig. 13. Carrier rotation [1].

2.11.3. Intra-segment carrier randomizing

This step just randomizes the carriers' position according to pre-defined tables available in ISDB-T Standard [1].

2.12. OFDM Frame Structure

The OFDM frame structure is shown by Fig. 13.

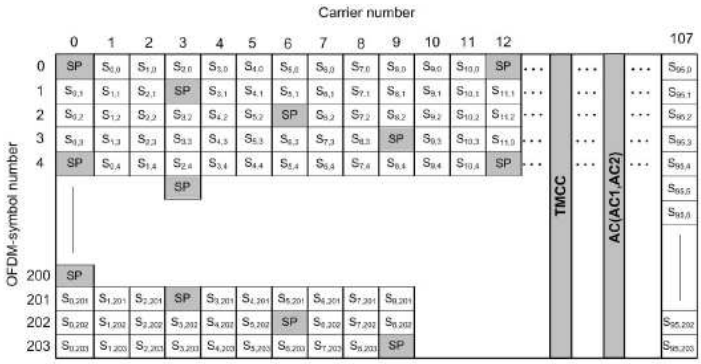


Fig. 13. Configuration of the OFDM frame for synchronous modulation at mode 1 [1].

It is important to notice that Fig. 13. above extends to the right 12 more times, so we have 13 segments per symbol.

It is also necessary to organize the segment as showed in Fig. 14.

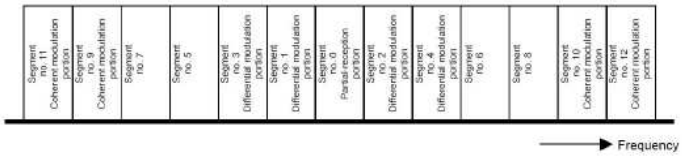


Fig. 14. OFDM segment numbers on the transmission spectrum [1].

To make up the entire transmission spectrum, a continuous carrier with its phase defined by W_i is allocated in right-hand end of the band. This means, each OFDM symbol will have an extra continual pilot in its structure.

2.13. Pilots, TMCC and AC

The SPs (Scattered Pilot), TMCC (Transmission and Multiplexing Configuration Control) and AC (Auxiliary Channel) are dependent on the output W_i of a PRBS-generating circuit, because it defines the modulation of these signals. Fig. 15. shows the circuit that generates de W_i and Table III shows the point on constellation according to W_i.

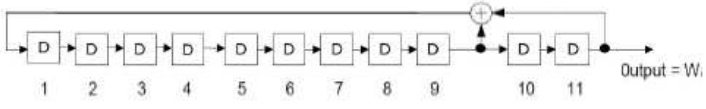


Fig. 15. PRSB-generating circuit [1].

TABLE III. W_i AND MODULATING SIGNAL [1]

W_i value	Modulating-signal amplitude (I,Q)
1	(- 4/3, 0)
0	(+ 4/3, 0)

In short, the SPs are used by the receiver to synchronize the signal, the TMCC contains all parameter necessary to configure the receiver and AC shall be used to transmit additional information on modulating signal-transmission control.

A SP is inserted every 12 carriers in OFDM segments direction. Then its index is shifted 3 positions for each new OFDM symbol. TMCC and AC carriers are fixed and defined in tables available in [1]. More details can be found in the ISDB-T Standard [1].

2.14. IFFT and Guard Interval

Once all processing tasks are completed, it is time to apply the IFFT (Inverse Fast Fourier Transform) to the signal. The configuration of the IFFT is showed in Table IV.

TABLE IV. IFFT PARAMETERS

Sample Frequency	8126984 Hz
Number of points for mode 1	2048
Number of points for mode 2	4096
Number of points for mode 3	8192

It is worth emphasizing that it is necessary to use the zero padding technique [8] in the symbol before applying the IFFT. The zeros should be inserted in the middle of the symbol, so that the total data is the same length of the number of points of the IFFT.

After the IFFT, it is necessary to insert the guard interval to the signal. This step consists in adding to the beginning of the effective symbol a part of its end. Fig. 16 illustrates this process. At this time, the signal is ready to be transmitted.

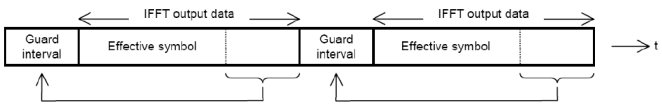


Fig. 16. Guard interval insertion [1].

3. SOFTWARE DEFINED RADIO

The main idea of a SDR (Software Defined Radio) is to transfer the tasks performed by hardware to software. By doing so, the implementation of a communication system becomes more flexible, allowing a radio to chance its operation characteristics during run time, such as modulation, bandwidth, output frequency, etc. Besides flexibility, SDRs have many other advantages over traditional radio implementations. They are more immune to temperature changes and component aging, since they transfer the processing task to digital domain, no longer depending on the precision of analog components. They also offer all advantages that a software environment provides to the development process, like simulation tools and error correction. All of these advantages can reduce time-to-market, since the same hardware platform can be used to implement different radios, just by replacing the software that controls the radio.

By transferring most part of processing task to software, the complexity of modern radios hardware is reduced, limiting it just to the RF front-end implementation. This also implies in high integration, given that many passive and active elements of the radio, once responsible to signal processing and modulation, are eliminated through the use of a processor. All these factors have an effect on the product final cost, which can be reduced.

There are several solutions in the market to implement a SDR system. In this paper, however, we will present a solution propose by Matt Ettus, the USRP (Universal Software Radio Peripheral) [4].

Composed by a motherboard, responsible for the base-band processing and by many RF daughterboards cards, which can be chosen according to the application's needs, the USRP abstracts the system RF front-end. A Ethernet interface establishes the communication between the PC and the USRP radio, allowing data exchange in a full-duplex manner [5].

GNU Radio is an open-source toolbox kit that provides a development environment and some processing blocks that can be used to create software defined radios. The software provides full integration with USRP boards [6]. It provides around a hundred signal-processing blocks and allows developers to design new ones using C++ as programming language. It also contains specific drives to communicate with the USRP.

To implement a ISDB-T modulator, though, it is necessary to create new blocks, since it has some specific processing steps that are not implemented in this package. The creation of blocks is not the scope of this paper, however.

GNU Radio applications are developed using the Python language, where the connection between blocks are made. The blocks are built in C++, for performance issues. A

graphical user interface, called GNU Radio Companion (GRC), is also provided. GRC could be used to simplify the system design, just like Simulink does with projects developed on Matlab.

4. MODELING AND IMPLEMENTATION

A model of the ISDB-T modulator was implemented in Matlab [7] as a reference for the SDR GNU Radio development. Each block of the system was implemented first as a Matlab Function and then, as a GNU Radio signal processing block. Comparisons between GNU Radio and the simulation outputs were made to validate each processing block.

In the simulation, the parallel processing of the three hierarchical layers were saved as states. Each component, when necessary, has a workspace to store its last state, so it is possible to reuse the same components in all 3 hierarchical layers without having data overwrite problem.

The simulator works with one OFDM frame for each iteration, once the multiplex frames that come from BTS are aligned with the OFDM frames. All configuration parameters of the modulator are automatically acquired from IIP (ISDB-T Information Packet) available in the BTS data stream. Each multiplex frame has one of these packets.

Concerning the implementation in GNU Radio, it was necessary to make an in-depth study of how to write blocks in this platform, since the open-source library does not offer all of the processing blocks needed to implement a ISDB-T modulator. Although this is not the scope of this paper, we will present part of the code of the Energy Dispersal in Matlab and GNU Radio (that is, in C++) in order to establish some comparisons between them.

```
% MATLAB CODE
function [tsp_ed] =
isdbt_energy_dispersal(tsp_rs, state ,
reset)

length_tsp_rs = length(tsp_rs) - 1; % The
sync. byte should not be randomized
tsp_syncByte = tsp_rs(end); % Sync Byte
is the last one of the input vector
% Convert data in a stream of bits
...
size  = length(data_in);

if reset == 1, % This value should be
initialized every OFDM frame.
    prbs_state = [1 0 0 1 0 1 0 1 0 0 0 0
0 0 0]; % Initial value of PRBS-gen
else
    load(state);
end
```

```
% randomize
for k=1:size
    fdB   =
bitxor(prbs_state(14),prbs_state(15)); %
calculate the feedback bit
    prbs_state = [fdB prbs_state(1:end-
1)]; % shift right and insert the fd bit
    data_out(k) = bitxor(data_in(k),fdB);
end

% rearrange bit stream in bytes
...
save(state,'prbs_state');
```

```
// C++ CODE
...
int
gr_isdbt_energy_dispersal_ff::work (int
noutput_items,
            gr_vector_const_void_star
&input_items,
            gr_vector_void_star
&output_items)
{
  const int *in = (const int *)
input_items[0];
  int *out = (int *) output_items[0];
  int x = 1000;
  for (int i = 0; i < noutput_items;
i++){
    cont_sinc++;
    if
((cont_sinc>INIBYTESINC)&&(cont_sinc<FIM
BYTESINC)) {
        out[i]=in[i];
        continue;
    }
    else if (cont_sinc==FIMBYTESINC) {
        out[i]=in[i];
        cont_sinc=0;
        continue;
    }
    else {
        out[i] = in[i] ^ prbs(v);
        shifter(v,1,'R');
    }
  }
  return noutput_items;
}
```

Analyzing both codes we see that they do not differ much. Except by syntax differences (Matlab Script and C++), the C++ code seems to be a direct transcription of the Matlab code. Though, in GNU Radio we need to care about input and output size precisely, data rate, etc, what makes the

porting process more complicated than expected. Despite those aspects, the simulation is still a good reference for implementation.

5. DISCUSSIONS

We have seen that the performance of the implementation in GNU Radio is far better than simulations. Maybe some improvements will have to be performed in order to achieve the real-time constraints of ISDB-T system.

Our next steps are toward concluding the implementation in GNU Radio. We are at the final stages of porting the code and then we have to integrate the blocks and perform tests in order to validate the implementation against the model.

6. FUTURE WORK

Currently, we are optimizing our ISDB-T simulator, using techniques such as parallel computing, GPU programming and hybrid implementation (C and Matlab), since the processing of one BTS for a full-seg transmission involves a great deal of data, which in turns make the simulation very slow. At the moment, the simulation time needed to generate 1 frame is around 7 minutes. It began taking 35 minutes. We believe that is possible to reduce the simulation time to 1.

We are also working on the conversion of the model, implemented through scripts to a Simulink model, since Mathworks recently offered support to communicate with USRPs via a specific driver, making possible to use the original model to generate data to the USRP, offering an alternative to the GNU Radio approach. For this reason, it is important to improve the simulation performance so it can emulate a real-time system for the ISDB-T transmission.

In terms of the SDR, it would be interesting to study some way to embed the ISDB-T transmitter within the USRP, so that the PC would be no longer necessary to the system, creating then a stand-alone application that still integrates a flexible hardware. There are indeed in the market some SDR platforms that incorporate a microprocessor, so this idea could become more feasible with such hardware.

7. CONCLUSION

This paper aimed not only to clarify some aspects of the ISDB-T Standard that can be confusing and lead to misinterpretations, but also to briefly discuss the implementation of a ISDB-T transmitter in GNU Radio and Matlab.

Offering great flexibility, SDR platforms show themselves as interesting solution to implement communication systems such as DVB [13] and ISDB-T.

8. REFERENCES

[1] "Digital terrestrial television – transmission system", Associação Brasileira de Normas Técnicas (ABNT), ABNT NBR 15601:2007, Dec. 2007.

[2] I. S. Reed and G. Solomon, "Polynomial code over certain fields", J. Soc. Ind. Appl. Math., 8:300-304, June 1960.

[3] HSCTechnicalWiki, "Orthogonal Frequency Division Multiplexing". [Online]. Available: http://wiki.hsc.com/wiki/Main/OFDM

[4] M. Ettus, "Universal Software Radio Peripheral". [Online]. Available: http://www.ettus.com.

[5] A. F. B. Selva, A. L. G. Reis, R. U. Labsch, K. G. Lenzi, L. G. P. Meloni and S. E. Barbin, "A software-defined radio approach: GNU Radio", 10th International Information and Telecommunication Technologies Conference, 2011, São José, SC. Proceedings of the 10th International Information and Telecommunication Technologies Conference, 2011. v. 1. p. 48-52.

[6] GNU Radio. [Online]. Available: http://gnuradio.org.

[7] Mathworks Matlab. [Online]. Available: http://www.mathworks.com/products/matlab/

[8] R. V. Van Nee and R. Prasad, "OFDM for Wireless Multimedia Communications Norwood", MA: Artech House, Jan. 2000.

[9] M. C. Paiva, "Uma implementação em software do subsistema de transmissão do padrão ISDB-TB". Master Thesis, INATEL, 2010.

[10] Z. Yan, Z. Ma, H. Cao, G. Li, and W. Wang, "Spectrum sensing, access and coexistence testbed for cognitive radio using USRP", 4th IEEE International Conference on Circuits and Systems for Communications, 2008. ICCSC 2008, 26-28 May 2008.

[11] A. Marwanto, M. A. Sarijari, N. Fisal, S. K. S. Yusof and R. A. Rashid, "Experimental study of OFDM implementation utilizing GNU Radio and USRP - SDR," Communications (MICC), 2009 IEEE 9th Malaysia International Conference, 15-17 Dec. 2009.

[12] "Digital terrestrial television – Operational guideline part 1: transmission system – guideline for ABNT NBR 15601:2007 implementation", Associação Brasileira de Normas Técnicas (ABNT), ABNT NBR 15608-1:2008, Aug. 2008.

[13] V. Pellegrini, G. Bacci, and M. Luise, "Soft-DVB, a Fully Software, GNURadio Based ETSI DVB-T Modulator," in 5th Karlsruhe Workshop on Software Radios, 2008.