Rapid Prototyping for Evaluating Vehicular Communications

Tiago M. Fernández-Caramés, Miguel González-López, Carlos J. Escudero and Luis Castedo
University of A Coruña
Spain

1. Introduction

During the last years, wireless communications between moving vehicles (Vehicle-To-Vehicle, VTV or V2V) and from vehicles to infrastructure (V2I or Roadside-to-Vehicle, RTV) have received a great deal of attention. Vehicular safety (collision prevention systems, accident warnings...) as well as payment and infotainment applications (automatic payment, mobile internet access at high vehicular speeds, traffic jam avoidance...) are increasingly demanded by the automotive industry in their way towards deploying ITS (Intelligent Transport Systems).

Several wireless communication standards targeted to vehicular scenarios have been proposed. Vehicular safety applications require fast exchange of messages in order to obtain a swift reaction from the car or the driver in dangerous situations, such as a sudden slowdown or when two cars approach an intersection. Due to the necessary quick response, transceivers have to send short packets and, therefore, small bandwidths are demanded. Among the different wireless standards, IEEE 802.11p (IEEE, 2010) is the best positioned to act as the reference standard for the PHYsical (PHY) and Medium Access Control (MAC) layer of vehicular communications for safety applications in the near future.

For non-safety vehicular applications, the discussion about which is the most suitable wireless access standard remains an open issue. The most cited candidates are the WiFi standards IEEE 802.11a/b/g and IEEE 802.16e (Mobile WiMAX). Most likely, vehicular communications will take place in the 5 GHz band since both US and European authorities have reserved spectrum for ITS at 5.9 GHz. Due to this, the final candidates might be reduced to IEEE 802.11p, IEEE 802.11a and IEEE 802.16e. Other wireless communication standards have been proposed for use in vehicular environments, such as HSDPA (High-Speed Downlink Packet Access), IEEE 802.20 (iBurst) or EDGE Evolution, but the peak data rates they offer for broadband communications (14.4 Mbit/s, 16 Mbit/s and 1 Mbit/s) are lower than those theoretically provided by IEEE 802.11p, IEEE 802.11a or IEEE 802.16e (27 Mbit/s, 54 Mbit/s and 39.9 Mbit/s). Also, there are several recently developed standards whose performance in vehicular scenarios has yet to be assessed and whose study will constitute an interesting topic for further research. Such new standards are expected to offer better global performance, but they are either in earlier development stages (e.g. LTE, IEEE 802.16m) or have not been explicitly designed for vehicular applications (e.g. IEEE 802.11n).

One of the most appropriate ways to evaluate the performance of transceivers compliant with a standard consists in using a testbed together with a channel emulator. Testbeds have

been previously used to assess different wireless communication standards (for instance, IEEE 802.11a (Angelakis et al., 2008), IEEE 802.11n (Nieto et al., 2006), IEEE 802.16e (Hu et al., 2007) and IEEE 802.11p (Fernández-Caramés, 2008)), whilst several channel emulators have been implemented to measure transceiver performance in realistic situations, like those developed for IEEE 802.11n (Dasatti et al., 2005), Dedicated Short-Range Communications (DSRC) (Faseth et al., 2010) or for UHF RFID systems (Arthaber & Schuberth, 2009).
In this book chapter we present a performance evaluation system made up of a software testbed and a flexible, low-cost, FPGA-based vehicular channel emulator. We detail the way we employed rapid-prototyping techniques for building both the testbed and the channel emulator. In order to decrease the development time required we decided to use MATLAB® and Simulink® for implementing three different transceivers compliant with the standards IEEE 802.16e, IEEE 802.11p and IEEE 802.11a. The vehicular channel emulator was implemented using another rapid-prototyping tool: Xilinx® System Generator ®. We obtained performance measurements over representative situations of V2V and V2I communications carried out by software-hardware co-simulation: MATLAB/Simulink software transceivers ran on a PC while the vehicular channel emulator was running on an external FPGA.
One of the best ways to increase the transmission capacity with respect to Single-Input Single-Output (SISO) systems (i.e. systems that use one transmit and one receive antenna), consists in building systems that use multiple antennas in transmission (known as MISO (Multiple-Input Single-Output) systems), reception (SIMO (Single-Input Multiple-Output) systems) or in both transmission and reception (MIMO (Multiple-Input multiple-Output) systems) (Foschini, 1998; Telatar, 1999).
Our previous work (Fernández-Caramés, 2010) focused on the performance in vehicular scenarios of IEEE 802.11p using SISO transceivers. As a novelty, we detail herein how we have updated the whole system with the aim of carrying out performance comparisons for multiple-antenna transceivers. We also explain the different optimizations we have performed during the design process of our MIMO channel emulator to deal with the fact that the FPGA computation resources are limited. Finally, we show the obtained performance evaluation results for SISO, SIMO and MIMO transceivers.
The rest of this chapter is organized as follows. Section 2 describes the three wireless standards considered as well as the corresponding transceivers we implemented. Section 3 details the design process of the FPGA-based vehicular channel emulator. Section 4 presents the upgrade of our system to MIMO. Finally, Section 5 details the experiments performed whereas Section 6 is devoted to the conclusions.

2. Wireless standards and transceivers: IEEE 802.11a, IEEE 802.11p and IEEE 802.16e

2.1 IEEE 802.16e (Mobile WiMAX)

We focused on the PHY layer referred to in the IEEE 802.16e standard as WirelessMAN-OFDMA. Fig. 1 depicts the block diagram of the evaluation system, which shows that Mobile WiMAX has been defined in a similar way to other broadband OFDMA (Orthogonal Frequency-Division Multiple Access) communication systems.
Among the different Mobile WiMAX working modes, in our experiments the transceiver operates in a mode called Downlink PUSC (Partial Usage of Subcarriers). In this mode, the 512 subcarriers are divided into 360 subcarriers for data, 60 for pilots and 92 for the guards and the DC. Each fourteen adjacent subcarriers over two OFDMA symbols constitute a cluster or resource block (24 subcarriers for data and 4 for pilots). Furthermore, each OFDMA symbol

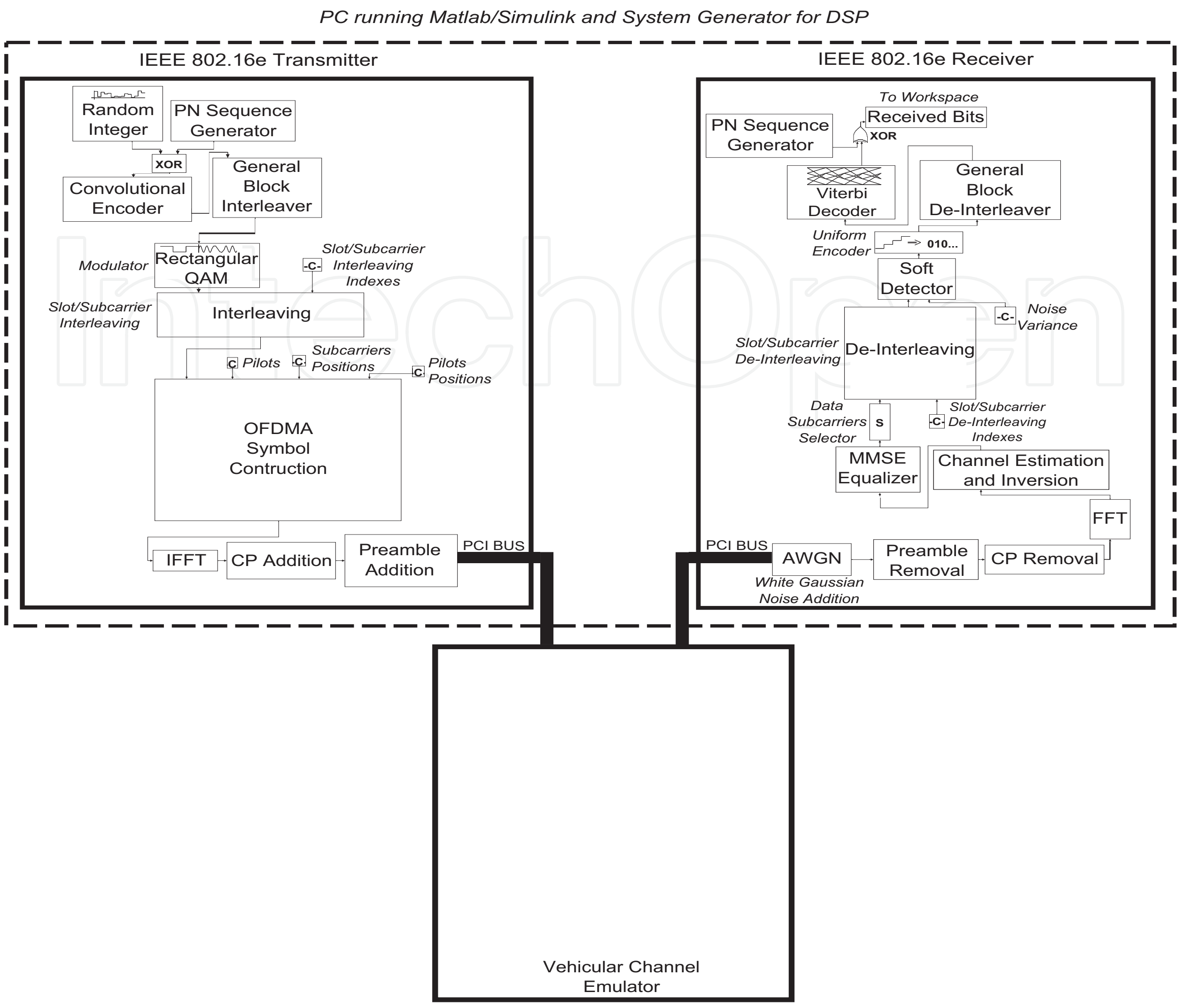

Fig. 1. Evaluation system for IEEE 802.16e.

is divided into fifteen subchannels but, for the sake of simplicity, we assigned all subchannels to a unique user.

2.1.1 Mobile WiMAX MATLAB/simulink transmitter

Fig. 1 (left) depicts the transmitter block diagram. We have followed closely the indications given in Section 8.4 of (IEEE, 2009), although we have made modifications in order to simplify the design and reduce simulation time. Such differences are described below.

An IEEE 802.16e transmitter works with slots made up of two consecutive OFDMA symbols. Also, it should be noticed that in our tests we have considered that a fair comparison between the different standards should be performed measuring the FER (Frame Error Rate), considering that a frame corresponds to a FEC (Forward Error Correction) block. Since each FEC block contains 48 data bits and there are 720 data subcarriers for each slot (two OFDMA symbols), using QPSK and a rate 1/2 code, it is concluded that in each slot the transmitter sends 15 FEC blocks.

With respect to the pilots, the standard uses a Pseudo-Random Binary Sequence (PRBS) generator during their modulation. However, since this generator is only used to provide

additional security, we decided not to include it in our design in order to reduce complexity and simulation time.
Although we always transmit the preamble in order to comply with the requirements of the standard, it is not used in reception since we assume perfect time synchronization. Moreover, the vehicular channel emulator operates in baseband and, therefore, there is no IF (Intermediate Frequency) stage, neither at the transmitter nor at the receiver.

2.1.2 Mobile WiMAX MATLAB/simulink receiver

The receiver block diagram is shown on the right of Fig. 1. The first step is the addition of white Gaussian noise in order to obtain BER (Bit Error Rate) and FER curves versus received SNR (Signal-to-Noise Ratio) or E_b/N_0 values. The E_b/N_0 parameter is the received bit energy to noise power spectral density ratio commonly referred to as received SNR per bit.
After removing the preamble and the CP, the FFT is applied to each OFDM symbol and the channel is estimated. We employ a simple channel estimation method consisting in extracting the pilots and divide them by their respective transmitted values (which are known at the receiver), obtaining the estimated channel coefficients for the pilot subcarriers. Such estimates are linearly interpolated to obtain the channel frequency response for the remaining subcarriers. Moreover, to improve channel inversion, we apply an MMSE (Minimum Mean Square Error) equalizer (Rugini, 2005).
The equalized symbols are de-interleaved (at slot and symbol levels) and then sent to a soft detector whose output LLRs are also de-interleaved and decoded using a Viterbi block. Finally, the decoded bits are de-randomized and the final bits are obtained.

2.2 IEEE 802.11p and IEEE 802.11a

The standard IEEE 802.11p (IEEE, 2010) is an amendment to IEEE 802.11-2007 (IEEE, 2007) and is technically compatible with the specifications given by ASTM E2213-03 (ASTM, 2003), which addresses the challenges that arise when providing wireless access in vehicular environments. Its MAC and PHY layers are very similar to those used in IEEE 802.11a, but they incur in a lower overhead to allow faster exchanges of safety messages.
In our work we focus on the PHY layer and, at such a level, the main difference between IEEE 802.11a and IEEE 802.11p is that the 20 MHz bandwidth used in IEEE 802.11a reduces to 10 MHz in IEEE 802.11p. Although the mentioned bandwidth reduction results in a loss of data transfer rate, it provides an important advantage when overcoming the effects of vehicular channels: the OFDM symbols are longer in the time domain and the system can deal with larger delay spreads, thus being able to avoid ISI (Inter-Symbol Interference). Therefore, if we ignore the IEEE 802.11p ACR (Adjacent Channel Rejection) and the SEM (Spectrum Emission Mask) requirements, the practical implementation of a basic IEEE 802.11p transceiver is straightforward: it suffices to double all the OFDM timing parameters used by IEEE 802.11a devices.
The design of our IEEE 802.11p/a transceivers (whose key parameters are shown in Table 1) can be found in our previous work (Fernández-Caramés, 2010). They include similar blocks to those present in Fig. 1.

3. Real-time FPGA-based vehicular channel emulator

Channel emulation is typically used when evaluating product performance in realistic situations before commercial release. With the aid of a channel emulator the equipment manufacturers avoid unintended interferences, hence the simulation environment can be controlled. Furthermore, the tiresome task of performing successive field measurements is

Parameter	**802.11p**	**802.11a**	**802.16e**
Carrier Modulation	BPSK, QPSK, 16-QAM, 64-QAM		
Code rate	1/2, 2/3, 3/4		
# subcarriers	48 data + 4 pilots		360 data + 60 pilots
OFDM symbol duration	8 μs	4 μs	64 μs
Guard time	1.6 μs	0.8 μs	12.8 μs
FFT period	6.4 μs	3.2 μs	51.2 μs
Total bandwidth	10 MHz	20 MHz	10 MHz
Subcarrier spacing	156.25 KHz	312.5 KHz	19.53 KHz

Table 1. Feature comparison of wireless standards.

limited to the minimum (to obtain the channel model, if there is none already available) and the rest of the experiments are carried out inside a testing lab.
There are many commercial channel emulators that are manufactured by companies such as Spirent, Rhode & Schwarz, Azimuth Systems, Agilent... These emulators are usually general-purposed (for instance, Spirent's SR5500 or Rhode's AMU200A), but there are some that are aimed at evaluating a specific technology, like Azimuth's 400WB MIMO Channel Emulator (for IEEE 802.11n and Mobile WiMAX MIMO systems) or Agilent's N5106A PXB MIMO Receiver Tester (with built-in LTE and Mobile WiMAX channels).
All these channel emulators are robust and work great for most applications, but they are normally quite expensive and may not offer enough flexibility to researchers when setting channel configuration parameters. To tackle these issues a number of low-cost ad-hoc channel emulators have recently been proposed.
Due to real-time constraints and suitability, most of the proposed low-cost, easily-reconfigurable ad-hoc emulators are based on FPGA technology. Some examples are described in (Alimohammad, 2008; Ghazel, 2003; Hwang, 2007). In (Ghazel, 2003) an FPGA-based AWGN channel emulator is implemented. The emulator is based on a hardware white Gaussian noise generator that is developed by combining the Box-Muller and Central limit theorems, and designing the whole model in VHDL (Very High Speed Integrated Circuit Hardware Description Language). Similarly, in (Alimohammad, 2008), the authors use a Xilinx Virtex-II Pro to implement a fading channel emulator. The fading process models use sum-of-sinusoids (SOS) algorithms that allow designing and implementing Rician and Rayleigh fading channels.
Finally, (Hwang, 2007) presents a baseband multipath fading channel emulator implemented on a Virtex-IV using the Xilinx XtremeDSP FPGA platform. The emulator is implemented using Simulink models and System Generator IP blocks. The final design is limited to a two-path channel due to the extensive use of FPGA resources; the input/output rate is set to 20 MHz; and the Doppler frequency is 5 Hz.
The above mentioned developments have at least two major drawbacks. First, the use of low-level description languages such as VHDL results in slow development stages. Although in most cases VHDL allows obtaining a resource-efficient FPGA design, programming can become a cumbersome task that may consume a large amount of time and economic resources. There are new sophisticated tools like System Generator which allow working with high-level blocks to build complex designs easier and faster.
The second problem is related to the use of high-level tools. These tools facilitate fast prototyping but they usually generate non-optimized large designs that may not fit into the FPGA. For instance, in (Hwang, 2007) the authors only download a two-path channel emulator due to the lack of available hardware resources. Hence, for large designs, optimizations must be made.
The vehicular emulator described in this chapter addresses these issues: we used System Generator to develop the channel emulator faster than using VHDL and we optimized our

design in order to be able to implement a twelve-path channel emulator, leaving space for future extensions, such as MIMO emulation, also described herein in Section 4.

3.1 Implemented vehicular channel models

The implemented channel models are based on the excellent work in (Acosta, 2007b) and (Acosta, 2007a), that is mainly based on a measurement campaign at 5.9 GHz carried out in the spring of 2006 in Atlanta, Georgia. From these measurements the authors obtained six different channel models that cover the most common situations where VTV and RTV communications may take place:

- Urban canyons, with dense and tall buildings, where vehicles move at speeds between 32 Km/h and 48 Km/h.
- Suburban expressways, with moderately dense, low-story buildings, where the measurement speed was approximately 105 Km/h.
- Suburban surface streets, with moderately dense, low-story buildings, where the driving speed was between 32 Km/h and 48 Km/h.

Although the measurement campaign was performed at 105 Km/h in expressways and 32 Km/h to 48 Km/h for surface streets, the authors scaled the models to make their Doppler frequencies consistent with vehicle speeds of 140 Km/h and 120 Km/h, respectively.

Vehicular Channel	Distance TX - RX (m)	Speed (km/h)	Path Modulation (number of paths)	Maximum Delay Spread (ns)	Rician K (dB)	Overall K Factor (dB)	Maximum Freq. Shift (Hz)	Maximum Fading Doppler (Hz)	LOS Doppler (Hz)
VTV-Expressway Oncoming	300-400	140	Rician (1) / Rayleigh (10)	302	-1.6	-3.6	1466	858	1452
RTV-Urban Canyon	100	120	Rician (1) / Rayleigh (11)	501	7.5	6.7	720	994	654
RTV-Expressway	300-400	140	Rician (1) / Rayleigh (11)	401	-5.3	4.3	769	813	770
VTV-Urban Canyon Oncoming	100	120	Rician (1) / Rayleigh (11)	401	4.0	3.0	1145	936	1263
RTV-Suburban Street	100	120	Rician (1) / Rayleigh (11)	700	3.3	2.1	648	851	635
VTV-Express. Same Dir. with Wall	300-400	140	Rician (2) / Rayleigh (10)	701	23.8, 5.7	3.3	-561	1572	-60, 40

Table 2. Main characteristics of the vehicular models.

Our channel emulator implements these six vehicular channel models, whose key characteristics are summarized in Table 2. For each model, the following parameters are shown: distance between the transmitter and the receiver, speed of the vehicle, number of paths of the channel and their modulation (Rician or Rayleigh), maximum delay spread, Rician K for the Rician paths, overall K factor (i.e. the ratio of the deterministic power over the total random power of all taps), maximum frequency shift for all paths, maximum fading Doppler (i.e. maximum half-width of the fading spectral shapes of all the paths of each channel) and LOS Doppler of the Rician paths.

Vehicular Channel	Coefficient generation rate [Effective rate] (Hz)	Interpolation rate	Occupied Slices (%)	Occupied Slice Flip-Flops (%)	Occupied LUTs (%)	Occupied FIFO16 / RAMB16s (%)	Occupied DSP48s (%)
VTV-Expressway Oncoming	3484 [4000]	x2500	76%	36%	50%	19%	60%
RTV-Urban Canyon	2194.6 [2500]	x4000	84%	39%	57%	20%	65%
RTV-Expressway	2168 [2500]	x4000	84%	39%	57%	20%	65%
VTV-Urban Canyon Oncoming	3314 [4000]	x2500	84%	39%	57%	20%	65%
RTV-Suburban Street	1988 [2000]	x5000	84%	39%	57%	20%	65%
VTV-Express. Same Direction With Wall	3170 [4000]	x2500	85%	40%	57%	24%	65%

Table 3. General parameters and resources occupied by the vehicular channel emulator.

Table 2 also gives an idea of the complexity involved in the implementation of these channels. These high speed and high delay spread scenarios own large Doppler shifts that force the emulator to interpolate and rapidly update each path coefficients. Moreover, although the amount of required FPGA hardware is reduced by working with the baseband IQ components

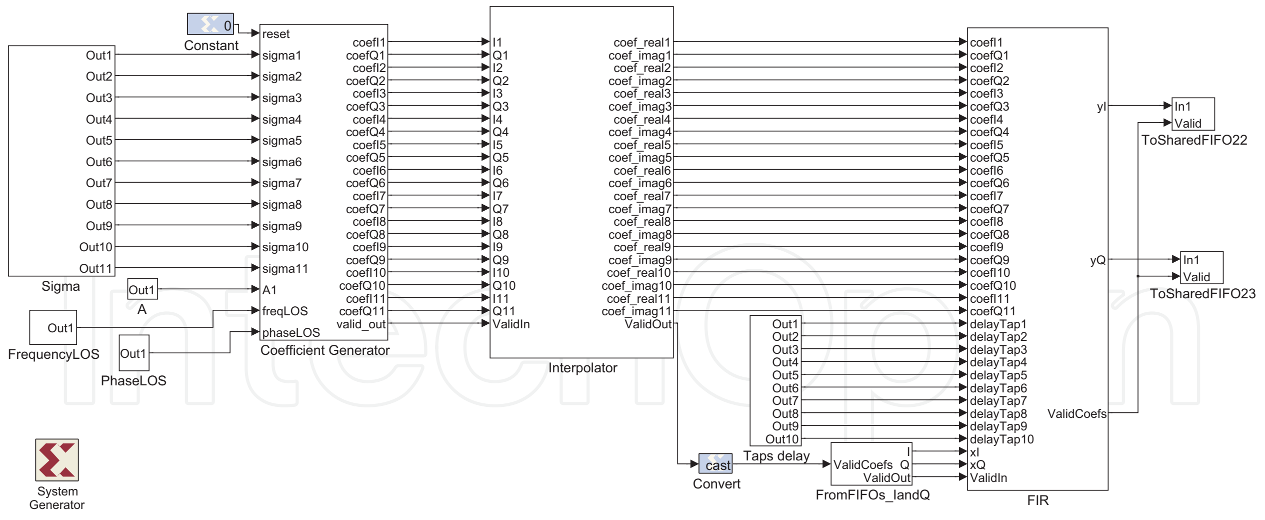

Fig. 2. General view of the System Generator model optimized for the vehicular channel *VTV-Expressway Oncoming*.

at 10 MHz, it is not possible to implement the six channels into our FPGA. Thus, six independent `.bit` files are generated though, in practice, only three different FPGA designs are needed due to the model similarities:

- One design is exclusively dedicated to the channel *VTV-Expressway Oncoming* which is the only one with eleven paths.
- Another model is used for *VTV-Expressway Same Direction with Wall* because it requires the existence of two Rician and ten Rayleigh paths, while the rest of the channels (apart from *VTV-Expressway Oncoming*) consists of just one Rician path and eleven Rayleigh paths.
- One design for the other four channels, which differ in their configuration parameters but share all their FPGA resources.

3.2 Theoretical channel model

For the generation of each channel coefficient at the i-th path in the time instant t, we used the following model

$$h(i,t) = \sqrt{K_i P_i/(K_i+1)}\,\overline{h}(i,t) + \sqrt{P_i/(K_i+1)}\,h_w(i,t) \tag{1}$$

where

- K_i and P_i are, respectively, the Rice factor and the power of the i-th path.
- $h_w(i,t)$ represents the contribution of the non-line-of-sight (NLOS) component to the i-th path at the time instant t. It is a random variable that follows a complex Gaussian distribution with mean zero and unit variance.
- $\overline{h}(i,t)$: contribution of the line-of-sight (LOS) component to the i-th path at the time instant t. It is determined by

$$\overline{h}(i,t) = e^{j(2\pi f_{D,i}\cos(\theta_i)t+\phi_i)} \tag{2}$$

  where $f_{D,i}$, θ_i and ϕ_i are, respectively, the maximum Doppler spread, angle of arrival and phase of the LOS component of the i-th path.

To decrease the number of input configuration parameters, we calculate off-line several of the operations involved in (1) and (2). As it can be viewed in Fig. 2, the emulator only needs five parameter blocks:

- `Sigma` block contains the power factors of the NLOS components: $\sigma = \sqrt{P_i/(K_i+1)}$.
- `A` block holds the power factors of the LOS components: $A = \sqrt{K_i P_i/(K_i+1)}$.
- `FrequencyLOS` block contains part of the exponent of $\overline{h}(i,t)$: $f_{\text{LOS}} = 2\pi f_{D,i} cos(\theta_i)$.
- `PhaseLOS` block is simply ϕ_i.
- `Taps delay` block holds the normalized delays of the different paths.

3.3 Hardware and software

The vehicular channel emulator is implemented on an FPGA using Nallatech's BenADDA-IV development kit which has the following features: it contains a Virtex IV (XC4VSX35-10FF668) that allows using Xtreme-DSP slices of up to 400 MHz; includes 4 MB of 166MHz ZBT-RAM, two 14-bit ADCs able to work up to 105 MS/s and two 14-bit DACs that can run up to 160 MS/s; provides a dedicated internal clock up to 105 MHz, although it can use an external clock; offers the possibility to operate either connected to a PC (via the PCI bus) or in stand-alone mode.

In order to diminish the amount of time required to implement the channel model on the FPGA, we decided to use System Generator for DSP because it enables to design and program our Virtex IV faster. It allows using libraries of high-level blocks and can interact with MATLAB and Simulink. Moreover, another advantage of this software is its ability to exchange data between a design running in the FPGA and a software implementation that is executed on a PC. In fact, for our tests (Section 5) we have run in MATLAB and Simulink the transceivers that interact with the vehicular emulator, which was running on the FPGA.

3.4 FPGA design overview

Fig. 2 shows a general view of the hardware design. Several blocks represented in such figure contain sub-blocks which are shown throughout this chapter: the `Coefficient Generator` block includes Figs. 3 and 4, the `Interpolator` block contains a number of interpolators like the one shown in Fig. 5.

The design depicted in Fig. 2 has been optimized for a specific channel (*VTV-Expressway Oncoming*), although the rest of channels models share most of the hardware resources. The design can be divided into different parts that carry out six different major tasks: acquisition of the channel parameters, Gaussian noise generation, Doppler filtering, LOS Doppler generation, interpolation and FIR filtering.

3.4.1 Acquisition of the channel parameters

The generation of the configuration parameters of the vehicular channel is performed offline since they remain constant throughout the emulation. The parameters are stored into registers readable by the FPGA. All the parameters equal to zero for a particular channel are removed to save hardware. For example, all the channels but *VTV-Expressway Same Direction With Wall* have one Rician component, so in these channels we only need one register to store each of the LOS parameters detailed in Section 3.2.

3.4.2 White gaussian noise generation

To obtain the NLOS coefficients, we need to use the System Generator's White Gaussian Noise Generator (WGNG) block that generates i.i.d samples from a Gaussian distribution with zero mean and unit variance. Since the maximum number of complex paths is twelve, we need to obtain twenty-four real-valued of such noise samples that will be filtered depending on the Doppler experienced by each path. Instead of using twenty-four independent WGNG blocks, we multiplex in time the samples produced by only one WGNG block (Fernández-Caramés, 2010). This optimization is crucial since each WGNG block consumes an important amount of FPGA resources. Since each WGNG block runs at 10 MHz, using a twenty-five output multiplexor (twenty-five is the integer divider of 10 MHz closest to twenty-four), leads us to generate twenty-four noise samples at a frequency of 400 KHz, that still is several orders of magnitude higher than the desired channel coefficient generation effective rates (see Table 3). Therefore, the optimization of the 24-output Gaussian noise generator makes the emulator to produce channel coefficients slower but at a sufficiently high rate, and saving a great deal of FPGA resources.
Using the System Generator's Resource Estimator block, there is a 95% of saving for all the FPGA resources thanks to this optimization (Fernández-Caramés, 2010). Also, if we needed to reduce the number of occupied resources, it would be possible to avoid using WGNG blocks: the channel coefficients could be generated in MATLAB and then transferred to the FPGA. However, there are important drawbacks in this approach:

- If the channel coefficients were only transmitted from MATLAB during the initialization phase, due to the limited amount of memory on the FPGA, there would be a time when the channel coefficients would have to be used again. Therefore, correlation in the output signal would appear.
- If we transfer a new set of channel coefficients from MATLAB at fixed intervals, we would not be able to use the emulator in stand-alone mode since we always would relay on having a computer running MATLAB linked to the FPGA.

3.4.3 Doppler filtering

To generate the actual NLOS components, the generated white Gaussian noise samples have to be filtered according to each path's Doppler spectrum. This spectrum is determined by a fading spectral shape, a frequency shift and a maximum Doppler shift. Table 2 shows these latter two parameters for the considered channel models. Four different spectral shapes are considered: *round*, *flat*, *classic 3 dB and classic 6 dB* (Acosta, 2007a).

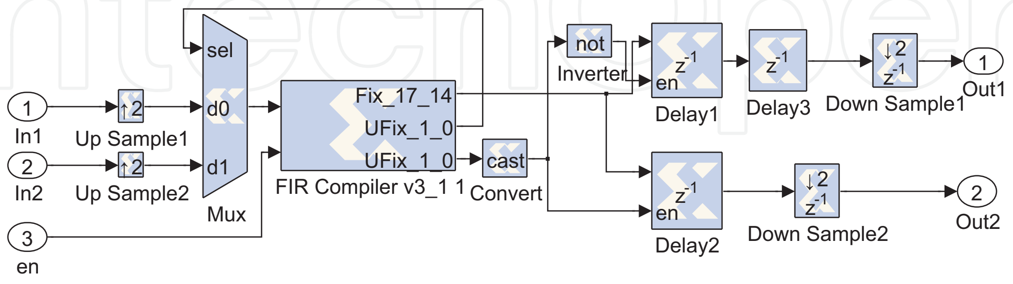

Fig. 3. Optimized blocks for applying the Doppler Spectrum.

Fig. 3 shows the blocks that allow applying the Doppler spectrum to each Rayleigh path. Each Doppler filter consists of 256 coefficients. This filter size provides a good tradeoff between precision and hardware complexity. Since each filter is unique for each path of each vehicular channel, we hard-coded the coefficients in each of the six `.bit` files.

To reduce to a half the required hardware, we exploit the fact that the real and the imaginary parts of the filter have to be used twice for each path to perform the complex FIR filtering (see Fig. 4). This optimized block can be seen in Fig. 3, which is contained under the block *Doppler Filter* shown in Fig. 4.
Table 4 shows some of the resource savings, achieved when reducing to a half the number of filters in a vehicular channel with eleven paths. Although the optimized block uses slightly more slices, the savings occur in the DSP48 and the RAMB16 blocks, that are reduced by 50%. This is important, since the lack of this type of blocks is a bottleneck to keep on designing the rest of the emulator.

Resource type	**Optimized 24-output Gaussian generator**	**Non-optimized 24-output Gaussian generator**	**Total FPGA resources**	**Savings (%)**
Slices	7382	7239	15360	-1.9%
DSP48 blocks	88	176	192	50%
RAMB16 blocks	44	88	192	50%

Table 4. Savings due to the optimization of the Doppler filter block

Finally, LOS Doppler has also to be taken into account and must be applied to each Rician path according to Eq. (2). To achieve this, we use the System Generator's DDS (Direct Digital Synthesizer) block that generates a sine and a cosine with the required phase and frequency parameters. Since the angle of arrival of the LOS component has not been considered in (Acosta, 2007b), we always set its value to zero, what means that the received Rician paths arrive straight from the driving direction. This implies that the LOS Doppler is always equal to the path's maximum Doppler spread.

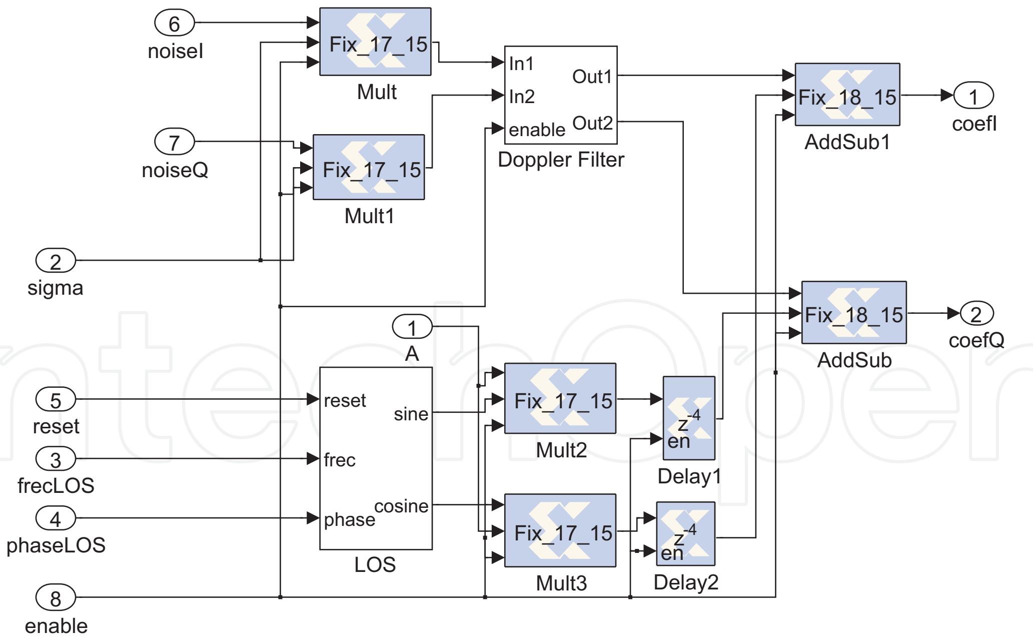

Fig. 4. Generation and addition of the LOS and NLOS components of each path.

3.4.4 Interpolation and FIR filtering

After adding the LOS and NLOS components according to Eq. (1) (see Fig. 4), the coefficients must adapt their rate to the rate of the incoming signal (i.e. the signal from our transmitters

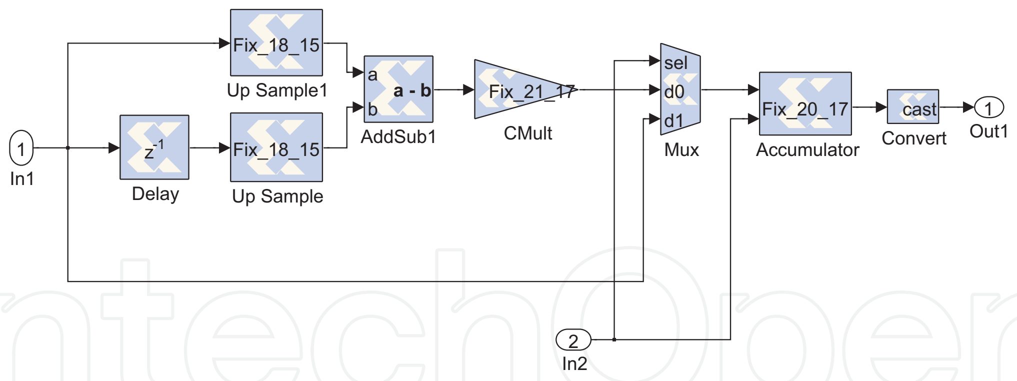

Fig. 5. One path's linear interpolator.

arrives at 10 MHz). These coefficients are generated at a rate that depends on the maximum Doppler shift and that is much lower than the FPGA's frequency. Indeed, in a specific vehicular channel, the implicit sample rate is twice the maximum Doppler shift of all paths. In the implemented vehicular channel models, this rate fluctuates between 1988 Hz and 3484 Hz (see Table 3). To avoid designing a complex resampling stage, instead of using the original coefficient generation rate, we use an effective sample rate that is equal to the nearest superior integer divider of 10 MHz. Thus, we only need an interpolator (we actually use two cascaded linear interpolators, whose global interpolation rates are also included in Table 3).
Fig. 5 shows a linear interpolator applied to one of the paths. The way it works is simple:

- At the time instant t, the current coefficient and the one generated at the time instant $t-1$ are copied p times, being p the interpolation factor. Hence, we would have two sets of upsampled coefficients: $\mathbf{s}_t = [\overbrace{s_t, s_t, ..., s_t}^{p}]$ and $\mathbf{s}_{t-1} = [\overbrace{s_{t-1}, s_{t-1}, ..., s_{t-1}}^{p}]$. At the time instant 0, the upsampled coefficients at $t-1$ are all equal to 0. If the interpolator is currently in a time instant superior to 0, it is assumed that there exists a previously interpolated value y_{t-1}.
- Next, the n-th upsampled coefficients from $\mathbf{s}_t$ and $\mathbf{s}_{t-1}$ are subtracted and divided by p: $\Delta_t = (s_{t_n} - s_{t-1_n})/p$.
- Finally, Δ_t is added or subtracted (depending on its sign) to/from the interpolation value calculated at the time instant $t-1$, obtaining the current output interpolated value: $y_t = y_{t-1} + \Delta_t$.

Finally, the signal from the transmitter is filtered with the interpolated coefficients. Details about the implemented FIR filter can be found in (Fernández-Caramés, 2010).

3.5 Emulator basic operation

The emulator operation can be summarized as follows:

1. The configuration parameters of the vehicular channel are initially read from registers (shown in Fig. 2).
2. The emulator starts to generate channel coefficients, both for the LOS and the NLOS components (illustrated in Figs. 2 to 4).
3. The coefficients are interpolated to have their rate adapted to the incoming signal rate, passing each path through linear interpolators (like the one shown in Fig. 5). The interpolation is carried out in two stages, whose interpolation factors depend on the effective generation rates shown in Table 3. For instance, in the channel *RTV-Urban Canyon*,

the coefficients have an effective generation rate equal to 2500 Hz. Since the incoming signal rate is 10 MHz, the coefficients need to be interpolated with a global factor of 4,000, which can be applied in two stages with interpolation factors 2^5 and 5^3.

4. Finally, the incoming signal is applied to a complex FIR filter that uses the generated channel coefficients.

4. Upgrading to MIMO

First, note that the channels modeled in (Acosta, 2007b) are based on SISO measurements, but we use them because they have become the reference for evaluating IEEE 802.11p. Further investigation is still needed to adapt such channels to multiple-antenna environments, but when that occurs, the transceivers and channel emulator model presented in this chapter will continue to be valid, only requiring slight modifications or no modifications at all. Also, for the sake of brevity, regarding MIMO we will restrict ourselves to IEEE 802.11p transceivers.

4.1 IEEE 802.11p MIMO transceivers

4.1.1 Multiple-antenna transmitter

In the transmitter, the use of several antennas lead us to change our SISO channel estimation and use orthogonal pilots that constitute matrices called OSTPM (Orthogonal Space-Time Pilot Matrices). Specifically, we use Hadamard matrices created using Sylvester's method, which generates a sequence of matrices that are known as Walsh matrices. Such matrices are orthogonal in space and time and, in the case of transmitting with two antennas, they are generated according to:

$$\mathbf{P} = \begin{pmatrix} p_k & p_k \\ p_k & -p_k \end{pmatrix} \tag{3}$$

where p_k is the BPSK-modulated pilot symbol transmitted at the k subcarrier. Since IEEE 802.11p uses four pilots inside each OFDM symbol, the pilot matrix is generated by replicating Equation (3) to obtain a 2×4 matrix.

Using this scheme, channel estimation only requires simple linear processing. For instance, in the case of transmitting with two antennas, the received signal at the k-th subcarrier for two consecutive OFDM symbols would be:

$$\begin{aligned} y_{1,k} &= p_k h_{1,k} + p_k h_{2,k} + n_1 \\ y_{2,k} &= p_k h_{1,k} - p_k h_{2,k} + n_2 \end{aligned} \tag{4}$$

where n_1 and n_2 are AWGN samples and h_1 and h_2 are the channel coefficients. Thus, the channel coefficient estimations are obtained as:

$$\hat{h}_{1,k} = \frac{y_{1,k} + y_{2,k}}{2p_k} \qquad \hat{h}_{2,k} = \frac{y_{1,k} - y_{2,k}}{2p_k} \tag{5}$$

Note that this channel estimation method has several limitations. First, it assumes that the channel remains constant over two consecutive pilots, so when the Doppler spread is high, performance will be degraded. The second drawback is related to the pilot generation matrix: it is only possible to use this pilot scheme when the number of transmit antennas is a power of two. Moreover, it requires an even number of transmitted OFDM symbols to be transmitted.

In spite of the above mentioned issues, we stick to using this method due to its simplicity and because the maximum Doppler shift of the implemented channels is 1742 Hz, that corresponds to a channel coherence time of 574 μs, which is clearly higher than the time required to transmit two consecutive OFDM symbols (16 μs).

Apart from the modifications required by the channel estimation step, to exploit space-time diversity, MIMO systems need an additional coding stage. In the case of 2×2 system Alamouti coding is used (Alamouti, 1998), whereas the 4×4 transceiver implements a quasi-orthogonal code proposed by Jafarkhani (Jafarkhani, 2000).

4.1.2 Multiple-antenna receiver

At the receiver side the main changes with respect to the SISO system are related to support diversity schemes. In the SIMO case the system implements the MRC (Maximum-Ratio Combining) technique, while the MIMO transceiver requires the use of an Alamouti decoder in the 2×2 case, and a ML (Maximum-Likelihood) detector otherwise. We do not give herein more details about the receivers since they use standard MIMO algorithms and techniques.

4.2 MIMO vehicular channel emulation

There are several examples of academic MIMO FPGA-based channel emulators. Some of them are generic (e.g. (Ren, 2010; Wang, 2008; Zhan, 2009)), while others (e.g. (Eslami, 2009)) are specifically oriented towards the implementation of the IEEE 802.11n reference channel models. However, none of the existing channel emulators has been explicitly developed for recreating VTV or RTV environments.

One of the main problems when implementing MIMO channel emulators in an FPGA is that they require large designs and, therefore, the use of resources has to be optimized. Most of the channel emulators described in the literature are able to implement the whole system into only one FPGA. To fit the design into one FPGA, researchers have to save resources using several clever tricks, being one of the most recurrent the off-line generation of the channel coefficients (Eslami, 2009; Zhan, 2009). Also, some authors (Eslami, 2009) are able to save up to 67% of the FPGA resources by applying the channel coefficients in the frequency domain.

These academic developments present at least three drawbacks. First, the use of low-level description languages such as VHDL slows down the development stage.

The second problem is related to the portability of the channel emulator. A good channel emulator should be able to work in stand-alone mode, i.e. without needing external devices to generate and transfer channel coefficients to the FPGA.

The third drawback is related to scalability. As it can be derived from the results exposed in (Eslami, 2009), when we work with a time-domain based channel emulator, the gate count (i.e. the number of 2-NAND logic gates that would be required to implement the same number and type of logic functions) roughly doubles every time we add a transmit and a receive antenna to the system. Therefore, a scalable solution would have to be able to deal with more inputs and outputs without requiring such important hardware complexity increases.

The vehicular emulator described in this chapter addresses these three drawbacks: we use Xilinx System Generator to develop the channel emulator faster than using an HDL, we optimize our design in order to fit a MIMO twelve-path channel emulator into one FPGA, we design the emulator bearing in mind that it has to be able to work in stand-alone mode with minimal modifications and we propose a time-multiplexing solution that has a very low impact on the emulator design, thus facilitating scalability.

4.3 Refining the emulator: from SISO to MIMO

Our first attempt to expand our SISO emulator to accept more input and output antennas consisted in creating a SIMO 1×2 system by replicating the SISO design. The obtained design was too large to fit into our FPGA, so we proceeded to optimize it. For the sake of brevity, we will only cite the three most important optimizations we carried out, whose savings are summarized in Table 5 and where the most complex out of the six designs (in terms of FPGA resources consumed) is used as a reference (*VTV-Expressway Same Direction with Wall*).

Version	Slices	Slice flip-flops	LUTs	FIFO16/RAMB16	DSP48
SISO	85%	40%	57%	24%	65%
SIMO 1x2 (V1)	113%	74%	110%	40%	100%
SIMO 1x2 (V2)	107%	71%	104%	36%	100%
SIMO 1x2 (V3)	99%	69%	89%	37%	78%
MIMO 4x4	82%	43%	60%	27%	66%

Table 5. Resource utilization of different versions of the vehicular channel emulator.

The first optimization reduced the amount of resources dedicated to perform the Doppler filtering stage (i.e. the stage aimed at applying each path's Doppler spectrum) by using a four-output Doppler filter. This filter is a natural evolution of the SISO filter shown in Fig. 3. It makes use of a four-input multiplexor and has a four-output demutiplexor after the *FIR Compiler* block (see Fig. 6). This is possible since every path uses the same Doppler filters. The resources occupied by this optimized version of the emulator are shown in Table 5 in the row *SIMO 1x2 (V1)*.

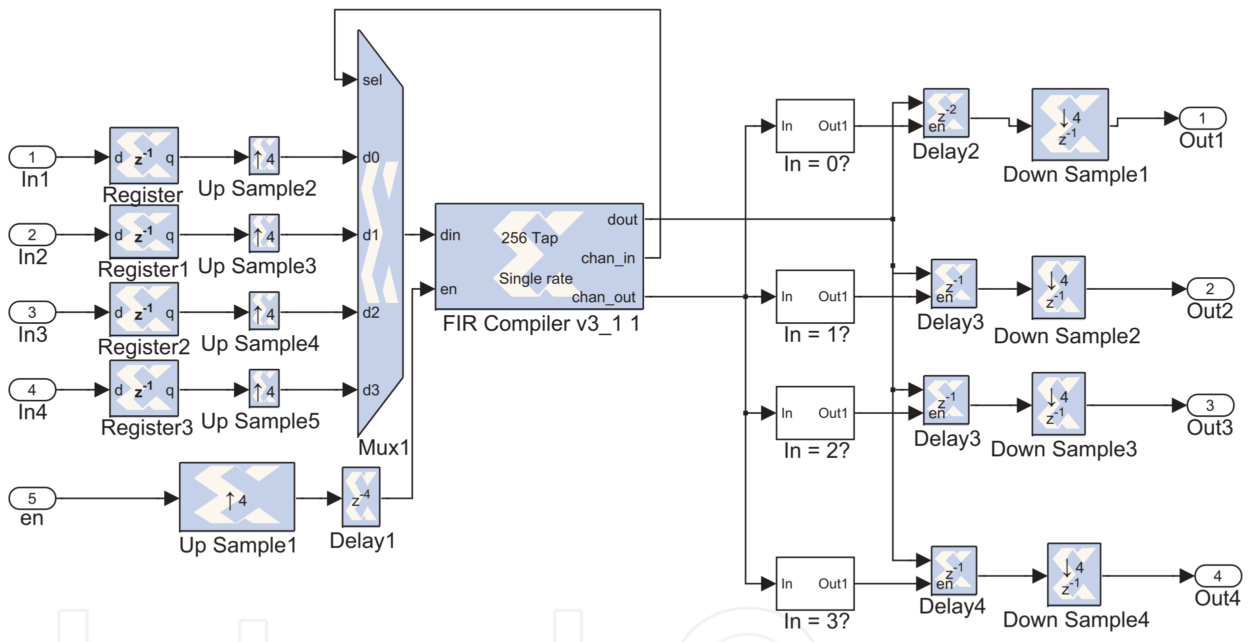

Fig. 6. Doppler filtering stage developed for SIMO 1×2 (V1).

Our second optimization was related to the high resource consumption of each System Generator's AWGN block, as already mentioned in Section 3.4.2. Thus, we removed one of the two AWGN generators and created a 50-output Gaussian generator (we only need 48 outputs, but 50 is the closest integer divider of 10 MHz). Note that System Generator's demultiplexors are restricted to use up to 32 outputs, so we had to build the 50-output demultiplexor depicted in Fig. 7. As it can be seen in Table 5, in the row *SIMO 1x2 (V2)*, this optimization allowed us to save 6% of the slices, 3% of the slice flip-flops, 6% of the LUTs and 4% of the FIFO16/RAMB16 blocks, but it was yet too large to fit into our FPGA.
The third optimization consisted in allowing every path to share Doppler filters, interpolators and FIR filters, being the Gaussian noise generated unique for each path. Hence, we built a multiplexed version of the emulator that buffered incoming signals, switched the Gaussian noise source and applied the channel to each transmitted signal at the proper time instants. As it is shown in the row *SIMO 1x2 (V3)* of Table 5, the used resources decreased substantially

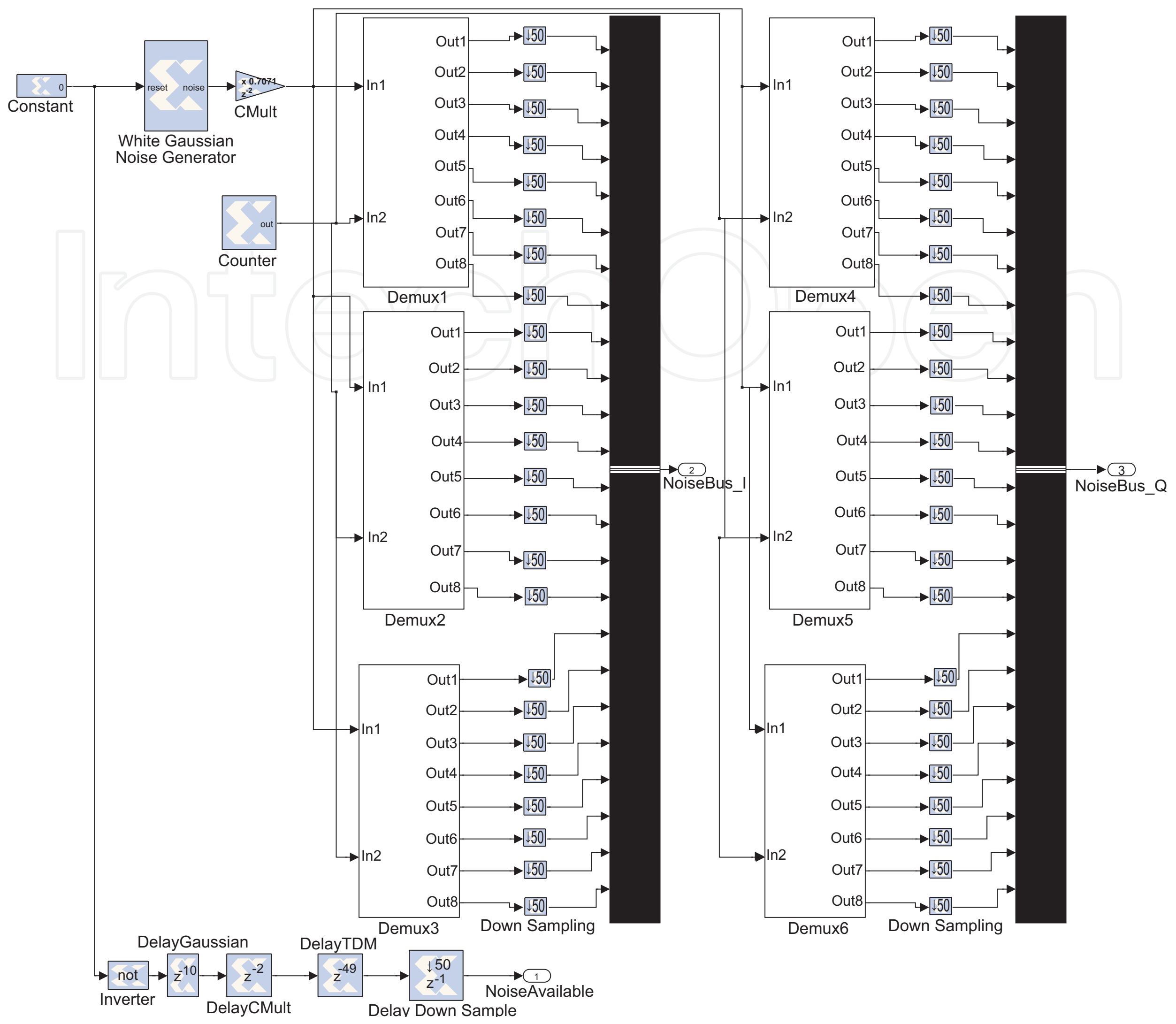

Fig. 7. Gaussian noise generator block developed for SIMO 1×2 (V2).

with respect to version V2 and permitted to save 8% of the slices, 2% of slice flip-flops, 15% of the LUTs and 22% of the DSP48 blocks. Thus, the design fitted into our FPGA, but it was clear that it would be very difficult to fit a MIMO 4×4 system (which has 8 times more paths than an SIMO 1×2 system) following this optimization strategy.

To solve the space issue we devised different alternatives, but we finally resorted to a scheme whose complexity lays in input and output buffers. Such buffers act similarly to two synchronized parallel-to-serial and serial-to-parallel converters. Each set of signals transmitted from an array of IEEE 802.11p transceivers is stored into a buffer and released at specific time instants, achieving a similar effect as if the parallel transceivers were executed in serial. In Table 5, row *MIMO 4x4*, it can be seen the important resource savings attained by using the time-multiplexing approach, which even consumes 3% less slices than our original SISO version thanks to some of the above described optimizations.

We would like to emphasize that although the resources consumed by the time-multiplexed approach have been indicated in Table 5 using the term *MIMO 4x4*, such resources would be the same for any system of up to four antennas in transmission and four antennas in reception. It is also important to note that the input buffer has to add zeroes between each pair of signals that were transmitted by different antennas in order to reduce time correlation. In our emulator we set the coefficient generation rate to frequencies that range between 2 KHz and 4 KHz, so the shortest time the channel remains almost constant would be 500 μs. Therefore, if the FPGA clock is set to 20 MHz, the number of cycles that the channel values remain almost constant would be 500 μs / 50 ns $= 10,000$. To stay safe, we can wait for 100,000 cycles (5 ms) to guarantee minimal correlation. Hence, we separate each pair of signals by 5 ms.

5. Experiments

Performance evaluation of the software transceivers was carried out by passing the signals they produce through the FPGA-based vehicular channel emulator. Taking advantage of the Xilinx Xtreme DSP software kit capabilities, measurements are performed using the *co-simulation mode*: the transmitter and the receiver run in MATLAB and Simulink, while the channel emulator runs on the FPGA. A maximum of 100,000 48-bit FEC blocks are averaged for each SNR (or E_b/N_0) value (the simulation stops when 100 erroneous FEC blocks are detected).

5.1 Mobile WiMAX Vs. IEEE 802.11p/a

In this subsection we present the results of a performance comparison between Mobile WiMAX, IEEE 802.11p and IEEE 802.11a transceivers. For a fair comparison we set the same transmission parameters for every transceiver. A rate 1/2 FEC is used and the subcarriers are filled with QPSK modulated symbols. The receiver assumes perfect time synchronization and, after estimating the channel using a pilot-aided scheme, an MMSE linear equalizer is applied. The remaining transceiver parameters are shown in Table 1. Additionally, for the sake of fairness, instead of comparing performances in terms of PER (Packet Error Rate), we obtained the FEC Frame Error Rate (FER) when all the transceivers make use of the same FEC block size.

5.1.1 Performance over AWGN and rayleigh fading channels

In order to obtain a performance reference, we evaluated the implemented transceivers over two non-vehicular environments. Fig. 8.(a) and (b) show, respectively, the transceivers performance over an AWGN channel and a frequency-flat Rayleigh block fading channel whose coefficients were constant during 15 FEC blocks (i.e. one Mobile WiMAX slot).

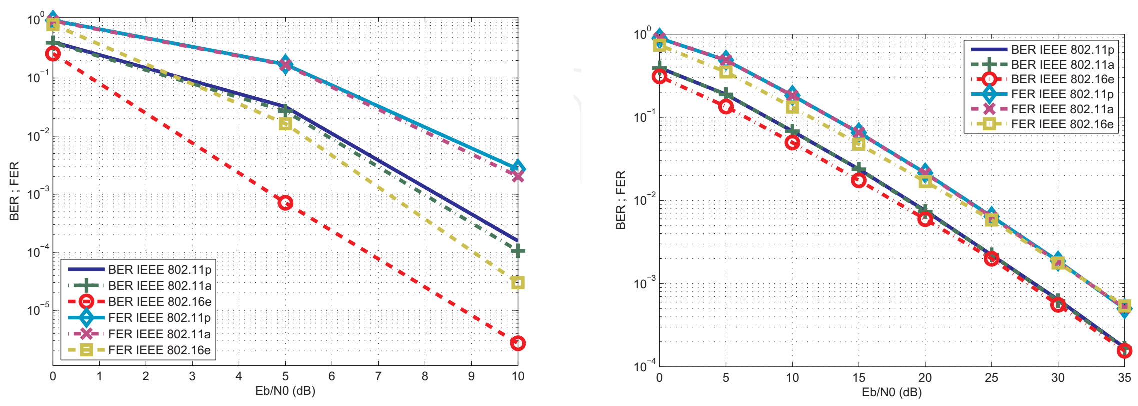

Fig. 8. Performance over (a) AWGN channel (left) and (b) block-fading Rayleigh channel (right).

For both channels, IEEE 802.11p and IEEE 802.11a produce roughly the same results. This was expected, since the transceiver is the same in all aspects apart from the bandwidth. However, the IEEE 802.16e transceiver yields much better results, especially in the AWGN channel. This is because channel estimation is far more accurate in the case of the IEEE 802.16e transceiver. Indeed, Fig. 9 shows the Mean Squared Error (MSE) between the estimated and the true channel for the IEEE 802.11p (the same results were obtained with IEEE 802.11a) and IEEE 802.16e transceivers when considering an AWGN channel. IEEE 802.16e better estimates the channel thanks to the use of one pilot for each group of six data subcarriers, while IEEE 802.11p makes use of only one pilot for each group of twelve data subcarriers.

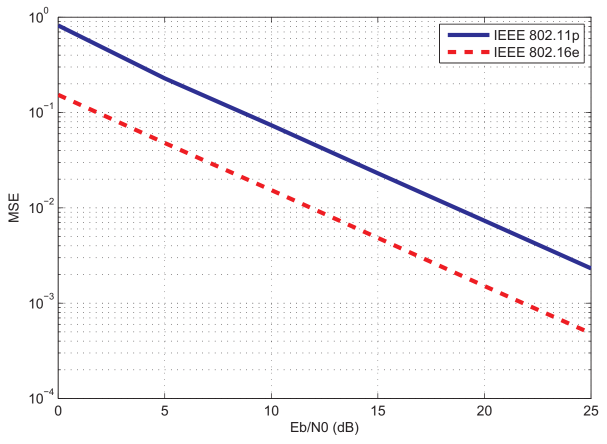

Fig. 9. Channel estimation error when transmitting over an AWGN channel.

5.1.2 Performance over vehicular channels

Figs. 10 to 15 depict the BER and FER curves for the three transceivers when transmitting over the six vehicular channels described in Section 3. In general, it can be observed that the IEEE 802.16e transceiver produces better results (both in terms of BER and FER) than IEEE 802.11p, while the IEEE 802.11a transceiver obtains the worst global results.

In urban environments (channels *VTV-Urban Canyon Oncoming* and *RTV-Urban Canyon*) Mobile WiMAX outperforms IEEE 802.11p/a in terms of BER and FER for E_b/N_0 values below 20 dB (see Figs. 10 and 11). Also, notice that Mobile WiMAX requires the lowest E_b/N_0 values to reach a target FER of 10%.

In surface streets (*RTV-Suburban Street*) Mobile WiMAX also performs better than the other standards (see Fig. 12). For instance, to reach an FER of 10% Mobile WiMAX requires 8.6 dB, while IEEE 802.11p and IEEE 802.11a need, respectively, 11.1 dB and 14.6 dB.

In expressways (*VTV-Expressway Oncoming*, *RTV-Expressway*, *VTV-Expressway Same Direction With Wall*) the results depend on the channel. In *VTV-Expressway Oncoming* (see Fig. 13) IEEE 802.11p clearly outperforms Mobile WiMAX at both low and high values of E_b/N_0. IEEE 802.11p and Mobile WiMAX both exhibit a similar performance when considering *RTV-Expressway* channels (see Fig. 14): Mobile WiMAX is slightly better than IEEE 802.11p while the situation reverses for high E_b/N_0 values. Finally, in the case of *VTV-Expressway Same Direction With Wall*, Mobile WiMAX clearly obtains a major gain over IEEE 802.11p/a: for example, it requires an E_b/N_0 of 6.2 dB less than that of IEEE 802.11p (7.6 dB vs 13.8 dB) to obtain an FER of 10% (see Fig. 15).

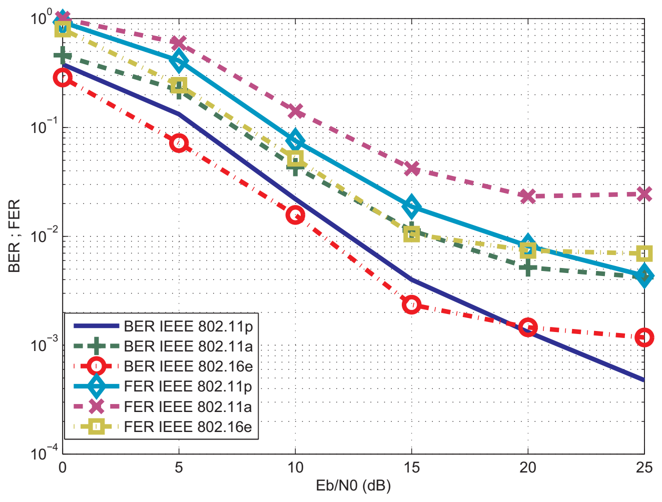

Fig. 10. Performance comparison when transmitting over *VTV-Urban Canyon Oncoming*.

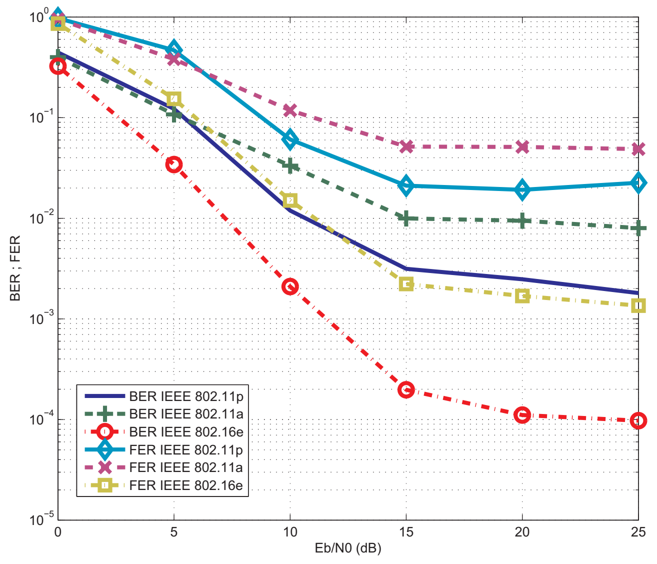

Fig. 11. Performance comparison when transmitting over *RTV-Urban Canyon*.

5.1.3 Discussion: Mobile WiMAX or IEEE 802.11p?

The BER/FER versus E_b/N_0 curves depicted in the previous subsection indicate that the PHY Layer of Mobile WiMAX outperforms that of IEEE 802.11p in most of the reference channel models used as benchmarks in vehicular communications.

An explanation of this behavior is the superior robustness to high channel delay spreads of the Mobile WiMAX PHY Layer. In IEEE 802.11p, assuming a bandwidth of 10 MHz and 64 subcarriers, a 1/4 cyclic prefix will lead to a guard time of 1.6 μs. In the case of Mobile WiMAX, a transceiver that uses 10 MHz of bandwidth and 512 subcarriers has a 1/4 cyclic prefix that lasts 12.8 μs. Thus, the OFDM symbols used in Mobile WiMAX can equalize channels with a larger delay spread. Another advantage of the Mobile WiMAX PHY Layer is that the maximum data rate that it can reach is 39.9 Mbits/s while this value is only 27 Mbits/s in the case of IEEE 802.11p.

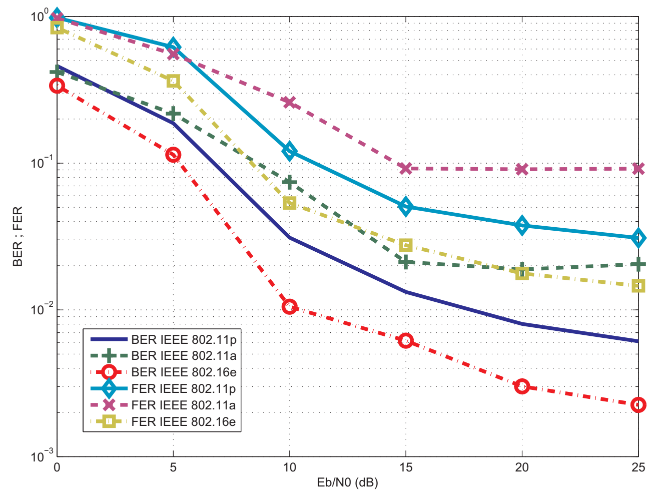

Fig. 12. Performance comparison when transmitting over *RTV-Suburban Street*.

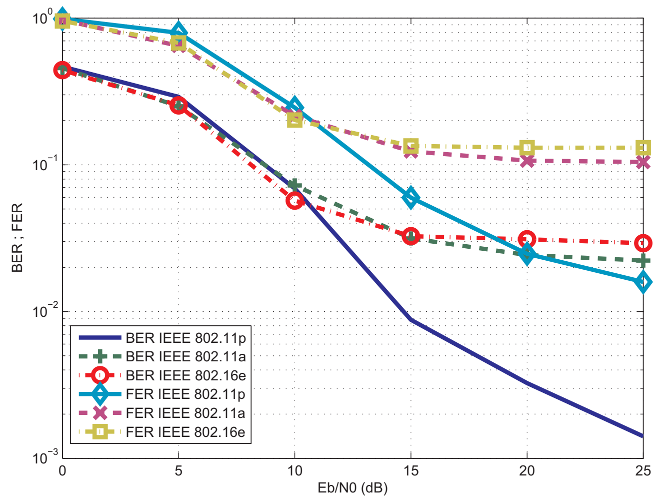

Fig. 13. Performance comparison when transmitting over *VTV-Expr. Oncoming*.

On the other hand, it should be mentioned that the PHY layer of IEEE 802.11p supports larger vehicle speeds. Indeed, IEEE 802.11p is designed to transmit 1000 byte data packets with an FER lower than 10% at a maximum Doppler shift of ±2100 Hz, what means that a top speed of roughly 385 Km/h can be reached when using the 5 GHz band (the transmitter and receiver would drive at almost 193 Km/h). This explains the superior performance of IEEE 802.11p over the *VTV-Expressway Oncoming*, which is the channel with maximum Doppler shift (see Table 2).

5.2 IEEE 802.11p MIMO measurements

In order to achieve a fair comparison we have set the same transmission parameters for every transceiver and we have assumed that all of them send signals with the same transmission power. A rate 1/2 FEC code was used and the OFDM subcarriers were filled with QPSK-modulated symbols. The receiver assumed perfect time synchronization and the channel was estimated using the OSTPM-based method described in Section 4.1.1. In SISO systems an MMSE linear equalizer followed by an ML detector was used, whilst SIMO

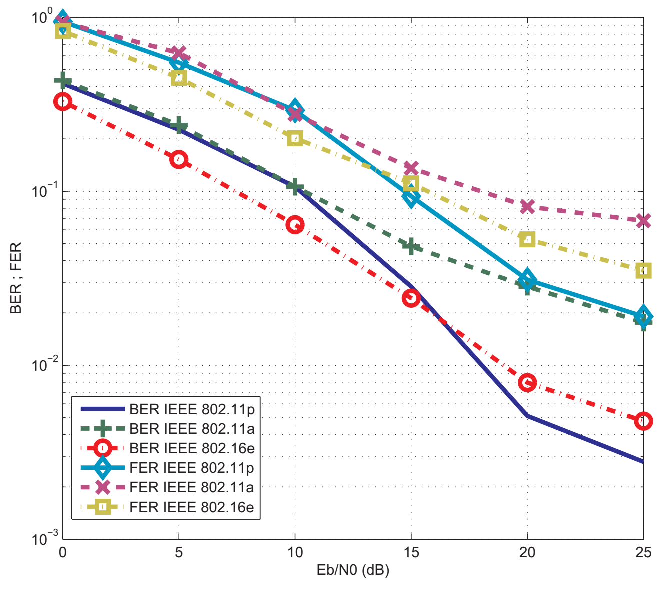

Fig. 14. Performance comparison when transmitting over *RTV-Expressway*.

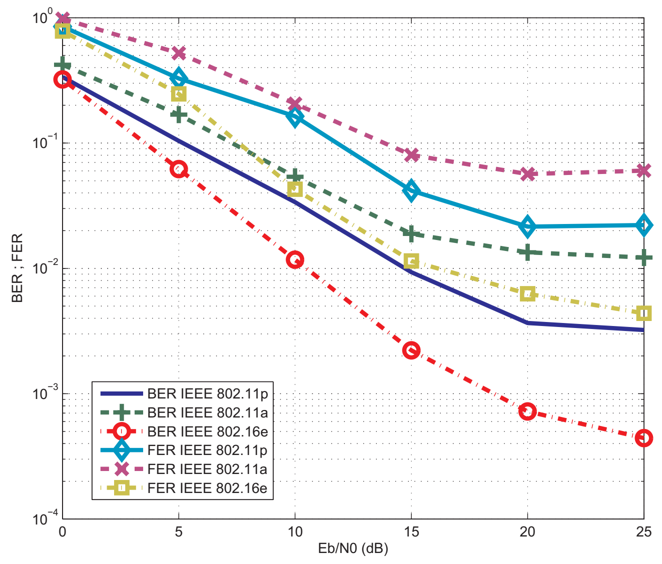

Fig. 15. Performance comparison when transmitting over *VTV-Expressway Same Direction with Wall*.

transceivers implemented the MRC technique. In the case of MIMO receivers, symbols were decoded if needed (an Alamouti decoder was used for 2×2 systems) and an ML detector was applied.
A maximum of 10,000 48-bit FEC blocks were averaged for different SNR values (the simulation stopped for each SNR value when 100 erroneous FEC blocks were detected).

5.2.1 Performance in vehicular channels

For the sake of space, we only show and compare the results for three different vehicular channels (Figs. 16 to 18), which give a good overview of the performance drawn by the implemented multi-antenna IEEE 802.11p transceivers. As expected, it can be observed that SIMO/MIMO transceivers outperform SISO systems.

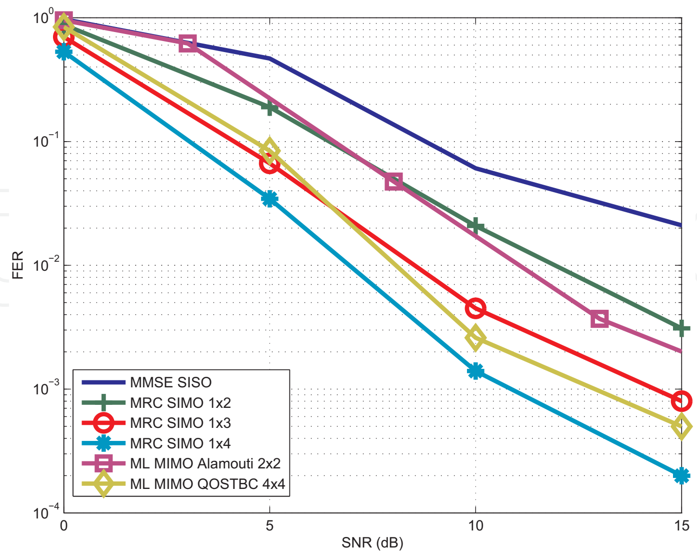

Fig. 16. FER performance for *RTV-Urban Canyon*.

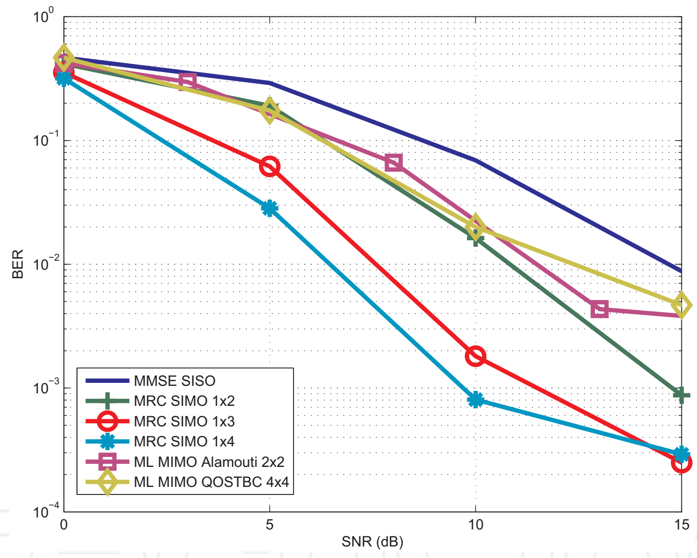

Fig. 17. FER performance for *VTV-Expressway Oncoming*.

In the emulated conditions, SIMO systems seem to offer the best tradeoff between hardware complexity and performance. In all environments, the SIMO 1×4 system obtains the best performance and in almost every vehicular channel the SIMO 1×3 is the transceiver that gets the second best results.
SIMO 1×2 and MIMO 2×2 attain similar BER/FER performance, while the results obtained by the MIMO 4×4 strongly depend on the vehicular channel. Such performance difference is related to two main factors: the channel estimation misbehavior in presence of high Doppler frequencies and the presence of a low overall K factor.
On the one hand, as mentioned in Section 4.1.1, due to the particular channel estimation technique implemented, a high Doppler frequency leads to a bad channel estimation, what is really harmful for the performance of multi-antenna systems. In fact, if we rank the

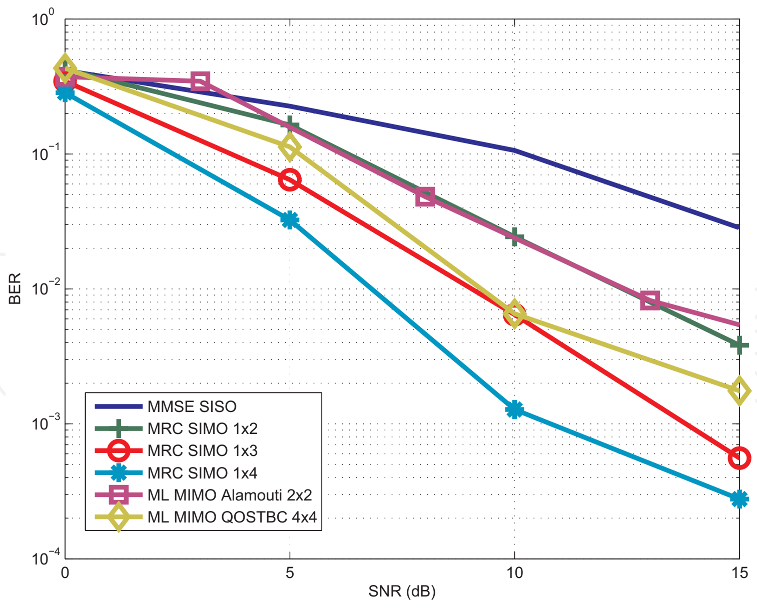

Fig. 18. FER performance for *RTV-Expressway*.

three channels by their LOS Doppler frequency (654 Hz for *RTV-Urban Canyon*, 770 Hz for *RTV-Expressway* and 1452 Hz for *VTV-Expr. Oncoming*), it is apparent that the lower the Doppler, the better the performance.
On the other hand, if we rank the channels by their overall K factor (6.7 dB for *RTV-Urban Canyon*, 4.3 dB for *RTV-Expressway* and -3.6 dB for *VTV-Expr. Oncoming*), we can conclude that the higher overall K factor, the better the performance.
Furthermore, the results shown in Table 6, corresponding to the required SNR to obtain a FER of 10%, give a good idea about the performance of each transceiver and confirms our previous statements. The maximum differences in SNR occur when comparing the SISO and the SIMO 1×4 systems, ranging between 5.73 dB (for *RTV-Urban Canyon*) and 8.96 dB (for *RTV-Expressway*). That is, a SIMO 1×4 system requires between 4 and 8 times less power than a SISO system to obtain the same FER.

Channel	SISO	SIMO 1×2	SIMO 1×3	SIMO 1×4	MIMO 2×2	MIMO 4×4
RTV-Urban Canyon	8.78	6.44	4.14	3.05	6.56	4.62
RTV-Expressway	14.71	10.30	7.35	5.75	10.07	8.62
VTV-Expressway Oncoming	13.17	10.05	7.19	5.81	10.82	11.91

Table 6. SNR (dB) required to obtain a FER of 10% in each vehicular channel.

Finally, it must be pointed out that multiple-antenna transceivers obtain their largest SNR gains when transmitting over channels that assume high vehicular speeds (i.e. the scenarios located in expressways), achieving gains from 7.36 dB to 8.96 dB with respect to SISO systems. This is a quite interesting result, since it means that mobile communications performed in high speed scenarios can be greatly improved by placing antenna arrays along the roadside and/or in vehicles and using relatively simple space-time diversity techniques.

6. Conclusions

We have presented a performance evaluation tool for wireless standards suitable for vehicular communications. It consists of a software testbed and a flexible, low-cost, FPGA-based

channel emulator. We have detailed the way we employed rapid-prototyping techniques for building both the testbed and the channel emulator. The resulting evaluation system has been used to assess the performance of the PHY layer of IEEE 802.11e (Mobile WiMAX) and IEEE 802.11p/a over representative situations where vehicular communications can take place.
In addition, we have shown how this performance evaluation system has been upgraded for multiple-antenna transceivers. The different hardware optimizations we have performed during the design process of our MIMO channel emulator have been also explained. Finally, we have presented interesting performance evaluation results for SISO, SIMO and MIMO transceivers over reference vehicular channel models.

7. Acknowledgements

This work has been supported by Xunta de Galicia through grants 09TIC008105PR and 10TIC003CT, by Ministerio de Ciencia e Innovación of Spain through grant TEC2010-19545-C04-01 (COSIMA project) and by Ministerio de Ciencia e Innovación of Spain and FEDER funds of the EU under grant CSD2008-00010 (COMONSENS project).

8. References

Acosta-Marum, G. (2007). Measurement, modelling and OFDM synchronization for the wideband mobile-to-mobile channel. *Doctoral Thesis*, May 2007.

Acosta-Marum, G.; Ingram, M. A. (2007). Six time- and frequency-selective empirical channel models for vehicular wireless LANs, *Proceedings of VTC Fall*, ISSN 1090-3038, Baltimore, USA, Oct. 2007.

Alamouti, S. M. (1998) A simple transmitter diversity scheme for wireless communications, *IEEE Journal on Selected Areas of Communications*, ISSN 0733-8716, vol. 16, No. 8 (1998) 1451-1458.

Alimohammad, A.; Fard, S. F.; Cockburn, B. F.; Schlegel, C. (2008). An accurate and compact Rayleigh and Rician fading channel simulator, *Proceedings of VTC Spring*, ISSN 1550-2252, Singapore, May 2008.

Angelakis, V.; Kossifidis, N.; Papadakis, S.; Siris, V.; Traganitis, A. (2008). The effect of using directional antennas on adjacent channel interference in 802.11a: Modeling and experience with an outdoors testbed, *Proceedings of WiOPT2008*, ISBN 978-963-9799-18-9, pp. 24-29, 978-963-9799-18-9, Berlin, Germany, April 2008.

Arthaber, H.; Schuberth, C. (2009). A channel emulator for UHF RFID systems, *Proceedings of IEEE Radio and Wireless Symposium*, ISBN 978-1-4244-2698-0, pp. 518-521, 978-1-4244-2698-0, San Diego, USA, Jan. 2009.

ASTM Intl. (2003). *Standard specification for telecommunications and information exchange between roadside and vehicle systems - 5 GHz band Dedicated Short Range Communications (DSRC), Medium Access Control (MAC) and Physical Layer (PHY) specifications*, E2213-03, Sep. 2003.

Dassatti, A.; Masera, G.; Nicola, M.; Concil, A.; Poloni, A. (2005). High performance channel model hardware emulator for 802.11n, *Proceedings of IEEE International Conference on Field-Programmable Technology*, ISBN 0-7803-9407-0, pp. 303-304, Singapore, Dec. 2005.

Eslami, H.; Tran, S. V.; Eltawil, A. M. (2009) Design and implementation of a scalable channel emulator for wideband MIMO systems, *IEEE Transactions on Vehicular Technology*, ISSN 0018-9545, vol. 58, No. 9 (2009) 4698-4709.

Faseth, T.; Winkler, M.; Schuberth, C.; Arthaber, H.; Magerl, G. (2010). Design and implementation of a wireless link coupled channel emulator for DSRC wireless systems, *Proceedings of IEEE MTT-S International*, ISSN 0149-645X, pp. 1632-1635, Anaheim, USA, May 2010.

Fernández Caramés, T. M.; García Naya, J. A.; González-López, M.; Castedo, L. (2008). FlexVehd: a flexible testbed for vehicular radio interfaces, *Proceedings of ITST*, ISBN 978-1-4244-2857-1, Phuket, Thailand, Oct. 2008.

Fernández-Caramés, T. M.; González López, M.; Castedo, L. (2010). FPGA-based vehicular channel emulator for real-time performance evaluation of IEEE 802.11p transceivers, *EURASIP Journal on Wireless Communications and Networking* (2010).

Foschini, G.; Gans, M. (1998). On limits of wireless communications in a fading environment when using multiple antennas, *Wireless Personal Communications*, ISSN 0929-6212, vol. 6, No. 3 (1998) 311-335.

Ghazel, A.; Boutillon, E.; Danger, J.; Gulak, G.; Laamari, H. (2003). Design and performances analysis of high speed AWGN communication channel emulator, *Analog Integrated Circuits and Signal Processing*, ISBN 0-7803-7080-5, vol. 34, pp. 133-142, Feb. 2003.

Hu, S.; Wu G. W.; Guan, Y. L.; Law, C. L; Yan, Y.; Li, S. (2007). Development and performance evaluation of mobile WiMAX testbed, *Proceedings of IEEE Mobile WiMAX Symposium*, ISBN 1-4244-0957-8, pp. 104-107, Orlando, USA, Mar. 2007.

Hwang, J.; Lin, K.; Li, J.; Deng, J. (2007). Fast FPGA prototyping of a multipath fading channel emulator via high-level design, *Proceedings of ISCIT*, ISBN 978-1-4244-0976-1, Sidney, Australia, Oct. 2007.

IEEE 802.11 (2007), *IEEE Standard 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE (2007).

IEEE 802.16 (2009), *Air interface for fixed broadband wireless access systems*, IEEE, 2009.

IEEE (2010), *IEEE Standard Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 6: Wireless Access in Vehicular Environments* (2010).

Jafarkhani, H. (2000) A quasi-orthogonal space-time block code, *IEEE Transactions on Communications*, ISSN 0090-6778, vol. 49, No. 1 (2001) 1-4.

Nieto, X.; Ventura, L.M.; Mollfulleda, A. (2006). GEDOMIS: a broadband wireless MIMO-OFDM testbed, design and implementation, *Proceedings of TRIDENTCOM2006*, ISBN 1-4244-0106-2, Barcelona, Spain, Mar. 2006.

Ren, F.; Zheng, Y. R. (2010) A novel emulator for discrete-time MIMO triply selective fading channels, *IEEE Transactions on Circuits and Systems*, ISSN 1549-8328, vol. 57, No. 9 (2010) 2542-2551.

Rugini, L.; Banelli, P.; Leus, G. (2005). Simple equalization of time-varying channels for OFDM," *IEEE Communications Letters*, ISSN 1089-7798, vol. 9, No. 7 (July 2005), 619-621.

Telatar, I. E. (1999). Capacity of multi-antenna Gaussian channels, *European Transactions on Telecommunications*, ISSN 1541-8251, vol. 10, No. 6 (1999) 585-595.

Wang, T.; Liao, C. H.; Chiueh, T. D. (2007) A real-time digital baseband MIMO channel emulation system, *Proceedings of ISCAS*, ISBN 1-4244-0920-9, New Orleans, USA, May 2007.

Wang, Y.; Ahmed, A.; Krishnamachari, B.; Psounis, K. (2008) IEEE 802.11p performance evaluation and protocol enhancement, *Proceedings of Intl. Conf. on Vehicular Electronics and Safety*, ISBN 978-1-4244-2359-0, Columbus, USA, Sep. 2008.

Zhan, Z.; Jun, J.; Ping, Z.; Xin, W. (2009) A generalized hardware implementation of MIMO fading channels, *Proceedings of ISCIT*, ISBN 978-1-4244-4521-9, Incheon, Korea, Sept. 2009.