Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2010, Article ID 607467, 18 pages
doi:10.1155/2010/607467

Research Article

FPGA-Based Vehicular Channel Emulator for Real-Time Performance Evaluation of IEEE 802.11p Transceivers

T. M. Fernández-Caramés, M. González-López, and L. Castedo

Department of Electronics and Systems, University of A Coruña, A Coruña 15071, Spain

Correspondence should be addressed to T. M. Fernández-Caramés, tmfernandez@udc.es

Received 1 August 2009; Revised 17 October 2009; Accepted 15 December 2009

Academic Editor: Markus Rupp

IEEE 802.11p is one of the most promising future wireless standards due to the increasing demand of vehicular communication applications. At the time of writing, the document of the standard is in draft and much research is still required to study and improve the performance of transceivers in common vehicular scenarios. In this paper, we present a framework to evaluate the PHY layer of IEEE 802.11p systems in realistic situations. We detail the design and implementation of an FPGA-based real-time vehicular channel emulator. Contrarily to commercial emulators, ours is cheap, very flexible, and reconfigurable. We show its capabilities by evaluating performance in different high-speed scenarios. We also study the importance of coding and the benefits of using IEEE 802.11p instead of IEEE 802.11a in vehicular environments. Towards this aim, we developed a reference IEEE 802.11p PHY transceiver software model that can be taken as a convenient starting point for transceiver design.

1. Introduction

Wireless communications between moving vehicles (Vehicle-To-Vehicle, TV or V2V) and from vehicles to infrastructure (V2I or Roadside-to-Vehicle, RTV) have recently received a lot of attention due to the increasing demand for solutions to tackle critical issues such as vehicular safety and to provide services like traveling information, payment automation or infotainment. Related to these services, a huge amount of vehicular applications has been proposed: collision prevention, accident warnings, hazardous vehicles monitoring, traffic jams avoidance, road works alerts, route recommendation, weather forecast, toll collection, payment in parking facilities and gas stations, mobile internet access at vehicular high speeds, tourist information, and so forth. All these applications constitute the realm of Intelligent Transportation Systems (ITS).

ITS relies on the existence of a suitable wireless communication architecture and the IEEE 802.11p standard is the best positioned to be the reference of vehicular communications in the near future. The standard (still in draft 5.0 at the time of writing) is based on the specifications given in [1]. The Medium Access Control (MAC) and Physical (PHY) layers are very similar to those used in the wireless local area network standard IEEE 802.11a [2]. Slight changes are included to allow operating in high delay spread scenarios, which are typically present in urban canyons.

To assess the performance of IEEE 802.11p systems, it is desirable to evaluate it in realistic situations. The tests can be performed directly in a vehicle, driving through different environments. However, this is a time-consuming task and the experiments can be affected by unintended side effects that may be uncontrollable. It is more convenient to use a hardware channel emulator and measure the performance inside a testing lab with a testbed platform as in [3–5]. Nevertheless, commercial channel emulators are usually very expensive and may not offer enough flexibility when configuring the wireless channel parameters.

Motivated by these observations, we decided to build a cheap and flexible alternative to evaluate the performance of an IEEE 802.11p system in vehicular environments: we implemented a software transceiver and designed an emulator that can be reconfigured fast and easily, providing enough flexibility to implement customized wireless channel models. To achieve these goals, we made use of rapid-prototyping software for developing the evaluation framework: Simulink for the transceiver and Xilinx System Generator for the FPGA (Field-Programmable Gate Array) -based vehicular channel emulator. System Generator is specially useful because it allows us to design the channel emulator much faster than using conventional hardware description languages like VHDL or Verilog.

The implemented channel models are based on [6, 7], where the authors obtain channel models from data acquired during a measuring campaign carried out in a metropolitan area at a frequency of 5.9 GHz. Six different situations are considered, including measures in common vehicular environments such as a highway, an urban canyon and a suburban area.

The remaining of this article is organized as follows. Section 2 includes a comprehensive review of the current state of the art of IEEE 802.11p transceivers and ad-hoc channel emulators. Section 3 shows the theoretical and practical design of our FPGA-based channel emulator. Section 4 briefly describes our reference IEEE 802.11p transceiver software model and Section 5 is dedicated to experimental IEEE 802.11p performance evaluation using the vehicular channel emulator. Finally, Section 6 is devoted to conclusions.

2. State of the Art

IEEE 802.11p is an amendment to the IEEE 802.11 standard that addresses the challenges that arise when providing wireless access in vehicular environments. The standard has its origin in 1999, when the Dedicated Short Range Communications (DSRC) spectrum band, a band of 75 MHz at 5.9 GHz, was allocated in the United States. In that moment, a need to define a common framework to develop hardware and applications for the DSRC band emerged. A deep description of the standard is beyond the scope of this paper, but we encourage the interested reader to take a look at the excellent overviews given in [8, 9].

2.1. IEEE 802.11p: Current Standard and Transceivers. The main advantage of IEEE 802.11p over IEEE 802.11a when overcoming the effects of vehicular channels, is its bandwidth. The 20 MHz bandwidth used in IEEE 802.11a is reduced to only 10 MHz in IEEE 802.11p. Thus, the OFDM (Orthogonal Frequency Division Multiplexing) symbols are longer in the time domain and the system can deal with large delay spreads, being able to avoid ISI (Inter Symbol Interference). The practical implementation is straightforward, since there is only the need to double all the OFDM timing parameters used by the IEEE 802.11a devices.

Due to the novelty of the MAC layer, most of the evaluations of the standard in the literature are aimed at studying it, by using simulators like `ns-2` [10] or following analytical approaches [11]. There are also software implementations of the standard: some of them describe generic transceivers [12], but the majority are focused on the performance of the receiver by using techniques for getting better channel estimations [13–16] or improved decisions [17].

Before the development of the IEEE 802.11p amendment there were several proposals to support DSRC applications. Most of the already available prototypes were designed for the particular regulations of countries like Japan [18] or Korea [19], or for the European standard [20]. As the draft of the IEEE 802.11p has been evolving, more prototypes have arisen, such as [21], that the authors declare as the first platform where IEEE 802.11p and IEEE 1609 (that defines the upper MAC layer and the network pulse session layer) have been implemented.

Finally, DSRC-based processing modules have been developed as well. For instance, in [22], a fast OFDM modulator/demodulator allows performing an IFFT/FFT in less time than the symbol interval (8 μs) and in [23] a software-defined FPGA-based channel simulator was designed for testing baseband transceivers of different wireless communication systems, showing an example of its use with an IEEE 802.11p system.

2.2. Traditional and Ad-Hoc Channel Emulators. 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 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 [24], Rhode & Schwarz [25], Azimuth Systems [26], Agilent [27]. 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 Multiple-Input Multiple-Output 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 while setting channel configuration parameters. To tackle these issues a number of low-cost ad-hoc channel emulators have recently been proposed.

To develop a low-cost and easily-reconfigurable channel emulator, different technologies can be assessed. Due to real-time constraints, microcontrollers and affordable DSPs (Digital Signal Processors) are not valid. Among the rest of the semiconductor technologies, three of them are specially suitable for implementing a channel emulator: CPLDs (Complex Programmable Logic Devices), ASICs (Application-Specific Integrated Circuits) and FPGAs. Although ASICs offer superior performance, their developing time is too high for a prototype (they are mainly used for final products). Thus, the choice has to be made between CPLDs and FPGAs. The latter are the most commonly used due to two reasons:

(i) FPGAs own specific resources that are more adequate for implementing a channel emulator, such as counters or arithmetic operator blocks.

(ii) Although CPLDs can execute tasks at a faster rate, the maximum allowed complexity of the designs is inferior to that offered by an FPGA.

Therefore, most of the proposed ad-hoc emulators are based on FPGA technology. Moreover, during the last years FPGAs have become a more attractive tool to researchers: their unitary cost has been dramatically reduced and new implementation tools make the formerly tedious tasks of programming and design verification easier.

Some examples of FPGA-based channel emulators are described in [28–30]. In [28] 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.

Similarly, in [29], 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. The final designs use only between 2% and 5% of the FPGA slices and are able to generate 201 million 16-bit complex-valued fading samples per second.

Finally, [30] presents a baseband multipath fading channel emulator implemented on a Virtex-IV using 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. After the verification stage, the emulator was downloaded to the FPGA board and tested with a baseband QPSK signal, achieving results similar to those obtained through MATLAB simulations.

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 to obtain 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 that enable to build complex designs easier and faster.

The second problem is precisely 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 [30] 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.

Several papers have also identified and quantified the effects related to the cited drawbacks. For instance, in [31], the authors used rapid prototyping tools (Generic C, Matlab and Simulink) to accelerate the implementation process. Besides, in order to save space, they limited their channel emulator to 8 or 16 paths (depending on the channel model). To quantify the error introduced by this reduction of the number of paths, they compared the average RMS (Root Mean Squared) delay spread of the original and the reduced model, finding a lower delay profile in the latter, what caused an error in the channel emulation.

Another interesting paper is [32], which presents a MIMO development platform made of scalable processing boards that contain TI C6414 DSPs and Xilinx Virtex 2 FPGAs. The article is not focused on saving FPGA resources, but it constitutes a good example of how to shorten the development cycle using rapid prototyping and hardware-in-the-loop techniques. In fact, the authors were able to build fast a MIMO channel emulator based on the COST 259 recommendations, splitting the processing into two stages: the impulse response of the channel resides in the DSP, while the convolution of the transmitted signal with the impulse response is performed in the FPGA.

The last example we would like to mention is [33]. There, the authors implemented a configurable multipath fading channel emulator using Matlab and Xilinx System Generator. They built a resource-efficient white Gaussian noise generator (it occupies less than 450 logic cells of a Xilinx Virtex-IV) and they made use of IIR Doppler filters instead of FIR filters in order to avoid the utilization of a large number of taps, typically required by the latter (however, their IIR filters are less accurate when representing the original Doppler spectrum).

The vehicular emulator described in this paper addresses the two mentioned drawbacks: we use System Generator to develop the channel emulator faster than using VHDL and we optimize our design in order to be able to implement a twelve-path channel emulator, even leaving space for future extensions.

3. Real-Time FPGA-Based Vehicular Channel Emulator

3.1. Implemented Vehicular Channel Models. The implemented channel models are based on the excellent work in [6, 7], that is mainly based on a measurement campaign at 5.9 GHz carried out in the spring of 2006 in Atlanta, Georgia. Thanks to these measurements the authors were able to obtain channel models for six different environments that cover some of the most common situations where VTV and RTV communications may take place. The six selected environments can be grouped in three major scenarios:

(i) urban canyons, with dense and tall buildings, where vehicles move at speeds between 32 km/h and 48 km/h,

(ii) suburban expressways, with moderately dense, low-story buildings, where the measurement speed was approximately 105 km/h,

(iii) suburban surface streets, with moderately dense, low-story buildings, where the driving speed was between 32 km/h and 48 km/h.

In these environments, the measurements were taken in two different ways depending on the situation. For VTV communications, data was exchanged between vehicles with 8 dBi omnidirectional antennas mounted magnetically on the roof. For RTV channels, the communications were performed between a vehicle with one of the mentioned omnidirectional antennas and a monopole or half-dome antenna mounted on a pole by the side of the road at a height of 6.1 meters.

The main characteristics of the vehicular scenarios are the following (for a more detailed description of the measurements see Section 5.2 in [7]).

3.1.1. VTV-Expressway Oncoming. Two oncoming vehicles approached at high speeds in a stretch of highway without middle wall. Both vehicles were synchronized to enter the highway at the same time and then each vehicle accelerated to 105 km/h. During the modelling, the authors only considered the measurements taken when the vehicles were at a distance of 300–400 meters, where they observed a fixed LOS Doppler shift in the first path.

3.1.2. VTV-Urban Canyon Oncoming. Two oncoming vehicles at 32–48 km/h approached in an urban canyon. During the modelling, only the data segments obtained with a separation of 100 meters were selected.

3.1.3. RTV-Suburban Street. A vehicle at 32–48 km/h received packets from a monopole antenna placed next to a suburban street intersection. The vehicle approached the intersection from any of the four possible directions. Only the data acquired at 100 meters were taken into account in the model. Besides, the data segments with a signal level lower than a threshold were discarded from the modelling process.

3.1.4. RTV-Expressway. The communications were performed between a half-dome antenna mounted on a pole off the side of an expressway and a vehicle at 105 km/h. The measurements were taken at a distance of 300–400 meters as the vehicle approached from both directions of the expressway.

3.1.5. VTV-Expressway Same Direction with Wall. For this scenario, the data were obtained in different locations along several expressways where there was a middle wall between oncoming lanes. In all these situations, two vehicles at 105 km/h carried out communications when travelling in the same direction. Only the measurements at a distance of 300–400 meters were used during the modelling of the channel.

3.1.6. RTV-Urban Canyon. A transmitting monopole antenna was mounted on a pole near an urban intersection and a vehicle was approaching at a speed of 32–48 km/h. For the model, only the measurements at a distance of 100 meters were taken into account. To avoid deep fades, the data with a signal level lower than an arbitrarily set threshold were not used in the modelling of the channel.

The values of the parameters of the vehicular models implemented in our channel emulator are clearly summarized at the end of [6]. 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. Furthermore, the distances were also made consistent with 100 meters for surface streets and approximately 400 meters for expressways, what seem to be typical ranges where vehicles carrying IEEE 802.11p transceivers are expected to operate.

It must be mentioned that the authors have also proposed a modification of the *RTV-Expressway* channel with Doppler frequencies consistent with a speed of 200 km/h [7]. We have not implemented it because of the similarities with the model of the same scenario at 140 km/h and to carry out a fair comparison at the same speed among the different expressway channels.

The key characteristics of the vehicular channels are summarized in Table 1. 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.

Table 1 also gives an idea of the complexity involved in the implementation of the 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 at 10 MHz, it is not possible to implement the six channels into a single FPGA: as it can be seen in Table 2, 76% of the slices are occupied in the best case and, regarding the rest of the resources (flip-flops, LUTs (Look-Up Tables), FIFO16/RAMB16 memory blocks and DSP48 blocks), between 19% and 65% are used. Thus, six independent. bit files are generated, though in practice, only three different FPGA designs are needed due to the model similarities.

(i) One design is exclusively dedicated to the channel *VTV-Expressway Oncoming* which is the only one with eleven paths.

(ii) 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.

(iii) One design for the other four channels, which differ in their configuration parameters but share all their FPGA resources.

Finally, it should be mentioned that all models experience the same latency in our channel emulator. The latency is the time delay between the input and output signals. We have measured this parameter and obtained the value of 2.813 ms. The latency parameter is important when testing real transceivers although in this paper this information is not necessary, since the channel emulator is evaluated using a software transceiver.

Table 1: Main characteristics of the vehicular models.

Vehicular Channel	Distance TX-RX (m)	Speed (km/h)	Path Modulation (number of paths)	Maximum Delay Spread (ns)	Rician K (dB)	Overall 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	7.2	6.7	720	994	654
RTV-Express way	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, 4

Table 2: General parameters and resources occupied by the vehicular channel emulator.

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%

3.2. Theoretical Model. For the generation of each channel coefficient at the ith path in the time instant t, we used the following model:

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

where we have

(i) K_i: Rice factor of the ith path,

(ii) P_i: power of the ith path,

(iii) $h_w(i,t)$: 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,

(iv) $\overline{h}(i,t)$: contribution of the line-of-sight (LOS) component to the i-th path at the time instant t. It is determined by

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

where we have

(i) $f_{D,i}$: maximum Doppler spread of the ith path.

(ii) θ_i: angle of arrival of the ith path.

(iii) ϕ_i: phase of the LOS component of the ith 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 Figure 1, the emulator only needs five parameter blocks.

(i) `Sigma` block contains the power factors of the NLOS components:

$$\sigma = \sqrt{\frac{P_i}{(K_i+1)}}. \tag{3}$$

(ii) `A` block holds the power factors of the LOS components:

$$A = \sqrt{\frac{K_i P_i}{(K_i+1)}} \tag{4}$$

(iii) `FrequencyLOS block` contains part of the exponent of $\overline{h}(i,t)$:

$$f_{\text{LOS}} = 2\pi f_{D,i}\cos(\theta_i.) \tag{5}$$

(iv) `PhaseLOS` block is simply ϕ_i.

(v) `Taps delay` block holds the normalized delays of the different paths. Note that the delays need to be normalized since the FPGA's sampling period is 100 ns, but the model has delay values that differ in only one nanosecond (for instance, paths 6 to 8 of *VTV-Expressway Oncoming* are delayed 200 ns, 201 ns and 202 ns, resp.). The authors of the model justify this in [6, Section III]: to avoid problems with their commercial channel emulator (Spirent SR5500), paths comprising a single tap were separated in delay by one nanosecond. Instead, our FPGA-based channel emulator does not present such kind of problems and, thus, the one-nanosecond artificial delays in the models are suppressed. That is, the previously mentioned delays of 200 ns, 201 ns and 202 ns, will occur at the same time instant (200 ns).

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.

(i) It contains a Virtex IV (XC4VSX35-10FF668) that allows using Xtreme-DSP slices of up to 400 MHz.

(ii) It has 2 14-bit ADCs able to work up to 105 MS/s and 2 14-bit DACs that can run up to 160 MS/s.

(iii) Dedicated internal clock up to 105 MHz, although it can use an external clock.

(iv) 4 MB of 166 MHz ZBT-RAM.

(v) Possibility to connect the kit using 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 an IEEE 802.11p transceiver that interacts with the vehicular emulator, which is running on the FPGA.

3.4. Hardware Setup to Evaluate Physical Transceivers. Although in this paper we describe and show performance results obtained using a software transceiver, it is straightforward to connect the emulator with a hardware testbed or with commercial transceivers. Depending on the transceiver outputs, there are two ways of setting up the hardware.

On the one hand, if the transceiver provides IF (Intermediate Frequency) or RF outputs, we need a hardware setup like the one depicted in Figure 2. The transmitter is connected to a downconverter, which obtains the IQ representation of the signal, that is sent to the emulator ADCs. After the emulator applies the channel model in real time to the IQ digitized signal, the output samples are forwarded to the emulator DACs. Then, the analog signal is sent to the upconverter, which, finally, transmits it to the receiver.

Since the emulator currently does not perform down or up conversions, additional hardware is required for this purpose. In the case shown in Figure 2, both operations are carried out by commercial devices. A spectrum analyzer equipped with a downconversion module (Agilent E4404B) is used to downconvert signals of up to 6.7 GHz. The upconversion is performed by using a vector signal generator (Agilent E4438C) that receives the IQ signal from the emulator and upconverts it to frequencies up to 6 GHz.

On the other hand, if the physical transceiver is able to provide the emulator with IQ outputs, then the up and downconversion stages are not needed. The signals would be acquired by the emulator ADCs and sent directly from the DACs to the receiver.

3.5. FPGA Design Overview. Figure 1 shows a general view of the hardware design. Several blocks represented in such figure contain sub-blocks which are shown throughout this paper: the `Coefficient Generator` block includes Figures 3, 4 and 5, the `Interpolator` block contains a number of interpolators like the one shown in Figure 6 and the FIR block is shown in Figure 7.

Notice that the design presented in Figure 1 is optimized for a specific channel (*VTV-Expressway Oncoming*) in order to minimize the amount of required hardware, though the rest of channels share most of the hardware resources, as it was mentioned at the end of Section 3.1.

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.

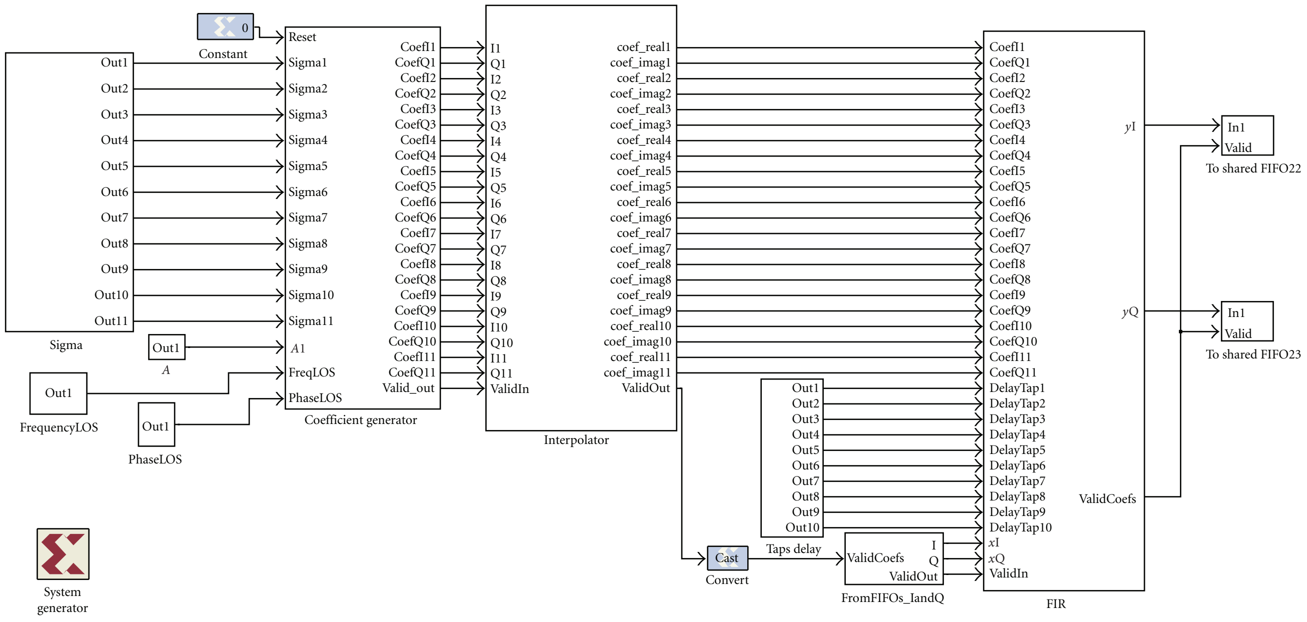

Figure 1: General view of the System Generator model optimized for the vehicular channel *VTV-Expressway Oncoming*.

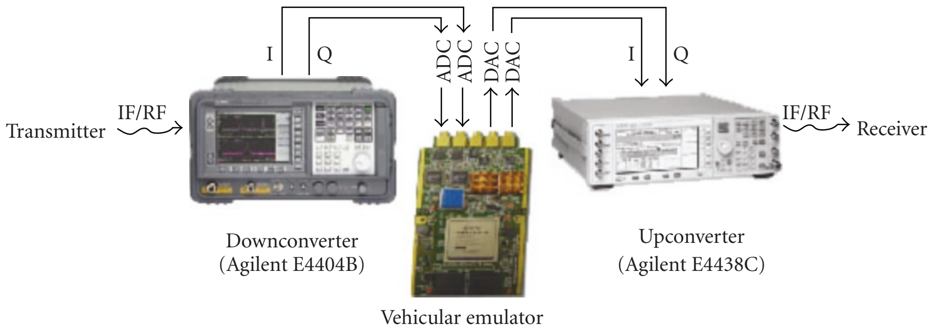

Figure 2: Hardware set-up for testing an IEEE 802.11p physical transceiver with IF/RF outputs on our real-time channel emulator.

3.5.1. Acquisition of the Channel Parameters. The generation of the configuration parameters of the vehicular channel is performed offline to reduce the amount of used hardware and due to the fact that these parameters remain constant throughout the emulation. The parameters are stored into registers that will be read by the FPGA-based emulator. Notice that every channel has its own peculiarities and all the parameters equal to zero can be removed from the design 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.

3.5.2. White Gaussian Noise Generation. To obtain NLOS coefficients, first, 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 twenty four real values of such noise samples that will be filtered depending on the Doppler shift experienced by each path.

Figure 3 shows the way we reduce the amount of FPGA resources consumed by the 24-output Gaussian noise generator: instead of using twenty four independent WGNG blocks, we multiplex in time the samples produced by only one WGNG block. This optimization is crucial since each WGNG block consumes an important amount of FPGA resources.

For this specific vehicular emulator, the use of only one WGNG block together with a multiplexor results in a functionality that is equivalent to making use of twenty four parallel WGNG blocks. Since each WGNG block runs at 10 MHz, using a twenty five output multiplexor like the one shown in Figure 3 (twenty five is the integer divider of 10 MHz closest to twenty-four), produces 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 2).

To quantify the amount of resources saved thanks to this optimization, we isolated the 24-output Gaussian noise generator in a different design (to measure the resources that it requires) and we compared the optimized and the non-optimized versions (the latter uses twenty-two WGNG blocks

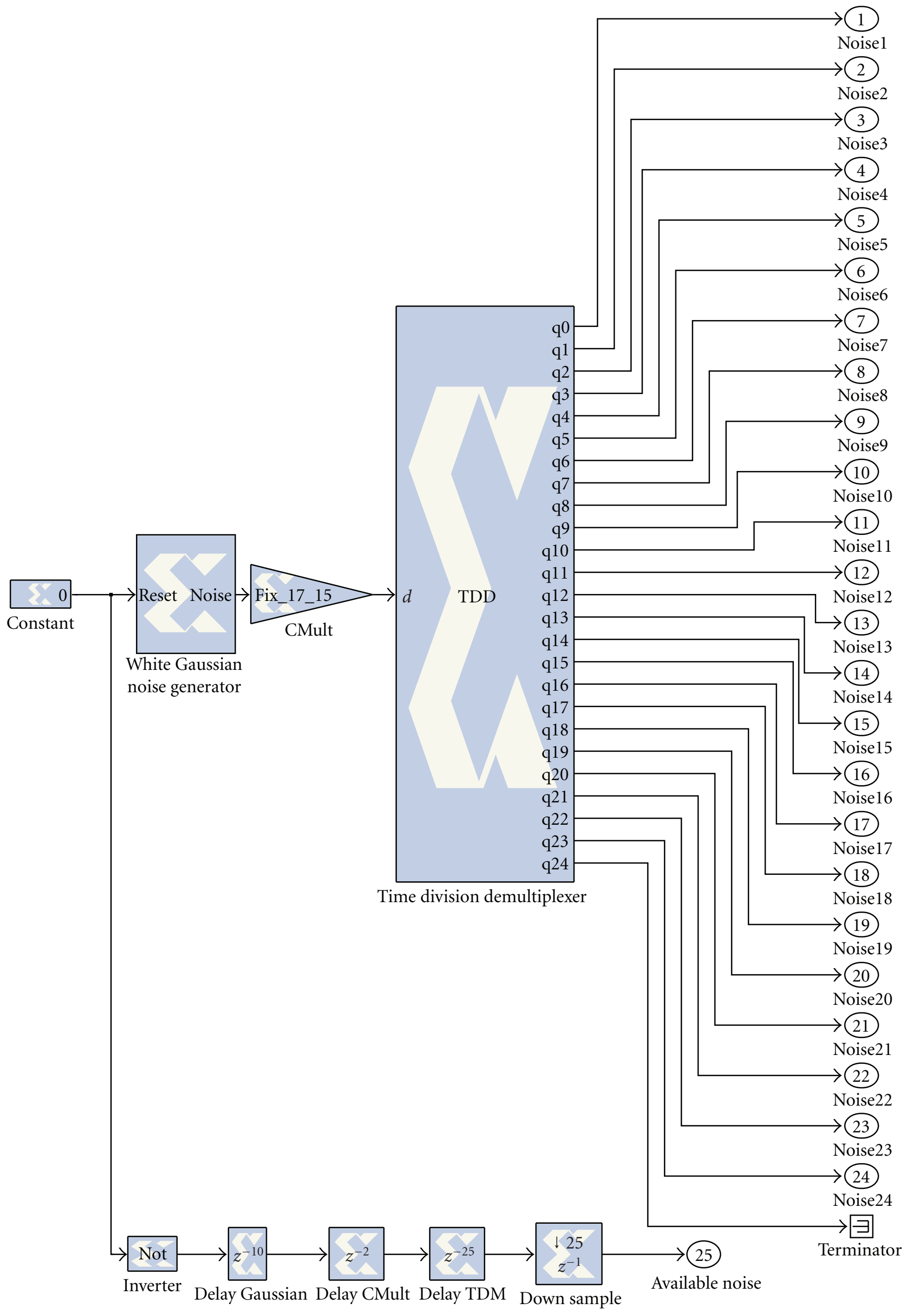

FIGURE 3: Resource-efficient 24-output Gaussian noise generator.

as if it was aimed at emulating the channel *VTV-Expressway Oncoming*). We would like to show the real savings but, unfortunately, our PC (Intel Core 2 Duo E8500@4.33 GHz, 2 GB of RAM, Windows XP) always runs out of memory while performing the synthesis of the non-optimized model due to the large amount of resources it requires. However, it is possible to obtain an estimation of the required resources using System Generator's Resource Estimator block. Table 3 shows the estimated amount of resources occupied by both versions. It can be clearly observed that WGNG blocks should be used carefully in order to avoid wasting FPGA resources.

If we needed to reduce even more the number of occupied resources, it would be possible to generate the channel coefficients in MATLAB and then transfer them to the FPGA. However, this solution is not desirable due to the following reasons.

(i) 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.

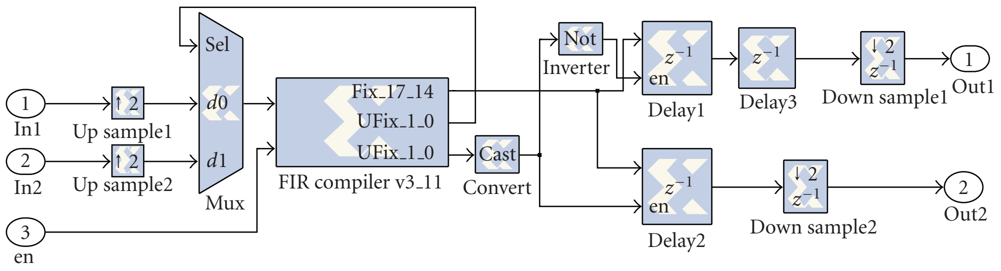

FIGURE 4: Optimized blocks for applying the Doppler Spectrum.

TABLE 3: Estimated savings due to optimizing the Gaussian generator.

Resource type	Optimized 24-output Gaussian generator	Non-optimized 24-output Gaussian generator	Estimated Savings (%)
Slices	1347	20355	93.3%
Flip-flops	1729	19607	91.2%
RAM Blocks	8	176	95.4%
LUTs	997	21850	95.4%
Embedded Multipliers	4	88	95.4%

(ii) 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.

3.5.3. 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 1 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* [7].

Figure 4 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. If we desired to build a more generic vehicular emulator, it would be possible to reload dynamically the coefficient sets, but each filter block would need additional control lines.

To reduce to a half the required hardware, we exploit the fact that the real and the imaginary parts of the filter can be used twice for each path to perform the complex FIR filtering (see Figure 5). This optimized block can be seen in Figure 4, which is contained under the block *Doppler Filter* shown in Figure 5.

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. Like we did in the previous subsection with the Gaussian generators, we isolated the optimized and non-optimized versions of the filter block and we obtained the size they occupy. In this case, we could synthesize both designs and obtain the actual hardware occupation without using the Resource Estimator block. It can be observed in Table 4 that, 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.

Finally, LOS Doppler has also to be taken into account in the vehicular channels and must be applied to each Rician path according to (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 [6], 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.

3.5.4. Interpolation. After adding the LOS and NLOS components according to (1) (see Figure 5), the coefficients must adapt their rate to the rate of the incoming signal (i.e., the signal from the IEEE 802.11p transmitter that 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 2). 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 to interpolate between two already generated coefficients. For doing this, we use two cascaded linear interpolators, each one having its own interpolation factor. The product of both interpolation factors gives the global interpolation rate of the corresponding channel (shown in Table 2).

Every single cascaded linear interpolator (see Figure 6) obtains p intermediate values following the following steps.

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

Resource type	Optimized Doppler Filter block	Non-optimized Doppler Filter block	Total resources in the FPGA	Savings (%)
Slices	7382	7239	15360	−1.9%
DSP48 blocks	88	176	192	50%
RAMB16 blocks	44	88	192	50%

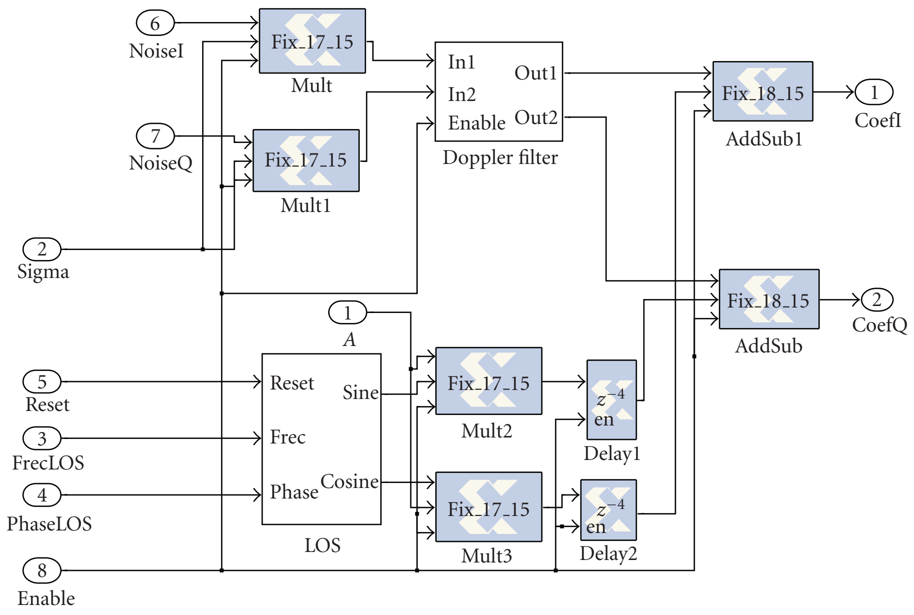

Figure 5: Generation and addition of the LOS and NLOS components of each path.

(i) At the time instant $t = 1, 2\ldots$ the current coefficient, s_t, and the one generated at the previous time instant, s_{t-1}, are copied p times, being p the interpolation factor. As a result, we have two vectors of upsampled coefficients

$$\mathbf{s}_t = \Bigl[\overbrace{s_t, s_t, \ldots, s_t}^{p}\Bigr] \tag{6}$$

and

$$\mathbf{s}_{t-1} = \Bigl[\overbrace{s_{t-1}, s_{t-1}, \ldots, s_{t-1}}^{p}\Bigr]. \tag{7}$$

We assume the initial condition $s_0 = 0$.

(ii) Next, the vectors $\mathbf{s}_t$ and $\mathbf{s}_{t-1}$ are subtracted and divided by p to produce the difference vector

$$\mathbf{\Delta}_t = \frac{\mathbf{s}_t - \mathbf{s}_{t-1}}{p}. \tag{8}$$

Obviously, $\mathbf{\Delta}_t = [\overbrace{\Delta_t, \Delta_t, \ldots, \Delta_t}^{p}]$ where

$$\Delta_t = \frac{s_t - s_{t-1}}{p}. \tag{9}$$

(iii) Finally, $\mathbf{\Delta}_t$ inputs an accumulator, that recursively computes the output as $y_t = y_{t-1} + \Delta_{\lfloor t/p \rfloor}$ for $t = 1, 2\ldots$ where $\lfloor \cdot \rfloor$ denotes the integer value (floor) operator. Notice that provided that $y_0 = 0$, $y_t = s_{t/p}$ for $t = p, 2p, 3p\ldots$ and the accumulator output corresponds to the output of a linear interpolator.

3.5.5. FIR Filtering. Finally, the signal from the IEEE 802.11p transmitter is filtered with the interpolated coefficients. Figure 7 depicts a general view of the developed complex FIR filter (this figure shows twenty-two paths instead of twenty-four because it is optimized for the *VTV-Expressway Oncoming* channel as in Figure 1). In the diagram, two groups of blocks can be distinguished: blocks aimed at delaying the incoming signal (`DelayBufferI` and `DelayBufferQ`) and blocks for applying the complex FIR filter to the delayed signals (*FIR_Taps1* to *FIR_Taps4*).

3.6. 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 Figure 1).

(2) The emulator starts to generate channel coefficients, both for the LOS and the NLOS components (illustrated in Figures 1 to 5).

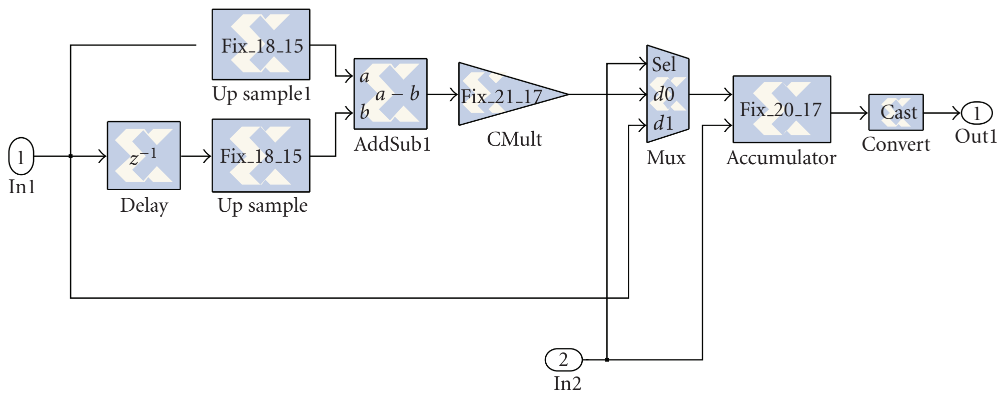

FIGURE 6: One path's linear interpolator.

(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 Figure 6). The interpolation is carried out in two stages, whose interpolation factors depend on the effective generation rates shown in Table 2. 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 (in Figure 7).

4. IEEE 802.11p Reference Transceiver Model

Figure 8 shows the basic setup of the IEEE 802.11p evaluation system, whose key parameters can be viewed in Table 5. In a nutshell, the system consists of a transmitter that generates packets that are sent to the receiver, passing through the vehicular channel emulator. The designs of the transmitter and the receiver, and the interface that connects both with the emulator, are detailed in the following subsections.

4.1. Transmitter. The blocks of the IEEE 802.11p transmitter are depicted at the top of Figure 8. The scheme is very similar to any OFDM transmitter with coding.

First, equally likely data bits are generated using a Bernoulli binary generator. In each cycle of the model, enough bits are generated to fill a frame with the number of desired OFDM symbols to transmit. Then, the generated bits are scrambled using a 127-bit sequence obtained from the generator polynomial $S(x) = x^7 + x^4 + 1$. A different pseudo random sequence is used for each transmission.

The scrambled bits are then processed by a rate 1/2 convolutional encoder with the industry-standard generator polynomials $g_0 = 133$ and $g_1 = 171$. In case of needing higher rates (2/3 and 3/4 are supported), puncturing is employed. Afterwards, data interleaving is carried out in a two-step permutation. The first permutation ensures that adjacent coded bits are mapped onto non-adjacent subcarriers, whilst the second permutation ensures that adjacent coded bits are mapped onto less and more significant bits of the constellation to avoid long runs of low reliability.

TABLE 5: Main features of the implemented IEEE 802.11p transceiver.

Parameter	Available Value(s)
Carrier Modulation	BPSK, QPSK, 16-QAM, 64-QAM
Code rate	1/2, 2/3, 3/4
No. subcarriers	48 data + 4 pilots
OFDM symbol duration	8 μs
Guard time	1.6 μs
FFT period	6.4 μs
Total bandwidth	10 MHz
Subcarrier spacing	0.15625 MHz

After interleaving, the bits are Gray-mapped into QAM symbols and placed into subcarriers. Figure 9 shows how the assignment of positions in each OFDM symbol is performed. Of the total 64 subcarriers, four are dedicated to pilot signals. These pilots are always situated in the same subcarriers (in positions −21, −7, 7, and 21, since subcarriers are numbered between −32 and 31). Besides, the pilots are BPSK modulated by a pseudo binary sequence. Forty-eight of the rest of the subcarriers are used for placing the data symbols. The subcarrier 0 is reserved for the DC and the remaining subcarriers are for frequency guards. Finally, the IFFT is performed.

For each OFDM symbol, a 1/4 cyclic prefix (CP) is added to prevent ISI and, eventually, a preamble is appended to the whole frame. Although the preamble is always transmitted in order to respect the requirements of the standard, it is not used in reception since we assume perfect time synchronization and we have not implemented any algorithms that may need it. Furthermore, the vehicular channel emulator operates in baseband, so there is no IF stage, neither in the transmitter nor in the receiver.

4.2. Receiver. The receiver blocks are shown at the bottom of Figure 8. The first step performed is the addition of white Gaussian noise in order to obtain BER (Bit Error Rate) and PER (Packet Error Rate) curves versus E_b/N_0 values.

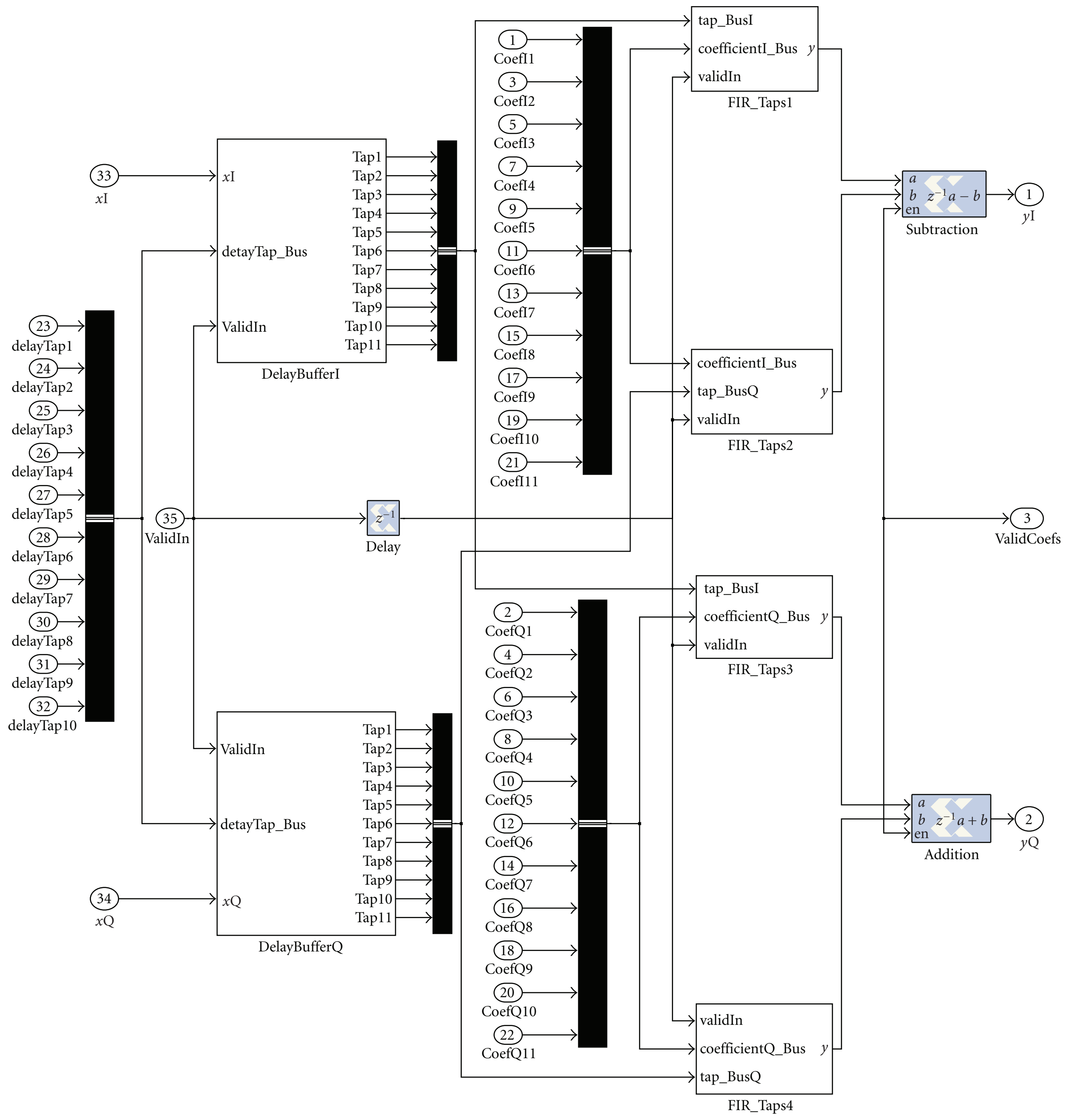

Figure 7: General view of the complex FIR filter.

After the preamble and the CP are removed, the FFT is applied to each OFDM symbol. Next, the channel has to be estimated. Although we have tested different techniques, we will only describe the simple method we used in the experiments in Section 5. We extract the four pilots and divide them by their respective transmitted values (which are known by the receiver), obtaining the estimated channel coefficients for the pilot subcarriers. These estimates are noisy and the implemented channel inversion method can even amplify the noise, but we decided to use it due to its simplicity.

Then, the four channel coefficient estimates are linearly interpolated to obtain the channel frequency response for the rest of the subcarriers. After this, an MMSE (Minimum Mean Square Error) equalizer [34] is employed.

Afterwards, the equalized symbols are sent to a soft detector. The output LLRs are deinterleaved, inverting the permutations performed in the transmitter, and the Viterbi block carries out decoding.

At the end, the decoded bits are descrambled, using the same scrambler as at the transmitter side and the final bits are obtained.

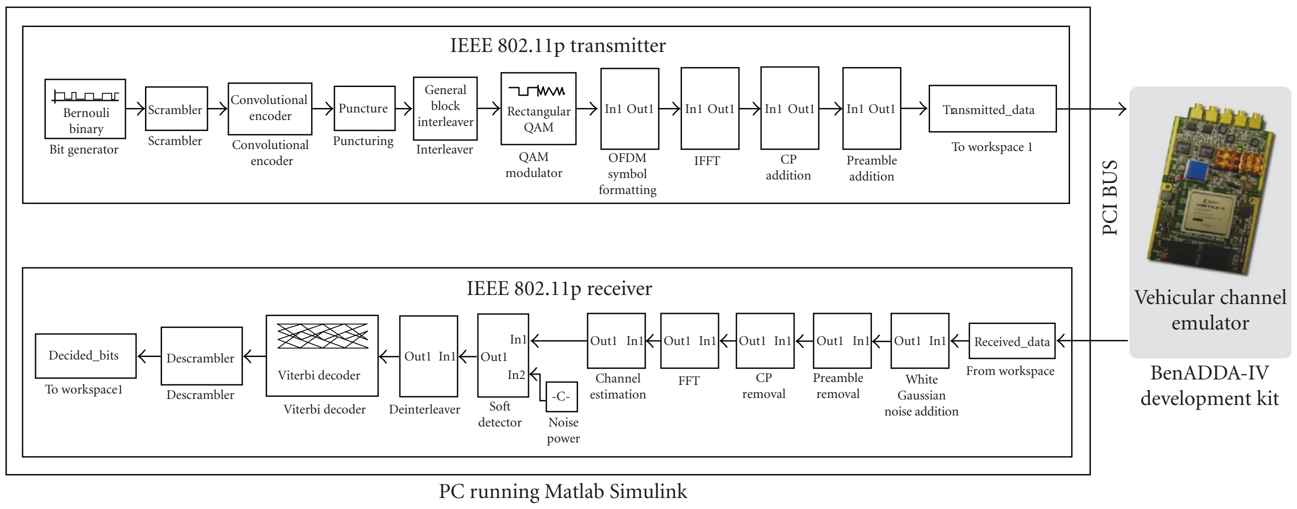

Figure 8: General view of the IEEE 802.11p evaluation system.

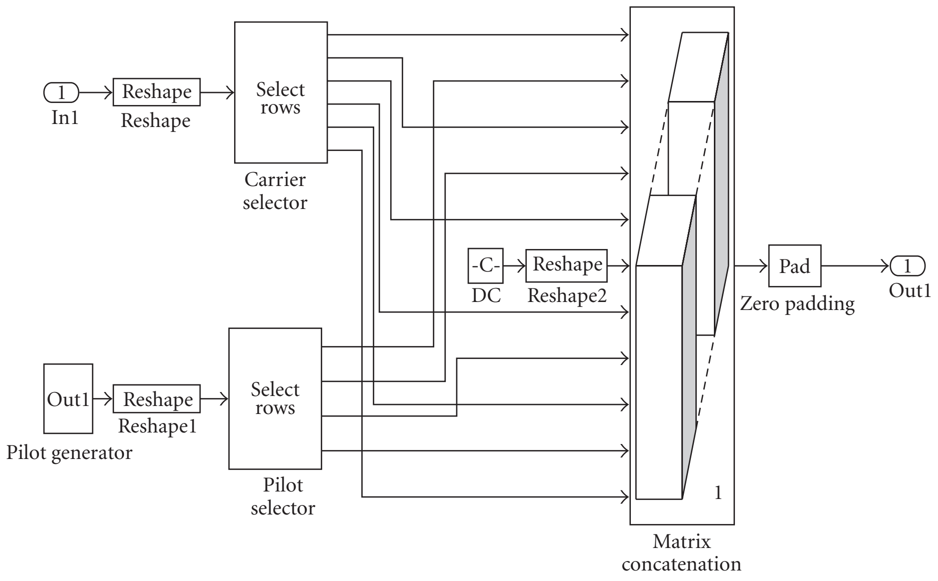

Figure 9: Placement of the DC, data and pilots into the OFDM symbol subcarriers.

4.3. Emulator Input/Output Interface. The exchange of data between the software transceiver and the FPGA is performed through the PCI bus, making use of two different kinds of memory blocks:

(i) *Registers*. The configuration parameters are set at the beginning of the emulation using registers. The value of these parameters do not change throughout the emulation, but, if desired, they could be modified dynamically.

(ii) *Shared FIFO (First-In First-Out) memories*. This sort of memories are used for driving the input/output IQ signals to/from the FPGA. The main advantage of using shared FIFOs is their ability to accelerate the simulation speed beyond what is typically possible with hardware co-simulation. Instead of transferring one sample per cycle, these blocks can work with frames of data that are sent in a single cycle to the FPGA. The largest available FIFO memory can hold up to 64,000 channel coefficients. Since each baseband OFDM symbol consists of 80 values (corresponding to 64 carriers plus the cyclic prefix), up to 800 OFDM symbols could be stored at the same time in one single FIFO.

Speed enhancements are achieved thanks to using the *Free-Running Clock* mode for hardware simulation, that allows the FPGA to operate asynchronously with respect to the Simulink simulation. This means that the FPGA is not kept in lockstep with the simulation. In our experiments, the vehicular emulator clock period was 100 ns (10 MHz), while Simulink's system period was one second, what implies that the FPGA was running much faster.

Note also that the size of the shared memories can be reduced to match the simulation requirements. For instance, when evaluating the performance in Section 5, we launched simulations with 64-byte packets, hence the input/output memories were adapted to such amount of data. In these simulations, when all the memory requirements were taken into account, only between 19% and 24% of the FIFO16/RAMB16 memory blocks were occupied (see Table 2).

Moreover, the fact of transmitting just one 64-byte packet at each time becomes an advantage when averaging BER/PER during performance measurements. In such case, the channel coefficients of two consecutive packets are less correlated than when sending the same two packets together in the same frame, because while the transmitter produces the second packet, the emulator keeps generating coefficients (with a period 10^6 times faster than the transmitter's period).

The data exchange process can be summarized as follows.

(1) The configuration parameters of the channel to emulate are set in registers that are read by the FPGA.

(2) The IEEE 802.11p transmitter generates the packets to be sent and creates a single data frame with all of them.

(3) The data frame to transmit is placed in a shared FIFO that is read by the vehicular emulator.

(4) The emulator applies the channel coefficients to the transmitted signal and sets the output values in shared FIFOs.

(5) The output shared FIFOs are read by the IEEE 802.11p software receiver that processes the transmitted packets.

5. Experiments

5.1. Evaluation of the IEEE 802.11p Physical Layer over Vehicular Channels. To evaluate the performance of our software implementation of the IEEE 802.11p PHY layer described in Section 4, we have passed the signal from the transmitter through the FPGA-based vehicular channel emulator. We have taken advantage of Xilinx Xtreme DSP software kit and we have performed measurements using the *co-simulation mode*: the transmitter and the receiver are implemented in MATLAB and Simulink, but the channel emulator runs on an FPGA.

Figures 10 and 11 depict, respectively, the BER and PER for the six vehicular channels presented in Section 3.1. We have also added the curve for the AWGN channel as a reference of the transceiver performance. In order to achieve a fair comparison, we set the same transmission parameters for the IEEE 802.11p transceiver, changing only the channel for each curve. A rate 1/2 FEC was used, each of the carriers was QPSK modulated, and a maximum of 10,000 64-byte data packets were averaged (the simulation stops for each E_b/N_0 value when 100 erroneous packets are detected). The receiver assumed perfect time synchronization, pilot-aided

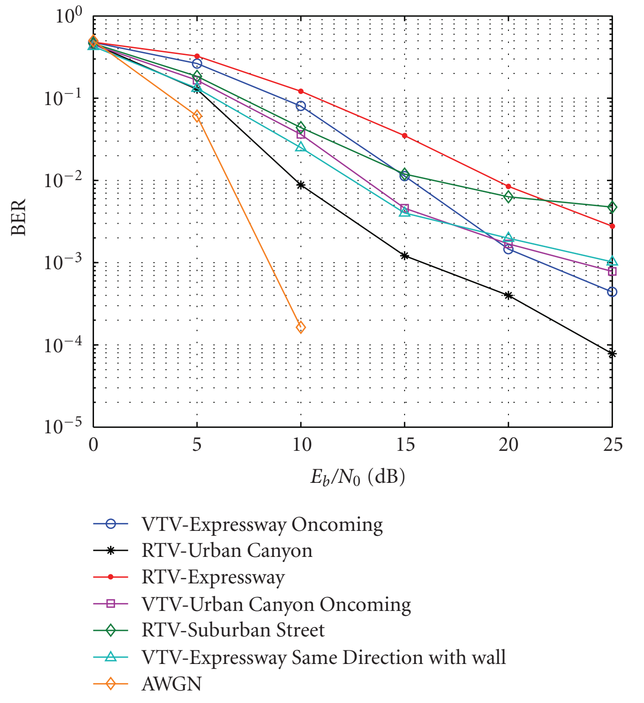

FIGURE 10: BER of the transceiver for the different vehicular channels.

channel estimation was performed and an MMSE linear equalizer was applied, as described in Section 4.

Notice that [1] states that a vehicular communications system must be capable of transferring messages to and from vehicles at speeds of 193 km/h with a PER of less than 10% for 64-byte data packets. Although the assumed speeds of the six implemented vehicular channel models are 140 km/h or 120 km/h (depending on the channel), the minimum E_b/N_0 to reach the 10% PER threshold constitutes a good reference of the system performance. Table 6 summarizes the required minimum E_b/N_0's to reach a PER of 10%. The values fluctuate between 10.1 and 18.2 dB, which are in an acceptable range when operating in a real scenario.

To justify our results, we compare them with the conclusions of [6]. There, the authors affirm that PER generally decreases with decreasing Doppler offsets and widths, and increasing K factors. Using Table 1 and Table 6, we are able to corroborate this assertion. On the one hand, if we sort the channels by their E_b/N_0 necessary to reach the 10% PER threshold and by their Rician K (using the *Overall K Factor* for *VTV-Expressway Same Direction with Wall*), we obtain the same ranking. This allows us to conclude that the larger the Rician K factor (clearer line-of-sight), the smaller E_b/N_0 is needed to reach the threshold. On the other hand, regarding the Doppler offset, it can be seen that the vehicular channel with the largest Doppler offset (*VTV-Expressway Oncoming*) is one of the channels that require more E_b/N_0 to reach the 10% threshold. However, in other channels there is not a clear impact in the E_b/N_0 threshold. This is the case of *VTV-Urban Canyon Oncoming* which owns large Doppler shifts, fading and LOS offsets. This latter case allows us to state that the degree of line-of-sight is a critical factor

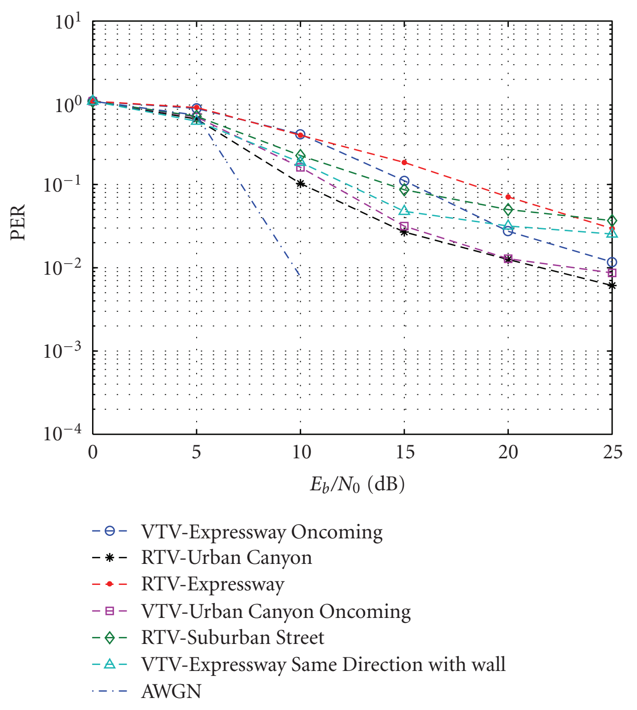

Figure 11: PER of the transceiver for the different vehicular channels.

Table 6: Minimum E_b/N_0 to obtain a PER of 10%.

Vehicular Channel	E_b/N_0 (dB)
VTV-Expressway Oncoming	15.36
RTV-Urban Canyon	10.10
RTV-Expressway	18.20
VTV-Urban Canyon Oncoming	11.45
RTV-Suburban Street	14.22
VTV-Expressway Same Direction with wall	12.25

since it shadows other influential factors like the mentioned frequency offsets.

Finally, we would like to stress that in a real transceiver there are other sources of error (i.e., RF impairments, synchronization errors, channel estimation mismatches...) which have not been added to the software transceiver but that could be easily implemented. In that sense, our IEEE 802.11p evaluation framework represents a very useful tool for transceiver designers, who can isolate the sources of error and analyze them while assessing the performance in a realistic scenario.

5.2. Observed Inter-Carrier Interference (ICI) in the Implemented Vehicular Channels. The use of an one-tap equalizer in the previous section was intentional. Due to the Doppler effect, it was very likely that ICI appeared in the received signal, but the practical results for this channel models and the assumed transmission conditions (i.e., no synchronization errors or RF impairments) have shown that there is almost no ICI at the receiver side. A simple explanation follows: let

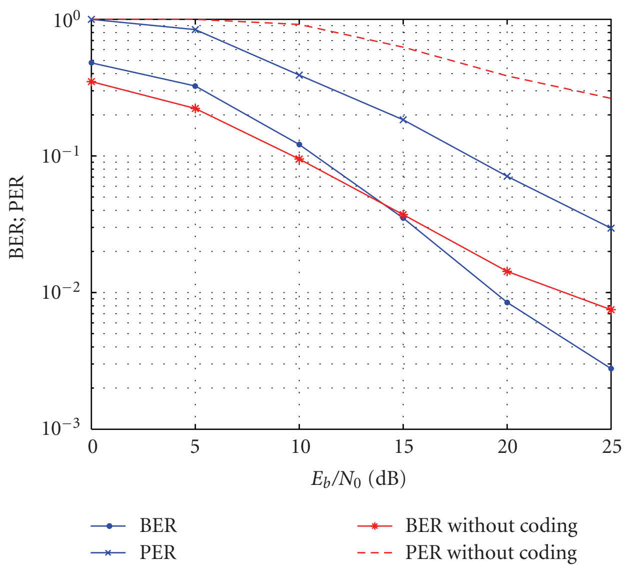

Figure 12: BER/PER when transmitting with and without coding through *RTV-Expressway*.

us first assume that the OFDM signal, after the FFT, can be modeled as

$$\mathbf{y}_f = \mathbf{F}y_t = \mathbf{F}\mathbf{H}_t\mathbf{F}^H\mathbf{s}_f + \mathbf{F}\mathbf{n}_t = \mathbf{H}_f\mathbf{s}_f + \mathbf{n}_f, \tag{10}$$

where $\mathbf{y}_t$ is the received signal vector, $\mathbf{F}$ and $\mathbf{F}^H$ are matrices that perform the FFT and the IFFT, respectively (^{H}denotes the Hermitian or conjugate transpose), $\mathbf{H}_t$ is the matrix with the channel coefficients in the time domain, $\mathbf{s}_f$ contains the transmitted symbols, and $\mathbf{n}_t$ and $\mathbf{n}_f$ represent vectors with Gaussian noise. In this model, the receiver recovers the transmitted symbols by estimating the matrix $\mathbf{H}_f = \mathbf{F}\mathbf{H}_t\mathbf{F}^H$.

The presence of ICI depends on the characteristics of $\mathbf{H}_f$. If this matrix is diagonal, the signal received in each subcarrier depends on the transmitted symbol and only on one channel coefficient and, therefore, there will be no ICI. However, if any non-diagonal coefficient of $\mathbf{H}_f$ is different from zero, ICI will appear.

For the implemented vehicular channel models, we observed that $\mathbf{H}_f$ was *almost* diagonal. As a specific measure of this observation, we considered the ratio between the energy of the coefficient in the diagonal and the sum of the energies of the rest of the values in the same row of the channel matrix. For the i-th row of one sample of the channel matrix, this ratio can be obtained as

$$q_i = \frac{|h_{ii}|^2}{\sum_{j=1\ j\neq i}^{N} \left|h_{ij}\right|^2}, \tag{11}$$

where h_{ij} are the coefficients of $\mathbf{H}_f$ and N is the length of the FFT ($N = 64$ for IEEE 802.11p). We averaged this ratio for 10,000 channel matrices. It is important to stress that the channel coefficients were obtained directly from the emulator, that is, they were not estimations.

Table 7 shows the average energy ratios for the six vehicular channels. This Table confirms that, on average, the

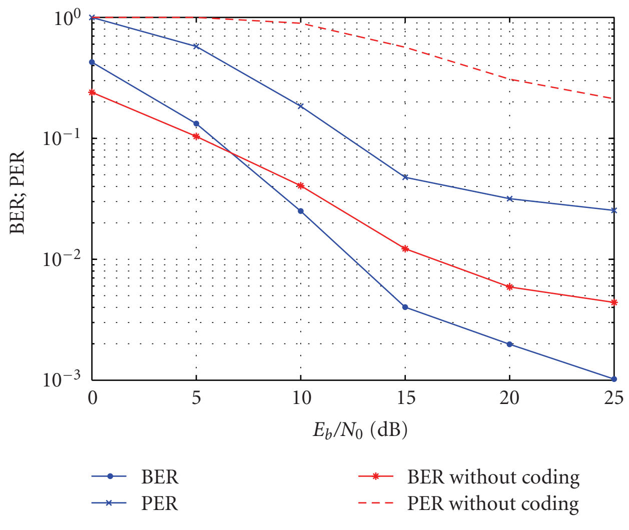

FIGURE 13: BER/PER when transmitting with and without coding through *VTV-Expressway Same Direction with Wall*.

TABLE 7: Energy ratios for the vehicular channels.

Vehicular Channel	Average Energy Ratio
VTV-Expressway Oncoming	2706.4
RTV-Urban Canyon	1151.4
RTV-Expressway	1746.3
VTV-Urban Canyon Oncoming	2769.6
RTV-Suburban Street	770.3
VTV-Expressway Same Direction with wall	940.1

energy of the diagonal element is at least 700 times higher than the sum of the energies of the rest of the coefficients in the same row. With such difference, the contribution to the ICI of the non-diagonal coefficients, though existent, can be ignored and, hence, a simple one-tap equalizer is enough to compensate the channel effects.

As said above, the considered emulation setup only accounted for channel estimation errors. In this case, the use of an one-tap equalizer was valid for overcoming the effects of the implemented vehicular channels. When testing actual transceivers, however, additional sources of error appear (e.g., synchronization errors, RF impairments...). In addition, the considered vehicular channel models do not cover all the situations that may appear in real operation. Thus, transceivers designed to operate in real environments should, of course, include more sophisticated equalizers to compensate for larger amounts of ICI.

5.3. Importance of Coding over Vehicular Channels. To show the importance of using coding when transmitting over vehicular channels, we chose two different channels and we measured the performance of our receiver when transmitting over them. We use the same transmission configuration described in Section 5.1.

Figures 12 and 13 show, respectively, the BER and PER for *RTV-Expressway* and *VTV-Expressway Same Direction with Wall* when transmitting with and without coding. In both channels a PER of 10% is achieved with an E_b/N_0 of more than 25 dB when coding is not used. Notice that performance is dramatically improved with the utilization of coding. The BER curves of the experiments that use coding remain, at low E_b/N_0 values, above the non-coded versions due to the generation of new errors when decoding. Once the decoder reaches a BER of a little less than a 10%, the decoder is able to correct errors and the versions with coding outperform the non-coded versions.

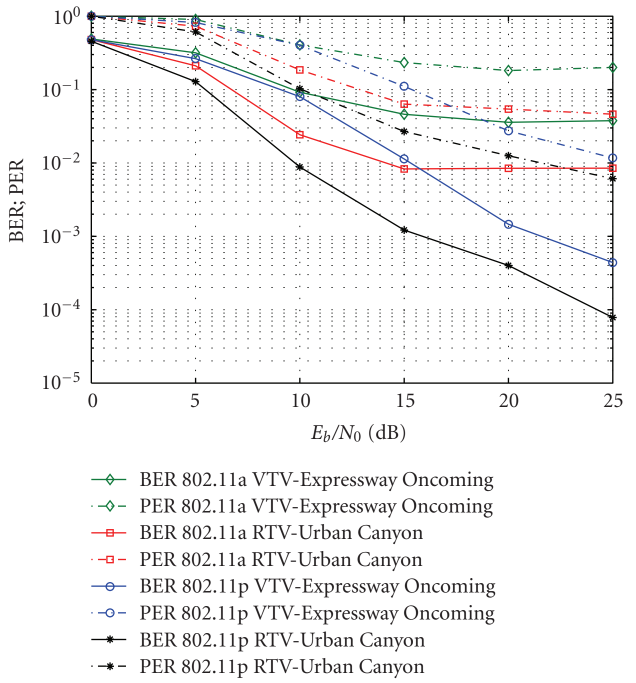

FIGURE 14: BER/PER comparison between IEEE 802.11p and IEEE 802.11a when transmitting over the channel emulator.

5.4. 802.11p versus 802.11a in Vehicular Channels. Our last experiment compares the performance of the PHY layers of IEEE 802.11a and IEEE 802.11p. As it was previously mentioned, the main difference between both standards at the physical level is the bandwidth: IEEE 802.11a uses 20 MHz, while IEEE 802.11p only 10 MHz. The reduction in bandwidth translates into larger OFDM symbols and halves the transmission rates.

Figure 14 gives a good example of the advantages of using IEEE 802.11p when transmitting through vehicular channels. For both chosen channels, the IEEE 802.11a transceiver performs worse and clearly experiences high error floors, starting at an E_b/N_0 of 15 dB. Furthermore, if we obtain the PER thresholds for the IEEE 802.11a transceiver, we see that in *RTV-Urban Canyon* it requires 12.87 dB (while IEEE 802.11p only needs 10.1 dB) and, in *VTV-Expressway Oncoming*, the IEEE 802.11a transceiver never reaches the threshold. This difference is due to the high delay spread of the vehicular channels that introduce interference among symbols in the case of IEEE 802.11a. Therefore, there exists an important improvement in using IEEE 802.11p instead of IEEE 802.11a when transmitting over vehicular environments.

6. Conclusion

We have presented a flexible, reconfigurable, and cost-effective solution for evaluating IEEE 802.11p transceivers through the real-time emulation of vehicular wireless channels. We presented a comprehensive review of the current state of the art of IEEE 802.11p transceiver designs and real-time vehicular channel emulators. We described the design and implementation of a real-time vehicular channel emulator using FPGA technology and rapid prototyping software tools. At the end, we have presented several examples of performance evaluation of the IEEE 802.11p PHY layer over six high-speed scenarios. We also studied the effects of using coding, which obtains important gains in BER/PER. Finally, we compared IEEE 802.11p and IEEE 802.11a when transmitting over vehicular channels, observing that the use of IEEE 802.11p dramatically improves the performance.

Acknowledgments

This work has been supported by Xunta de Galicia through contract Ref. PGIDIT06TIC10501PR and by Ministerio de Educación y Ciencia of Spain and FEDER funds of the E.U. under Grants TEC2007-68020-C04-01 (MultiMIMO project) and CSD2008-00010 (COMONSENS project).

References

[1] "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," ASTM Intl., E2213-03, September 2003.

[2] "IEEE 802.11a: Wireless LAN Medium Access Control and Physical Layer Specifications, High-Speed Physical Layer in the 5 GHz Band," IEEE, September 1999.

[3] R. Mangharam, J. Meyers, R. Rajkumar, et al., "A multi-hop mobile networking test-bed for telematics," in *Proceedings of Society for Automotive Engineers World Congress (SAE '05)*, Detroit, Mass, USA, April 2005.

[4] J. Dulmage, M. Tsai, M. Fitz, and B. Daneshrad, "COTS-based DSRC testbed for rapid algorithm development, implementation, and test," in *Proceedings of the 1st ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH '06)*, pp. 113–114, Los Angeles, Calif, USA, September 2006.

[5] T. M. Fernández-Caramés, J. A. García-Naya, M. González-López, and L. Castedo, "Flex vehd: a flexible testbed for vehicular radio interfaces," in *Proceedings of the 8th International Conference on Intelligent Transport System Telecommunications (ITST '08)*, pp. 283–287, Phuket, Thailand, October 2008.

[6] G. Acosta-Marum and M. A. Ingram, "Six time-and frequency-selective empirical channel models for vehicular wireless LANs," in *Proceedings of IEEE Vehicular Technology Conference (VTC '07)*, pp. 2134–2138, Baltimore, Md, USA, October 2007.

[7] G. Acosta-Marum, *Measurement, modelling and OFDM synchronization for the wideband mobile-to-mobile channel*, Doctoral thesis, May 2007.

[8] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich, "Design of 5.9 GHz DSRC-based vehicular safety communication," *IEEE Wireless Communications*, vol. 13, no. 5, pp. 36–43, 2006.

[9] D. Jiang and L. Delgrossi, "IEEE 802.11p: towards an international standard for wireless access in vehicular environments," in *Proceedings of IEEE Vehicular Technology Conference (VTC '08)*, pp. 2036–2040, Singapore, May 2008.

[10] Y. Wang, A. Ahmed, B. Krishnamachari, and K. Psounis, "IEEE 802.11p performance evaluation and protocol enhancement," in *Proceedings of IEEE International Conference on Vehicular Electronics and Safety (ICVES '08)*, pp. 317–322, Columbus, Ohio, USA, September 2008.

[11] S. Eichler, "Performance evaluation of the IEEE 802.11p WAVE communication standard," in *Proceedings of IEEE Vehicular Technology Conference (VTC '07)*, pp. 2199–2203, Baltimore, Md, USA, September 2007.

[12] M. Abbassi, *Characterization of a 5 GHz modular radio frontend for WLAN based on IEEE 802.11p*, M.S. thesis, Telecommunications, University of Gävle, Vienna, Austria, 2008.

[13] J. Reddy, L. Deneire, L. Van der Perre, B. Gyselinckx, and M. Engels, "On equalization for OFDM-dedicated short range communication (DSRC) modem," in *Proceedings of IEEE International Conference on Personal Wireless Communications (ICPWC '00)*, pp. 230–234, Hyderabad, India, December 2000.

[14] S. Sibecas, C. A. Corral, S. Emami, G. Stratis, and G. Rasor, "Pseudo-pilot OFDM scheme for 802.11a and R/A in DSRC applications," in *Proceedings of IEEE Vehicular Technology Conference (VTC '03)*, no. 2, pp. 1234–1237, Orlando, Fla, USA, October 2003.

[15] Y.-H. Cheng, Y.-H. Lu, and C.-L. Liu, "Adaptive channel equalizer for wireless access in vehicular environments," in *Proceedings of the 6th International Conference on ITS Telecommunications (ITST '06)*, pp. 1102–1105, Chengdu, China, June 2006.

[16] H. Abdulhamid, K. E. Tepe, and E. Abdel-Raheem, "Iterative channel-tracking techniques for 5.9 GHz DSRC applications," *Research Letters in Communications*, vol. 2008, Article ID 485013, 5 pages, 2008.

[17] J. Mar and C. Kuo, "Performance improvement of the DSRC system using a novel S and PI-decision demapper," in *Proceedings of the International Conference on Communications (ICC '08)*, Beijing, China, May 2008.

[18] N. Sasho, K. Minami, H. Fujita, et al., "Single-chip 5.8 GHz DSRC transceiver with dual mode of ASK and $\pi/4$-QPSK," in *Proceedings of the Radio and Wireless Symposium*, Orlando, Fla, USA, January 2008.

[19] S. Shin, S. Yun, S. Cho, et al., "0.18 μm CMOS integrated chipset for 5.8 GHz DSRC systems with +10 dBm output power," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '08)*, Seattle, Mass, USA, May 2008.

[20] N. Almeida, R. Abreu, J. N. Matos, N. B. Carvalho, and J. S. Gomes, "Low cost transceiver for DSRC applications," in *Proceedings of the Asia-Pacific Microwave Conference (APMC ' 06)*, vol. 3, pp. 1501–1504, Yokohama, Japan, December 2006.

[21] T. Tsuboi, J. Yamada, N. Yamauchi, and M. Hayashi, "Dual receiver communication system for DSRC," in *Proceedings of the 2nd International Conference on Future Generation Communication and Networking (FGCN '08)*, vol. 1, pp. 459–464, Sanya, China, December 2008.

[22] J. Mar, Y. R. Lin, T. H. Lung, and T. H. Wei, "Realization of OFDM modulator and demodulator for DSRC vehicular communication system using FPGA chip," in *Proceedings of the International Symposium on Intelligent Signal Processing*

and Communications (ISPACS '06), pp. 477–480, Tortori, Japan, December 2006.

[23] J. Mar, C.-C. Kuo, Y.-R. Lin, and T.-H. Lung, "Design of software-defined radio channel simulator for wireless communications: case study with DSRC and UWB channels," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 8, pp. 2755–2766, 2009.

[24] Spirent Communications, December 2009, http://www.spirent.com/.

[25] Rhode & Schwarz, December 2009, http://www2.rohde-schwarz.com/.

[26] Azimuth Systems, December 2009, http://www.azimuthsystems.com/.

[27] Agilent Technologies, December 2009, http://www.home.agilent.com/.

[28] E. Boutillon, J.-C. Danger, and A. Ghazel, "Design of high speed AWGN communication channel emulator," *Analog Integrated Circuits and Signal Processing*, vol. 34, no. 2, pp. 133–142, 2003.

[29] A. Alimohammad, S. F. Fard, B. F. Cockburn, and C. Schlegel, "An accurate and compact Rayleigh and Rician fading channel simulator," in *Proceedings of IEEE Vehicular Technology Conference (VTC '08)*, pp. 409–413, Singapore, May 2008.

[30] J.-K. Hwang, K.-H. Lin, J.-D. Li, and J.-H. Deng, "Fast FPGA prototyping of a multipath fading channel emulator via high-level design," in *Proceedings of the International Symposium on Communications and Information Technologies (ISCIT '07)*, pp. 168–171, Sydney, Australia, October 2007.

[31] C. Melfhürer, F. Kaltenberger, M. Rupp, and G. Humer, "A scalable rapid prototyping system for real-time MIMO OFDM transmissions," in *Proceedings of the IEE/EURASIP Conference on DSP Enabled Radio*, Southampton, UK, September 2005.

[32] F. Kaltenberger, G. Steinboeck, R. Kloibhofer, R. Lieger, and G. Humer, "A multi-band development platform for rapid prototyping of MIMO systems," in *Proceedings of the ITG/IEEE Workshop on Smart Antennas*, Duisburg, Germany, April 2005.

[33] J.-K. Hwang, J.-D. Li, R.-L. Chung, and C.-Y. Chen, "Efficient structure for FPGA implementation of a configurable multipath fading channel emulator," in *Proceedings of the International Symposium on Intelligent Signal Processing and Communications (ISPACS '06)*, pp. 481–484, Tottori, Japan, December 2006.

[34] L. Rugini, P. Banelli, and G. Leus, "Simple equalization of time-varying channels for OFDM," *IEEE Communications Letters*, vol. 9, no. 7, pp. 619–621, 2005.