PERFORMANCE OF A PARALLEL HAMMING CODING IN SHORT-FRAME OFDM SENSOR'S NETWORK

Raouia Masmoudi Ghodhbane[1] and Jorge Fernandez-Mayoralas[1,2]
[1]Safran Tech, Safran Sensing Systems $(\mathcal{S}^3)$, Rue des Jeunes Bois, Châteaufort , 78114 Magny-les-Hameaux, France,
[2]CentraleSupelec, Integrated Circuit and Electronic Systems, rue Joliot-Curie, F-91192 Gif-sur-Yvette Cedex, France.

NOTE: Corresponding author: Raouia Masmoudi Ghodhbane, raouia.masmoudi@gmail.com

Abstract – *In this paper, we focus on the most relevant Error Correcting Codes (ECCs): the Hamming code and the Reed-Solomon code in order to meet the trade-off between the low implementation complexity and the high error correction capacity in a short-frame OFDM communication system. Moreover, we discuss and validate via simulations this trade-off between complexity (Hamming is the easiest to code) and error correction capability (Reed-Solomon being the most effective). Therefore, we have to either improve the correction capacity of the Hamming code, or decrease the complexity cost for the Reed-Solomon code. Based on this analysis, we propose a new design of parallel Hamming coding. On the one hand, we validate this new model of parallel Hamming coding with numerical results using MATLAB-Simulink tools and BERTool Application which makes easier the Bit Error Rate (BER) performance simulations. On the other hand, we implement the design of this new model on an FPGA mock-up and we show that this solution of a parallel Hamming encoder/decoder uses a few resources (LUTs) and has a higher capability of correcting when compared to the simple Hamming code.*

Keywords – error correcting codes, FPGA, Hamming code, parallel Hamming coding, OFDM communication systems, Reed-Solomon code, short-frame, sensor's network, VHDL simulation

1. INTRODUCTION

Nowadays, Orthogonal Frequency Division Multiplexing (OFDM) systems are increasingly used in several applications such as: digital radio, television broadcasting systems, mobile communication systems and Power-Line Communications (PLC). OFDM is a convenient modulation scheme which combines the advantages of high data rates and easy implementation.

However, the main challenge associated with OFDM communication systems today remains related to Error Correcting Codes (ECCs) implementation [2]. Encoding and decoding are very important blocks in OFDM systems in which redundant information is added to the signal to allow the receiver to detect and correct errors that may have occurred in transmission. There are many techniques for error detection and correction such as: the Hamming code [3], Turbo code [4], Bose, Chaudhuri, and Hocquenghem (BCH) code [5], Reed-Solomon code [6], Convolutional code/ Viterbi [7], and LDPC [8].

One way to compare the efficiency of several ECCs is to compare their performance in terms of their complexity of implementation and their error detection or/and correction capability. Hamming and Reed-Solomon codes have proved to be a good compromise between efficiency and complexity. Hamming is very easy to implement and does not consume many resources, and it is a robust ECC, but the Bit Error Rate (BER) performance is not the most effective. The Reed-Solomon is more optimal to eliminate errors (especially for burst errors), but it is also more complex to implement.

Our domain of application is PLC based on OFDM systems for aeronautic sensor's network [9, 10]. According to [10], a PLC channel has major limitations especially in terms of bandwidth, impedance mismatches and noise.

Since power consumption has been a critical factor in the design of sensor's network, we consider sensor's network with short periods of activity, in which some sensors' devices go into standby mode. This consideration and the need for low latency access [11] to sensor data underscores the need for a short-frame OFDM and fast communication protocol.

In this paper, we focus on a low-powered short-frame OFDM communication system where the sensors' devices send only one OFDM symbol per frame on the PLC channel. We simulate and model the design of this low-powered short-frame OFDM communication system in terms of error correction. As mentioned previously, the complexity of the chosen ECC is one of the studied criteria: the ECC has to be effective and fairly easy to implement at the same time.

Usually, to have better performance in terms of error correction and/or error detection, we have to add several error correcting codes (ECCs) in serial either at the encoder or on the decoder level. Our case of applications requires a very simple ECC in order to implement it on a High Temperature (HT) ASIC whose number of cabled flip-flops and multipliers are limited by the HT technology.

Therefore, the novelty of this paper lies in the performance analysis of a new design based on the parallel Hamming coding which meets the trade-off between the low implementation complexity and the high error correction capacity. In the literature, the closest work to ours

is [12] in which the authors used parallel filters with only one Hamming code as an error correcting code. Their scheme allows more efficient protection when the number of parallel filters is large.
Our contributions can be summarized as follows:

- The design of this low-powered short-frame OFDM communication system in terms of error correction is modelled by MATLAB-Simulink tools.
- We analyse and choose the adapted Error Correcting Codes (ECCs); such as Hamming and Reed-Solomon for this low-powered short-frame OFDM communication model.
- We discuss the trade-off between the low implementation complexity and the high error correction capacity.
- Moreover, we propose a new model of parallel Hamming coding in order to increase the error correction capability of our model and we illustrate its performance in terms of Bit Error Rate (BER) vs. EB/N0.
- Finally, we validate the performance of parallel Hamming encoder/decoder in terms of complexity of implementation on an FPGA board using VHDL simulations.

The remainder of this paper is organized as follows: First, a brief description of the communication model in Section 2. Then, we study the trade-off between the two ECCs (Hamming code and Reed-Solomon code) which are more adapted to our communication model. We propose a new design of a parallel Hamming coding in the case of a short-frame OFDM sensor network in Section 3. Moreover, we illustrate the performance of these ECCs in terms of Bit Error Rate (BER) for different scenarios using BERTool application in Section 4. In Section 5, we validate the performance of parallel Hamming encoder/decoder in terms of complexity of implementation on an FPGA board using VHDL simulations. Finally, we conclude this paper in Section 6.

2. DESCRIPTION OF THE COMMUNICATION MODEL

Here, we consider a sensor's network which is composed of one master device and several slave-sensor node devices $(S_1, S_2, ..., S_{20})$ using Time Division Multiple Access (TDMA) to share the PLC channel illustrated in Fig. 1.

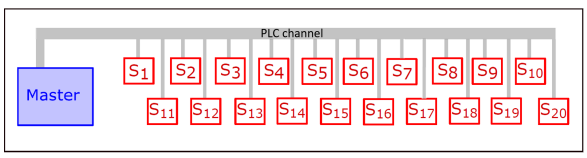


Fig. 1 – Shared bus architecture for our sensor's network

Since low-power consumption is a critical factor in the design of sensor's network, we consider a low-powered short-frame OFDM communication system where sensors' devices send only one OFDM symbol per frame on the PLC channel. Each time slot is composed of an OFDM symbol which represents a communication between the master device and each slave-sensor node device.
In Table 1, we denote our short-frame OFDM parameters that are considered later in our MATLAB-Simulink simulations:

Table 1 – Scenarios and short-frame OFDM parameters used in our MATLAB-Simulink simulations

Parameters	**Value**
N_{FFT}: FFT size	64
N_{SC}: Used sub-carriers	30/32
f_s: Sampling frequency	1.6 MHz
Δ_f: Sub-carrier spacing	25 KHz
T_s: Total Symbol duration	$45\mu s$
M: Modulation size	4 (QPSK)
L: Packets of L bits	50 bits
ECC: Error correcting codes	Hamming, Reed-Solomon
CR: Coding rates	$CR_1 = (4/7), CR_2 = (11/15),$ $CR_3 = (23/31), CR_4 = (26/31),$ $CR_5 = (57/63), CR_6 = (56/64)$
$BERTool$: BER Performance Analyser GUI	$N_{errors} = 10^6$; $N_{bits} = 10^{10}$; $\frac{E_b}{N_0} = 0 : 15$ dB

In this paper, we use MATLAB-Simulink tools in order to model this low-powered short-frame OFDM communication system as shown in Fig. 2.

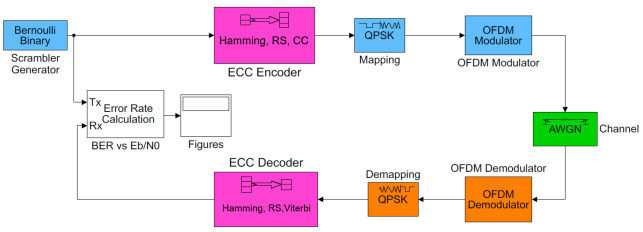


Fig. 2 – Short-frame OFDM communication model using MATLAB-Simulink tools: General model

Next, we will describe each block represented in Fig. 2 representing our general low-powered short-frame OFDM communication model:

2.1 Random input generation

In this block, the input data is randomized by a "Bernouilli Binary" block in order to spread the energy over all the bits before being encoded by the following block.

2.2 Error correcting code (ECC)

The purpose of this block is to add enough redundancy to the data packets being sent, so that even if some of the received data includes errors, there will be enough infor-

mation provided to the receiver to reconstruct the data from its origin. In our communication model, we suppose that the ECC will be in charge of correcting all the errors created by the OFDM communication model.

- ECC encoder: This block is the transmitter side of the ECC. The communication channel alters the signal, which introduces errors. We can use error correcting codes to add redundancy in the transmitted data. Many encoding methods have been studied in the literature, such as the Hamming and the Reed-Solomon codes, which are detailed in Section 3.
- ECC decoder: This block is on the reception side of the model, receiving the information that has passed through the channel and that has already been demodulated. The decoder has to decode the message and translate it back into its' original form by exploiting the redundant bits that were added previously by the encoder and that allows the decoder to detect an error and correct it. In general, the decoder is more complex than the encoder, especially concerning its implementation (for example Viterbi decoder [7]).

2.3 Mapping/De-Mapping

This block takes the coded messages and builds a constellation in an I/Q plane, in order to transmit our digital stream of bits through the analog channel, under the form of frequencies.

- Mapping: This block takes a binary stream and outputs a point in the I/Q plane. There are many types of modulation: such as shifting the phase of the signal 'Phase-Shift Keying (PSK)', modulate only the amplitude of the signal 'Amplitude Modulation (AM)', or both 'Quadrature amplitude modulation (QAM)'. In our communication model, we use a Quadrature phase-shift keying (QPSK) modulation. There are $M = 4$ possible points in the constellation and it can encode $n_b = \log_2(M) = 2$ bits per symbol.
- De-Mapping: This block does the inverse of the previous mapping block. It takes the signal in the time domain after the Fast Fourier transform (FFT) block (which is inside the OFDM modulator block), and with the constellation reference point it restores the original binary message corresponding to a given point.

2.4 Orthogonal Frequency Division Multiplexing (OFDM) MoDem

OFDM is a type of digital transmission and a method of encoding digital data on multiple carrier frequencies. In OFDM, multiple closely spaced orthogonal sub-carrier signals with overlapping spectra are transmitted to carry data in parallel.

- OFDM Modulator: This block introduces the pilot signals (which are used later in the channel estimation block to correct the effects of the channel), the guard bands, the cyclic prefix and a raised cosine filter (which serves as windowing to focus our study on the used frequencies). Then, it includes also an Inverse Fast Fourier transform (IFFT) module which allows the information to be transmitted in orthogonal frequencies through the communication channel.
- OFDM Demodulator: This block does the reverse operation of the previous one. An FFT module transforms the information back into the time domain, the pilot signals are used for the channel estimation, then they are removed to restore the original message (the guard bands and cyclic prefix are removed as well).

2.5 The communication channel

Here, the communication channel is modelized by an Additive White Gaussian Noise (AWGN) block. We induce noise by using the $\frac{Eb}{No}$ which represents a normalized Signal to Noise Ratio (SNR) by bit, and is well adapted to compare the Bit Error Rate (BER) of different ECCs. The relation between these two parameters is given by:

$$SNR = \frac{Eb}{No} * \frac{Bit\ Rate}{Channel\ Bandwidth} \qquad (1)$$

The deviation (which is equal to $10mV$ in our case) represents the thermal noise in our real Power Line Communication (PLC) channel [9, 10]. We have estimated this deviation on LT Spice tools by adding the quantification noise and converting the sine wave by the ADC/DAC block. In order to make the channel block more realistic, we fixed these parameters of the AWGN block the closest to reality.

In the following Section, we will focus only on the ECC blocks which are described in Section 2.2 and adapted to our short-frame OFDM communication system.

3. TRADE-OFF BETWEEN SIMPLE HAMMING CODE AND REED-SOLOMON CODE

In this Section, we will focus on two error correcting codes (ECCs) that meet a trade-off between the low implementation complexity and the high error correction capacity. Since we have considered a short-frame OFDM communication model, we will study Hamming code and Reed-Solomon code which are the most adapted in our case.

3.1 The Hamming code

The Hamming code is one of the error correcting codes that can be used to detect and correct bit errors that can occur when data is moved or stored. Like other error correcting codes, the Hamming code makes use of the concept of parity bits which are bits that are added to data so that the validity of the data can be checked when it is read or after it has been received in a data transmission.
The Hamming code method is based on two methods (even parity, odd parity) for generating redundancy bits.

The number of redundancy bits are generated using the following formula:

$$2^r = D + r + 1, \qquad (2)$$

where, r represents the number of redundancy bits and D the number of information data bits.
For example, if we calculate the number of redundancy bits for a $D = 11$ bits then it comes to add $r = 4$ redundancy bits. These parity/redundancy bits (P_1, P_2, P_4, P_8) are added to the information bits $(D_1, ..., D_{11})$ at the transmitter (Hamming encoder) and then removed at the receiver (Hamming decoder) which is able to detect and correct errors.

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Encoded data bits	P_1	P_2	D_1	P_4	D_2	D_3	D_4	P_8	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}
P_1	x		x		x		x		x		x		x		x
P_2		x	x			x	x			x	x			x	x
P_4				x	x	x	x				x	x	x	x	x
P_8								x	x	x	x	x	x	x	x

Table 2 – The encoded bits for Hamming code $[15, 11]$

The Hamming encoder calculates these parity bits according to Table 2, and outputs a 15 bits message. The Hamming Decoder calculates the parity bits:

- If each parity bit is equal to zero, i.e., $(P_1, P_2, P_4, P_8) = 0000$, there are no errors in this OFDM communication model.
- If not, the position of the error is displayed in the four parity bits (P_1, P_2, P_4, P_8). The decoder flips then the concerned bit and returns the 11 bits message $(D_1, ..., D_{11})$.

Next, we will simulate the Hamming code $[n, k]$ for several coding rates: $CR_1 = (4/7), CR_2 = (11/15), CR_4 = (26/31)$, where each coding rate $CR = k/n$ is the ratio between the code dimension k and the code length n, in order to study its influence on the system's performance.

The theoretical point of view "the longer the code, the better the error performance" is proved in Fig. 3 and Fig. 4. Fig. 3 shows that the coding rate is crucial to obtain a good performance. When $\frac{E_b}{N_0} \leq 9$ dB, the Hamming code curve with the coding rate CR_4 is very close to the Hamming code curve with the coding rate CR_2 in terms of BER.

3.2 The Reed-Solomon code

The Reed-Solomon code operates on a block of data treated as a set of finite-field elements called symbols. Reed–Solomon code is able to detect and correct multiple symbol errors especially burst errors. Since the Reed-Solomon code is a non-binary code, the code has symbols from $\mathbb{F}_q$ with parameters $(q-1, k)$, which is used to make a mapping of primitive polynomial with binary coefficients,

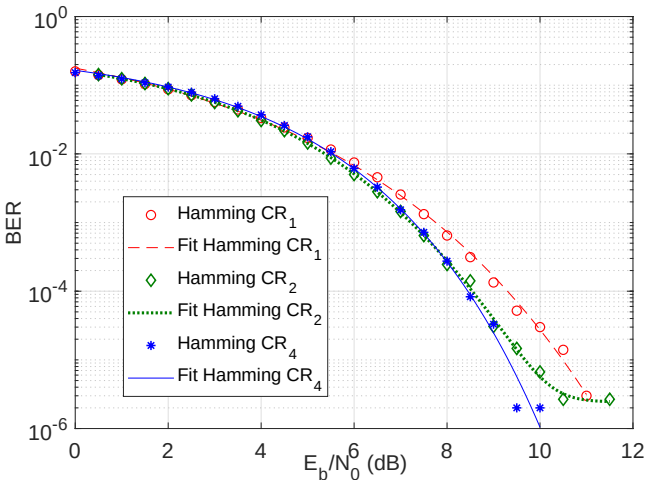


Fig. 3 – BER performance of the Hamming code for different coding rates CR_1, CR_2, CR_4

frequently we have $q = 2^m$ to use them as binary codes, each element being represented as a binary m-tuple.
In terms of complexity, the Reed–Solomon encoder is fairly simple in terms of blocks and only involves multipliers and adders in the Galois Field. We can either create a multiplication module or use RAM slots and create multiplication tables, However, the Reed–Solomon decoder includes several algorithms that consume a lot in resources, especially the Berlekamp algorithm.
Next, we will simulate the Reed-Solomon code (n, k) for several coding rates: $CR_1 = (4/7)$, $CR_2 = (11/15)$, $CR_3 = (23/31)$, where each coding rate $CR = k/n$ represents the ratio between k the code dimension and the code length $n = q - 1$, in order to study its influence on the system's performance.

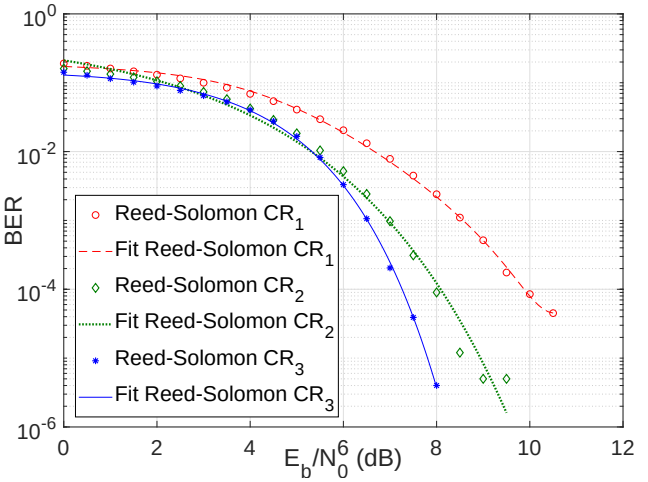


Fig. 4 – BER performance of the Reed-Solomon code for different coding rates CR_1, CR_2, CR_3

Once again, it is a trade-off between performance and complexity with different coding rates: as the most effective coding is the highest, but also the most complex. Fig. 4 shows that the three curves converge faster to zero when compared to the hamming code curves in Fig. 3. In fact, the Reed-Solomon curve with the highest coding rate

CR_3 manages to converge to no errors for $\frac{E_b}{N_0} = 8$dB, which represents the noise level that we will find in the real PLC channel.

3.3 Discussion and comparison

As we discussed before, the complexity and the capacity of error correction of the chosen ECC are the most important criteria: the ECC has to be effective and fairly easy to implement at the same time.

The capacity of error correction of each ECC is known and depends on its structure and their way of coding the information. Table 3 summarizes the detection and correction capability of each ECC scheme. We denoted here by SED/DED/MED: the Single/Double/Multiple Error Detection, and by SEC/DEC/MEC: the Single/Double/Multiple Error Correction.

Table 3 – The capability of error detection or/and error correction for each ECC.

ECC	SED	SEC	DED	DEC	MED	MEC
Hamming code	x	x				
Extended Hamming	x	x	x			
Reed-Solomon Code	x	x	x	x	x	x
BCH Code	x	x	x	x	x	x

While the Hamming code can be implemented creating just two fairly simple modules (encoder/decoder) based on XOR gates, the Reed-Solomon code requires a higher number of blocks. In fact, the Reed Solomon encoder needs: Adder in Galois Field, Multiplier in Galois Field, Multiplex and Registers [13]. The Reed-Solomon decoder needs algorithms to decode the code word such as: Syndrome calculator; Berlekamp-Massey algorithm (or Euclid algorithm)which finds the location of the errors by creating an error locator polynomial; Chien Search Algorithm which finds the roots of the previous polynomial; Forney's algorithm where the symbol's error values are found and corrected. Thus, these blocks are complex and use multiplications (or RAM memory if we use tables).

Here, we have simulated in Fig. 5 the Hamming code and the Reed-Solomon code for the same coding rate CR_2 in order to compare their BER performance.

The Hamming and Reed-Solomon codes have proved to be a good compromise between efficiency and complexity. Hamming is very easy to implement and does not consume many resources, and it is a robust ECC, but the Bit Error Rate (BER) performance (in Fig. 5) shows that it is not the most effective ECC. Reed-Solomon is more optimal to eliminate errors, but it is also more complex than the Hamming code.

In the following Section, we will propose a new model of parallel Hamming coding in order to increase the error correction capability of our model.

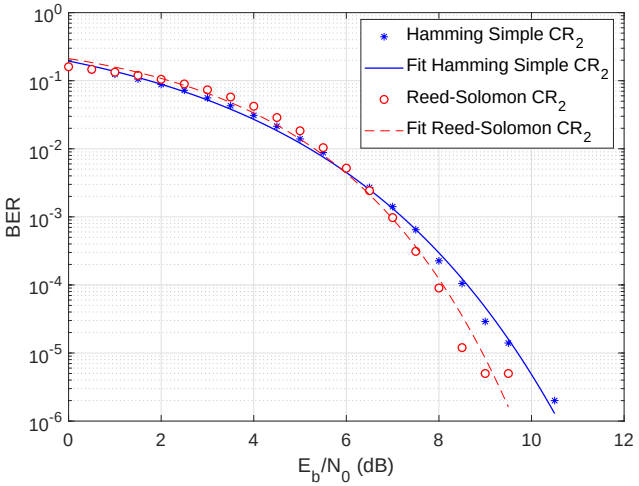


Fig. 5 – Comparing the simple Hamming code to the Reed-Solomon code

4. PARALLEL HAMMING CODING RESULTS USING MATLAB-SIMULINK AND BERTOOL

In this Section, we have selected only a few of the most illustrative and interesting scenarios to be presented here. In order to plot the Bit Error Rate (BER) function of the $\frac{E_b}{N_0}$ which is given in equation *(1)*, we have used the Matlab tools called "BERTOOL".

4.1 BER performance analyser for MATLAB-Simulink models

Here, we use the Bit Error Rate (BER) analysis GUI (called BERTool) from Matlab application. The BERTool application is able to analyze the BER performance of different communications systems as a function of signal-to-noise ratio $\frac{E_b}{N_0}$ given in equation *(1)*. It analyzes performance either with Monte-Carlo simulations of MATLAB functions and MATLAB-Simulink models or with theoretical closed-form expressions for selected types of communication systems[14].

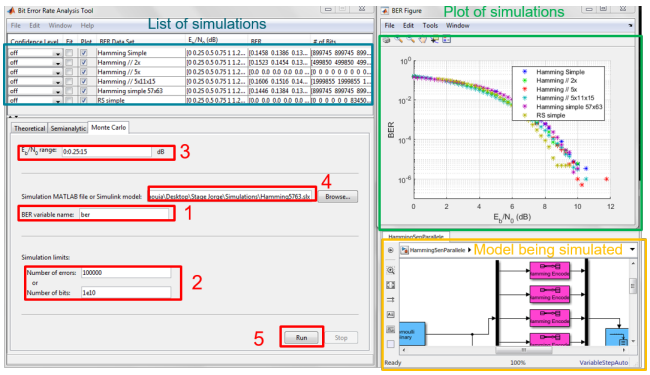


Fig. 6 – Bertool Interface: steps to plot the BER vs EB/No simulations

On the Monte Carlo window in Fig. 6, we have specified the BERTool parameters (which are detailed in Table 1) based on our scenarios. In order to generate BER data for each communication system using the Simulink models, we follow five steps based on the following BERTool process [15]:

1. We calculate Bit Error Rate (BER) as a function of the energy per bit to noise power spectral density ratio ($\frac{E_b}{N_0}$).

2. We fix the number of errors ($N_{errors} = 10^6$) and the number of bits ($N_{bits} = 10^{10}$) in order to make accurate error rate. We have chosen this number of bits value to prevent the simulation from running too long, especially at large values of $\frac{E_b}{N_0}$.

3. We have specified the $\frac{E_b}{N_0}$ range based on our PLC channel model: $\frac{E_b}{N_0} = 0 : 15$ dB.

4. We generate the BER data for a chosen Simulink model. This Simulink Model is displayed and run in real time on the models being simulated window (as shown in Fig. 6) for each value of the energy per bit to noise ratio $\frac{E_b}{N_0}$. BERTool iterates over our choice of the energy per bit to noise ratio $\frac{E_b}{N_0}$ value and collects the results on a list of simulations.

5. Finally, we run the simulation in order to plot the estimated BER values function of the previous steps. The plot of simulation window is displayed and shows each curve for each Simulink model. We save later in the list of simulations each curve in order to compare graphically the different models.

In the following paragraphs, we will describe the most interesting models that we have simulated with the BERTool application interface as shown in Fig. 6.
As we have discussed in the previous Section, Hamming seems to be the most appropriate ECC for our scenario, but the small correction capacity could be an obstacle if there are several errors in the 50 bit package. For instance, we consider the Hamming code $[n, k]$ with coding rate $CR = k/n$ where n represents the code length and k the code dimension.

- Model 1: We cut the message on 1 time when we are coding with Hamming coding rate $CR_5 = 57/63$; Since in Model 1, the Hamming code has length 63 and dimension 57, we have to append 7 zeros to 50 information bits and then do encoding.

- Model 2: We cut the message on 2 times when we are coding with Hamming coding rate $CR_4 = 26/31$; Since in Model 2, the Hamming code has length 62 and dimension 52, we have to append only 2 zeros to 50 information bits before encoding.

- Model 3: We cut the message on 5 times when we are coding with Hamming coding rate $CR_2 = 11/15$. Since in Model 3, the Hamming code has length 75 and dimension 55, we have to append 5 zeros to 50 information bits before encoding.

Model 1: Simple Hamming code $[63, 57]$

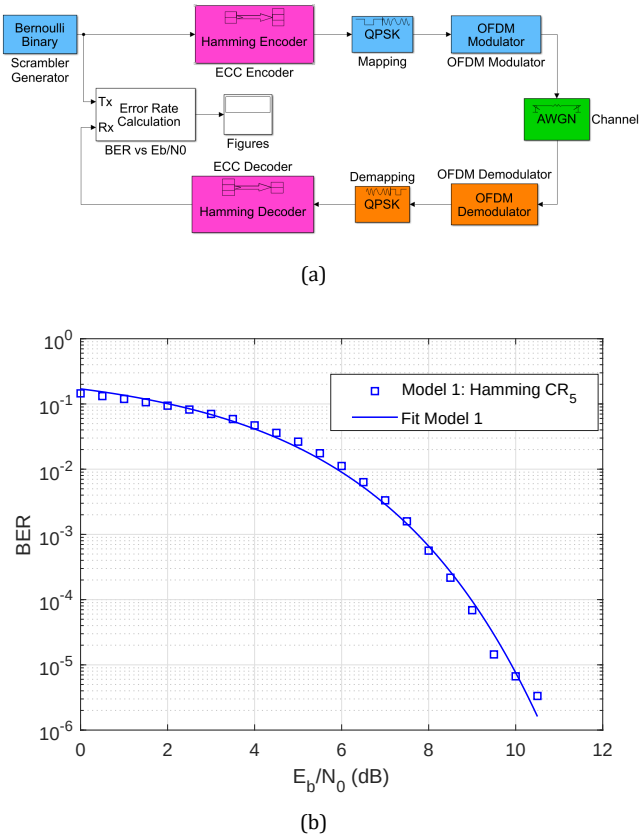


Fig. 7 – Model 1: (a)Short-frame OFDM communication model using MATLAB-Simulink tools (b) BER performance plotted on BERTool application

We use MATLAB-Simulink tools in order to model the simple Hamming code communication system $[63, 57]$ as shown in Fig. 7.

We generate the BER data for a simple Hamming code communication system model $[63, 57]$. This MATLAB-Simulink model (see Fig. 7 (a)) is displayed and run in real time on the models being simulated window (as shown in Fig. 6) for each value of the energy per bit to noise ratio $\frac{E_b}{N_0}$. Then, we obtain the plot in Fig. 7 (b) as the BER performance vs. $\frac{E_b}{N_0}$ for this model.

For Model 1, we have a very long coding rate (around 50 bits for the input message). In the following, we will compare to the concatenation of several encoders/decoders with shorter coding rates in order to prove that parallel Hamming has better performances than Hamming simple, while keeping the same simplicity of implementation.

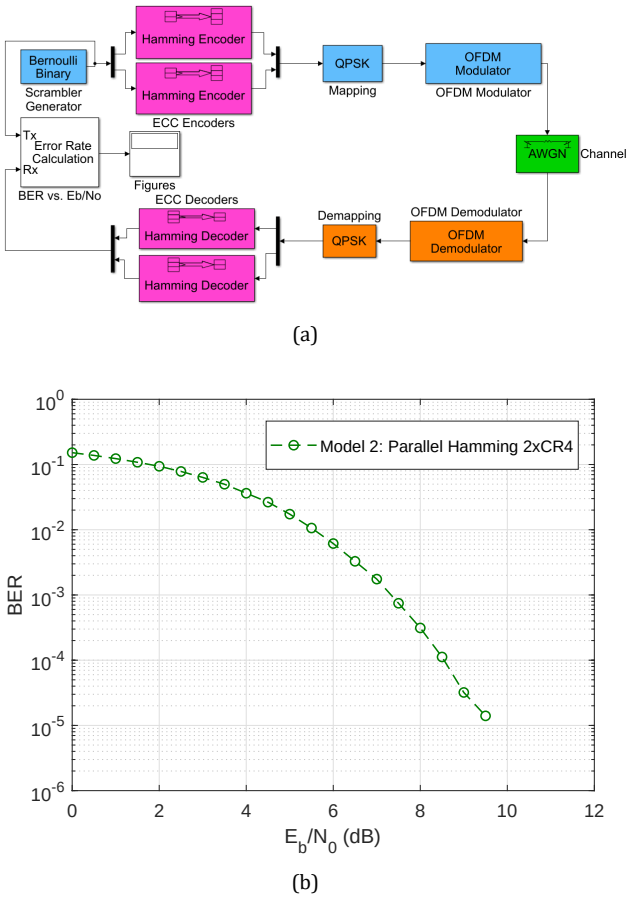


Fig. 8 – Model 2: (a)Short-frame OFDM communication model using MATLAB-Simulink tools (b) BER performance plotted on BERTool application

Model 2: Parallel Hamming code 2*[31,26]

In order to improve the correction capacity of Model 1: Simple Hamming code Model $[63, 57]$. Here, we propose to implement 2 couples of encoders/decoders in parallel respectively, so we can improve the correction capability of the Hamming code; which will be 2 bits out of 52 bits (20% of the total message), and detection would be 4 out of 52, while the first model could only detect 2 and correct 1. We use MATLAB-Simulink tools in order to model the parallel Hamming code communication system $2\times[31, 26]$ as shown in Fig. 8.

Model 3: Parallel Hamming code 5*[15,11]

For Model 3, we can cut the message on 5 times, each of $N = 11$ bits, which would be coded by 5 encoders Hamming $5 \times [15, 11]$. Thus, the capacity of correction in this case would be of 5 bits out of 55 (10% of the total message), and detection would be 10 out of 55, while the first model could only detect 2 and correct 1. We use MATLAB-Simulink tools in order to model the parallel Hamming code communication system $5 \times [15, 11]$ as shown in Fig. 9.

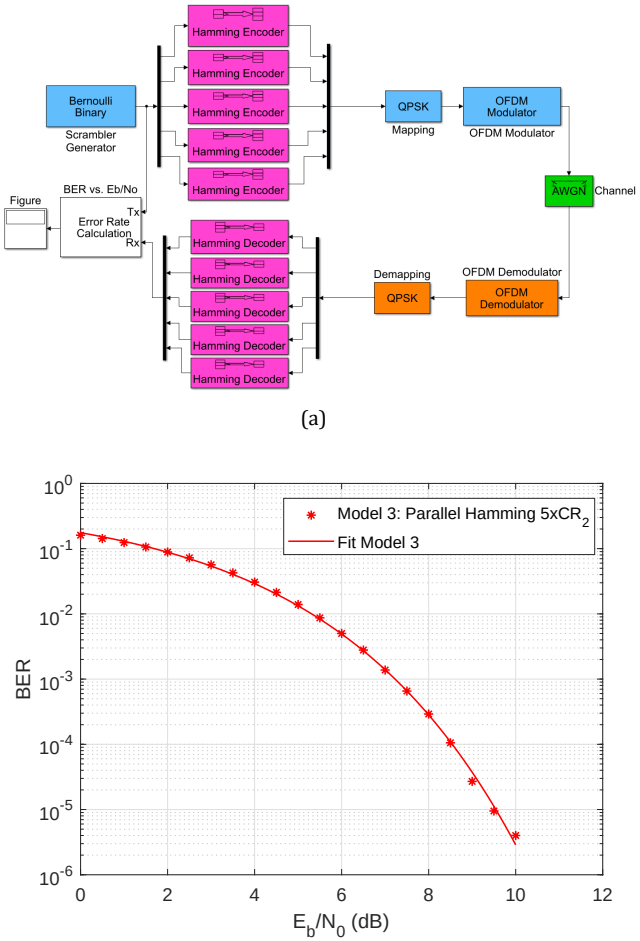


Fig. 9 – Model 3: (a)Short-frame OFDM communication model using MATLAB-Simulink tools (b) BER performance plotted on BERTool application

4.2 Comparison between parallel Hamming and simple Reed-Solomon

In this subsection, we will compare the previous models that are adapted to our technical requirements: Hamming (Model 1, Model 2, Model 3) to Reed-Solomon CR_6 (Model 4). We want to verify if there is a significant difference in BER performance: since parallel Hamming coding is very interesting in terms of simplicity and robustness, however Reed-Solomon is very interesting in terms of error correction capability.

In this scenario, we choose Model 4 where $CR_6 = 56/64$ is the simple Reed-Solomon coding rate (around 50 bits). Then we repeat the same simulations done for Model 1, Model 2 and Model 3 and for Reed-Solomon (Model 4) several times for each model and we calculate the average points for each value of $\frac{E_b}{N_0}$. We plot then the results in Fig. 10.

As we can see in Fig. 10, the parallel coding represents a gain in correction capacity, and both Model 2 and Model 3 converge faster to no errors than its equivalent with simple Hamming (Model 1). In our application case, and es-

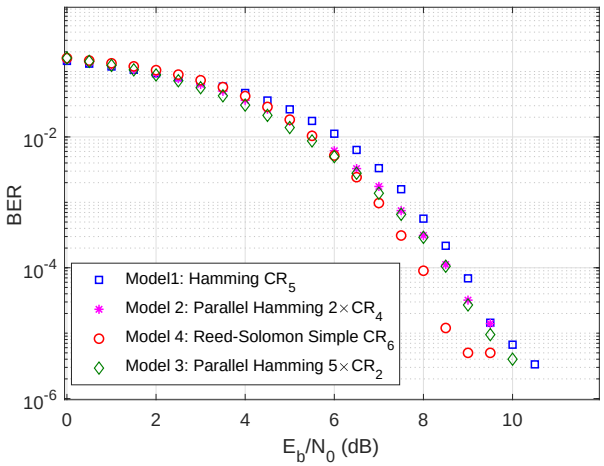


Fig. 10 – Comparing simple, parallel Hamming and Reed-Solomon codes for the same total size (around 50 bits).

pecially through our PLC channel, we have an average error probability almost equal to 3 errors for a packet of 50 bits. That 's why, Model 3 seems to be slightly better, especially towards the end, which can seem logical given the fact that Model 3 can have a correction of up to 5 errors, while Model 2 can only correct 2 errors.

Even if the Reed-Solomon code is apparently more efficient, we can see at the end of the Reed-Solomon simulation curve that the BER converges suddenly to zero: this is due to the complexity of the Reed-Solomon algorithm, MATLAB-Simulink tools couldn't simulate for long periods of time and we could only send a finite amount of bits before it made the simulation stop. The two last points of the Reed-Solomon simulation curve are stagnant because it's the limit of the simulation given its complexity.

However, Hamming curves (for Model 1, Model 2 and Model 3) can go to lower values of BER $\leq 10^{-6}$, which proves that it's easier and robust.

To give an order of magnitude: for Model 1 (simple Hamming code), we simulated 2 million bits (for each value of $\frac{E_b}{N_0}$), while for Model 4 (simple Reed-Solomon) we could only simulate 700 thousand bits which shows the complexity of the Reed-Solomon code when compared to the Hamming code.

Concerning their performance, in Fig. 10, we remark that Reed-Solomon is better, especially for $\frac{E_b}{N_0} \geq 8$ dB and with a high value of $\frac{E_b}{N_0}$ there are few errors to be corrected. Nevertheless, this gain in effectiveness for Reed-Solomon has a cost in complexity when compared to Model 3 of the Parallel Hamming coding.

Based on the previous analysis, we have discussed and validated via simulations the trade-off between complexity (Hamming is the easiest to code) and error correction capability (Reed-Solomon being the most effective). Table 4 summarizes the advantages and disadvantages of each ECC. Therefore, we have chosen to improve the correction capacity of the Hamming code instead of decreasing the complexity cost for the Reed-Solomon code since we have

Table 4 – ECCs comparison in terms of complexity of implementation and capacity of correction

ECCs	Advantages	Disadvantages
Simple Reed-Solomon	Very effective especially with burst errors; High correction capacity: can correct multiple errors simultaneously.	Complex and need more resources (LUT) than Hamming code.
Parallel Hamming $5\times$ $[15, 11]$	Easy to implement; Correction capacity: correct 5 of 10 detected errors for a 50 bits packet.	Not the most effective in terms of BER.

shown in Fig. 10, the curve of Model 3, which represents a new design of parallel Hamming coding, is closer to Reed-Solomon than the other models of Hamming coding.
For these reasons, we have chosen parallel Hamming $5\times$encoder/decoder CR_2 (Model 3) to be implemented next in VHDL in order to show that this solution uses a few resources and has a higher capability of correcting compared to the simple Hamming code.

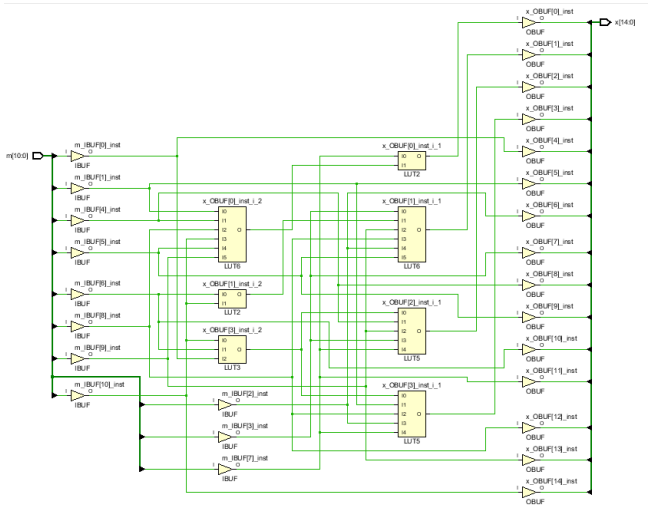
5. IMPLEMENTATION OF PARALLEL HAMMING ENCODER/DECODER

In this Section, we will analyse and validate the low complexity of Model 3 by implementing the design of the parallel enCOde/DECoder (CODEC) on an FPGA mock-up and simulating this design on VHDL code.

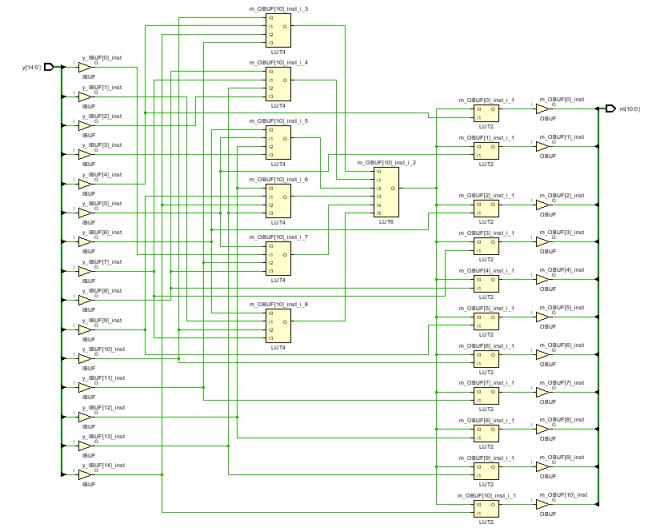
5.1 Parallel Hamming CODEC design

As we discussed before, the idea here is to make a trade-off between Hamming simplicity of implementation and Reed Solomon's capacity of correction and performance. In fact, with parallel Hamming encoder/decoder (Model 3), we will consume five times more resources than with simple one Hamming encoder/decoder, but we will have a notable improvement in terms of BER performance.

In Fig. 11, Hamming encoder/decoder $[15, 11]$ module is the base module to create our parallel Hamming encoder/decoder (Model 3), which is composed of $5\times$ Hamming encoders/decoders $[15, 11]$ added to Demux/Mux to concatenate the messages respectively.
The encoder/decoder circuit to compute the parity bits of the Hamming encoder/decoder $(11, 15)$ is shown in Fig. 11. These parity bits (P_1, P_2, P_4, P_8) are added to the in-



(a)



(b)

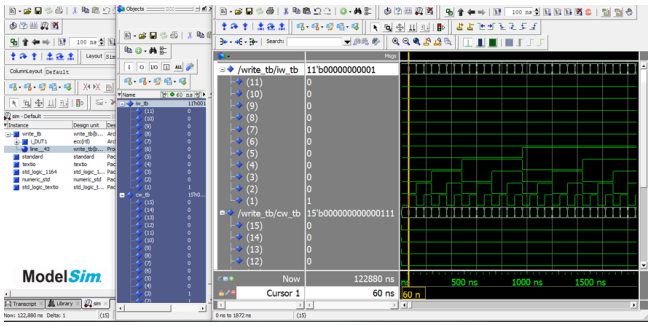
Fig. 11 – Design in terms of logic ports of: (a) One Hamming encoder $[15, 11]$; (b) One Hamming decoder (15,11)

formation bits $(D_1, ..., D_{11})$ at the transmitter (Hamming encoder) and then removed at the receiver (Hamming decoder) which is able to detect and correct errors.

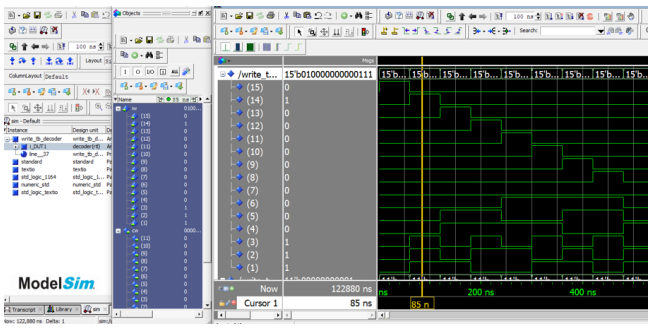
5.2 VHDL functional simulations

Next, we will create the VHDL functional simulation of the Hamming encoder/decoder $[15, 11]$ module based on two VHDL files, one for the encoder which calculates parity bits and outputs the original message with the parity bits added in specific positions and one for the decoder which recalculates the parity bits to locate the error, correct it, and outputs the original 11 bit message.

Therefore, we make a test bench for each file in order to test their encoding and correction capability. In the Fig. 12, we can see the simulation in ModelSim of the test benches of both the encoder and the decoder. The input is the 2048 values that the 11 bit message can take, the output is this same file so our encoder $[15, 11]$ works correctly.



(a)



(b)

Fig. 12 – VHDL functional simulation output of: (a) Hamming encoder $[15, 11]$; (b) Hamming decoder $(15, 11)$

Next, once this step has been simulated successfully in ModelSim, we we will implement and synthesize the code in Vivado which represent the Xilinx software for this FPGA mock-up.

5.3 Performance and experimental results

As we have seen before, we conclude our study of the Hamming encoder/decoder $[15, 11]$ by synthesizing and implementing the VHDL code using Vivado. We have used a Xilinx Spartan 7 mock-up for the implementation. Using the same code that was validated previously by ModelSim, we create one project for each module (encoder and decoder) and we seek to verify the consumed and utilized resources in the FPGA mock-up.

As we can see in Fig. 11, the synthesis was successful and we manage to create a logic port scheme of both encoder and decoder.

We denoted by LUT: a Look-up table which represents a small asynchronous SRAM used to implement combinational logic and by IO: an Input/Output Buffer.

Table 5 shows the amount of look-up tables that are taken by our encoder and decoder. With a total of $6 + 10 = 16$ $LUTs$ for the couple encoder/decoder, we confirm Hamming's code simplicity that was supposed in our analysis in Section 3.

Furthermore, knowing how much one Hamming encoder/decoder module would consume, we can deduce that our parallel coding with five couples of encoders/decoders would consume $5 \times 16 = 80$ $LUTs$, which is

still fewer than Reed-Solomon's and without recurring to multiplication or RAM-consuming tables. This amount of $LUTs$ consumed is a trivial and negligible amount compared to the available $LUTs$ in our board.

Table 5 – Parallel Hamming and Reed-Solomon encoder/decoder resources

ECC	Resource type	Utilisation
Hamming Encoder $5 \times [15, 11]$	LUT	30
	IO	130
Hamming Decoder $5 \times [15, 11]$	LUT	50
	IO	130
Reed-Solomon Encoder $(64, 56)$	LUT	90
	FDRE	20
Reed-Solomon Decoder $(64, 56)$	LUT	260
	FDRE	230

While our parallel Hamming code $5\times[15, 11]$ can be implemented using only 30 LUTs for the encoders and 50 LUTs for the decoders, Reed-Solomon code requires a higher number of blocks which are complex and uses more resources for the Reed-Solomon encoder/decoder up to $3\times$ parallel Hamming encoder and $5\times$parallel Hamming decoder. For more details, we have made the comparison between Hamming code and Reed-Solomon coding based on the aspects of memory occupation and running time in our work [13].

6. CONCLUSION

This paper deals with the design of an Error Correcting Code in a short-frame OFDM communication system. In order to respond favourably to the requirements of the low-powered sensor network, we have analysed the performance of several Error Correcting Codes (ECCs): such as Hamming code and Reed-Solomon code based on different parameters. Moreover, we have discussed the trade-off between the low implementation complexity (Hamming is the easiest to implement) and the high error correction capacity (Reed-Solomon being the most effective). Therefore, we have to: either improve the correction capacity of Hamming code, or decrease the complexity cost for Reed-Solomon code. That 's why, we propose a new design of parallel Hamming Coding. The parallel Hamming Code is chosen as five blocks of simple Hamming Code $[15, 11]$. Each encoder takes 11 bytes date block and generate 15 byte code block to be transmitted on the communication channel. After an implementation of this solution on an FPGA mock-up, we have shown that this parallel hamming encoder/decoder uses a few LUTs and has the capability of correcting up to five errors per message (packet with 55 bits). The encoder and decoder coding is done in VHDL on Xilinx tool. This process is implemented on Xilinx Spartan 7 FPGA.

Future work will include modelling the analog part of the PLC channel by the Matlab toolbox Simscape [16] instead of the AWGN channel estimation. Moreover, we will study also the possibility of decreasing the complexity cost [13] for the Reed-Solomon code by removing the multiplications on an FPGA processes and simplifying the complexity of the Reed-Solomon encoder/decoder design [17]. We will include then the implementation of the complete chain OFDM Tx-Rx on an FPGA.

REFERENCES

[1] S. Tilkioğlu, "OFDM Transmitter and Receiver Implementation on FPGA," in *Master Thesis dissertation*, Graduate School of Natural and Applied Sciences Electronic and Communication Engineering, 2018.

[2] Y. J. Qazi, J. A. Malik, and S. Muhammad, "Performance Evaluation of Error Correcting Techniques for OFDM Systems," in *Book*, 2014.

[3] A. Skylar *et al.*, "Optimization of Error Correcting Codes in FPGA Fabric Onboard Cube Satellites," in *Ph.D Thesis dissertation*, Montana State University-Bozeman Norm Asbjornson College of Engineering, 2019.

[4] A. M. Cruz *et al.*, "Low Complexity Turbo code Specification for Power-Line Communication (PLC)," in *2011 IEEE Electronics, Robotics and Automotive Mechanics Conference*. IEEE, 2011, pp. 349–354.

[5] L. M. Ionescu, C. Anton *et al.*, "Hardware implementation of BCH Error-correcting codes on a FPGA," in *International Journal of Intelligent Computing Research (IJICR)*, vol. 1, no. 3, pp. 148–153, 2010.

[6] S. Mahajan *et al.*, "BER Performance of Reed-Solomon Code Using M-ary FSK Modulation in AWGN Channel," in *International Journal of Advances in Science and Technology*, vol. 3, no. 1, pp. 7–15, 2011.

[7] V. Kavinilavu, S. Salivahanan *et al.*, "Implementation of Convolutional Encoder and Viterbi Decoder using Verilog HDL," in *International Conference on Electronics Computer Technology*, vol. 1. IEEE, 2011, pp. 297–300.

[8] Z. He, S. Roy *et al.*, "FPGA implementation of LDPC decoders based on joint row-column decoding algorithm," in *IEEE International Symposium on Circuits and Systems*. IEEE, 2007, pp. 1653–1656.

[9] C. H. Jones, "Communications over Aircraft Power Lines," in *IEEE International Symposium on Power Line Communications and Its Applications*. IEEE, 2006, pp. 149–154.

[10] V. Degardin, E. Simon *et al.*, "On the Possibility of Using PLC in Aircraft," in *IEEE International Symposium on Power Line Communications and Its Applications (ISPLC)*. IEEE, 2010, pp. 337–340.

[11] M. Shirvanimoghaddam *et al.*, "Short Block-length Codes for Ultra-Reliable Low Latency Communications," in *IEEE Communications Magazine*, vol. 57, no. 2, pp. 1–8, 2019.

[12] Z. Gao, P. Reviriego, *et al.*, "Fault tolerant parallel filters based on error correction codes," in *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 23, no. 2, pp. 384–387, 2014.

[13] J. Fernandez-Mayoralas, R. Masmoudi Ghodhbane, "Performance of Error Correcting Codes (ECCs) for Short-frame OFDM Sensors' Network," in *IoT & Sensor Networks Symposium of IEEE International Conference on Communications (ICC)*, 2021.

[14] James E. Gilley, "Bit-Error-Rate Simulation Using MatLab," in *Transcrypt International, Inc.*, August 19, 2003, pp. 1-7.

[15] T. Youssef and E. Abdelfattah, "Performance evaluation of different QAM techniques using Matlab/Simulink," in *IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, Farmingdale, NY, 2013, pp. 1–5.

[16] P. Degauque, V. Degardin, *et al.* , "Power-line communication: Channel characterization and modeling for transportation systems," in *IEEE Vehicular Technology Magazine*, 2015, 10(2), pp.28–37.

[17] P. Parvathi and P. R. Prasad, "FPGA based design and implementation of Reed-Solomon Encoder & Decoder for error detection and correction," in *Conference on Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG)*. IEEE, 2015, pp. 261–266.

AUTHORS

Dr. Raouia Masmoudi Ghodhbane (BSc.'10, MSc.'11, PhD.'15) received a Ph.D thesis in telecommunications from the University of Paris Cergy CY on the topic "Home automation telecommunications efficient in terms of energy consumption" in 2015. Currently, Raouia holds the position of research engineer in electronic interface and sensors' telemetry at the Safran Sensing Systems research department in Safran Tech. She has worked on different thematics: connected sensors in harsh aeronautic environments, low latency and low power sensor network communication, IoT networks, wireless networks, cognitive radio systems, opportunistic access to spectrum and green communications. Raouia is a member of the Technical Program Committees of the IEEE International Conference on Wireless for Space and Extreme Environments (WISEE).

Jorge Fernández-Mayoralas (MSc.'20) received an engineer degree in Integrated Circuit and Electronic Systems from CentraleSupelec in 2020. Since then, Jorge holds the position of Consultant on Energy, Utilities & Environment at Sia Partners. Jorge worked in his engineering intern-ship on the topic of "Integration of an error correcting code over a new aeronautic communication on an FPGA board" in Safran Tech.