Recent Results on the Implementation of a Burst Error and Burst Erasure Channel Emulator Using an FPGA Architecture

Caterina Travan, Francesca Vatta, and Fulvio Babich

***Abstract*—The behaviour of a transmission channel may be simulated using the performance abilities of current generation multiprocessing hardware, namely, a multicore Central Processing Unit (CPU), a general purpose Graphics Processing Unit (GPU), or a Field Programmable Gate Array (FPGA). These were investigated by Cullinan et al. in a recent paper (published in 2012) where these three devices capabilities were compared to determine which device would be best suited towards which specific task. In particular, it was shown that, for the application which is objective of our work (i.e., for a transmission channel simulation), the FPGA is 26.67 times faster than the GPU and 10.76 times faster than the CPU. Motivated by these results, in this paper we propose and present a direct hardware emulation. In particular, a Cyclone II FPGA architecture is implemented to simulate a burst error channel behaviour, in which errors are clustered together, and a burst erasure channel behaviour, in which the erasures are clustered together. The results presented in the paper are valid for any FPGA architecture that may be considered for this scope.**

I. Introduction

In telecommunications, the simulation of the burst error channel behaviour is an important topic, given the importance of the real scenarios in which this channel model may be applied (see next section). This topic has been widely considered in the recent literature through:

1) CPU implementations using:
   a) the software tool Matlab [1]: see, e.g., [2], in which four Matlab programs to simulate the Markov fading channel were developed, and [3], where a burst error correction scheme was implemented;
   b) the C/C++ libraries and interface: see, e.g., [4];
2) GPU implementations: see, e.g., [5], where the parallel error-resilient entropy coding (P-EREC) is shown to increase the resilience of the variable-length coding (VLC) bit-stream to random and burst errors, or [6], where a real-time RS decoding is obtained using a GPU direct transfer;
3) hardware measurement platforms: see, e.g., [7], where a single-hop network running the point-to-point protocol (PPP) connects a mobile host to a fixed host terminating a circuit-switched global system for mobile communications (GSM) connection, or [8], where the traces of traffic are taken from a 2.5 Gb/s interface of a broadband access router of Deutsche Telekom's internet protocol (IP) platform, connecting residential asymmetric digital subscriber line (ADSL) access lines to the backbone;
4) FPGA implementations: see, e.g., [9], where forward error correction (FEC) based on Reed-Solomon (RS) co-decoding was implemented on an FPGA, or [10], where an FPGA implementation of a channel co-decoder, interleaver and deinterleaver is presented, considering a Multiple Input and Multiple Output (MIMO) technology.

The simulation of the burst erasure channel – which is also important given the different real scenarios in which this channel model may be applied (see next section) – has been widely considered as well in the recent literature through:

1) CPU implementations using:
   a) the software tool Matlab: see, e.g., [11], where hybrid serial concatenated network codes are proposed to contrast burst erasures, and [12], where erasure-correcting codes are designed for channels with burst and random erasures;
   b) the C/C++ libraries and interface: see, e.g., [4] and [13];
2) GPU implementations: see, e.g., [14], where low-density generator matrix (LDGM) coding is proposed for counteracting long loss bursts, or [15], where the speed and accuracy of the quasi-cyclic low-density parity check (QC-LDPC) simulations are shown to be greatly increased by utilising the parallel architecture of GPUs;
3) hardware measurement platforms: see, e.g., [7], where artificial traces are generated with the same statistical characteristics as actual collected network traces, or [16], where real video and aggregated Internet traces are used to study the delay per packet in a burst erasure;
4) FPGA implementations: see, e.g., [17], where a design and FPGA implementation of a reconfigurable FEC decoder based on a RS code for WiMax networks is presented, or [18], where the RS decoder performance

Manuscript received May 2, 2019; revised December 19, 2019. Date of publication January 28, 2020. Date of current version January 28, 2020. The associate editor prof. Joško Radić has been coordinating the review of this manuscript and approved it for publication.

Part of this work was presented at the International Conference on Software, Telecommunications and Computer Networks, SOFTCOM '14, Split, Croatia, September 17-19, 2014.

Caterina Travan is with the TU Graz, Graz University of Technology, Graz, Austria (e-mail: caterina.travan@student.tugraz.at). Francesca Vatta and Fulvio Babich are with the DIA, University of Trieste, Trieste, Italy (e-mails: vatta@units.it, babich@units.it).

Digital Object Identifier (DOI): 10.24138/jcomss.v16i1.766

is reviewed as far as its FPGA implementations are concerned.

As put in evidence by the above presented literature overview, the simulation of a transmission channel behaviour may be implemented using the performance abilities of current generation multiprocessing hardware, namely, a multicore CPU, a general purpose GPU, or an FPGA. These were recently investigated in [19] where these three devices capabilities were compared to determine which device would be best suited towards which specific task. Many benchmarks were taken into account to compare all three platforms. In particular, the Fast Fourier Transform (FFT) benchmark was investigated since it is portable quite easily among all three devices. The timing differences, reported in Table 1 of [19], show that the GPU is the fastest in processing the FFT. But, if also the transfer times for the GPU to send and receive data are taken into account, the GPU becomes the slowest of all three platforms. In particular, when considering also the transfer time to send and receive data, the FPGA was shown to be 26.67 times faster than the GPU and 10.76 times faster than the CPU. Since in a channel simulation process, not only the signal processing speed is important, but also a low latency, i.e., the time between an input and its output response[1], which must be as short as possible, in [21] and [22] a hardware emulation has been proposed and implemented, with the objective of potentially accelerating the evaluation of the performance characterizing a communication system and the optimization of its parameters. In particular, inspired by the above mentioned results of [19] on FPGA speed, and by those of [10] and [17], all concerning FPGA implementations involving the burst error/erasure channel scenarios, in [22] was proposed a Cyclone II FPGA architecture implementation to simulate a burst error channel and a burst erasure channel behaviour, using the Altera development and education (DE1) board. This choice was driven principally by the results on FPGA performance speed reported in [19] and by the consideration that the use of an FPGA architecture allows to reproduce the same hardware environment that may be found in a real digital communication system.

Afterwards, further results on FPGA implementations, concerning the burst error/erasure channel scenarios, came out (reported in the already cited [9] and [18]), and in [20] further motivations appeared, with respect to the results of [19], for using FPGA implementations instead of CPU or GPU ones. In particular, in [20] the advantages and disadvantages of FPGA implementations were deeply analyzed, showing that FPGA implementations are to be preferred when low latency, very good connectivity (namely, very high bandwidth) and good energy efficiency are needed, even if the last property is, actually, application dependent. The only disadvantage of an FPGA implementation concerns the engineering costs, which are typically higher than those afforded to program or configure instruction based architectures (i.e., CPUs and GPUs).

Encouraged by the motivations put in evidence in [20] but also by the growing interest in FPGA implementations demonstrated by important companies such as Microsoft, using FPGAs in its data centers, and Amazon, offering them on cloud services, we decided to go on further, with respect to [22], in analyzing the FPGA implementation therein presented. In particular, in this paper we recall the principal results of [22], giving a more detailed description of the Altera DE1 board[2] and of the considered scenario, i.e., of the channel models and their emulation. Moreover, we give further results, with respect to [22], as far as the duration, in symbols, of the error bursts affecting the amplitude of the audio tracks considered are concerned. In addition, we give new results as far as the amplitude spectra of these audio tracks are concerned, and also show the amplitude and the root mean square (RMS) level of the error bursts affecting the amplitude of the considered audio tracks. Finally, to demonstrate the veracity and accuracy of the advantages, claimed in [20], of an FPGA implementation with respect to an instruction based architecture implementation, such as a CPU, we consider the core module used to realize the emulation of the channel, and compare its FPGA implementation (at disposal on the Altera DE1 board) with a software implementation of the same module, showing a great advantage of the FPGA implementation in terms of latency and energy efficiency.

The paper is organized as follows. Section II presents the considered scenario and the channel models, and Section III their emulation. In Section IV the implementation of the chosen FPGA architecture is presented. In Section V the obtained results are described and discussed. Finally, Section VI summarizes the main conclusions.

II. Considered Scenario and Channel Models

In telecommunications, a *burst error channel* is a data transmission channel in which errors are grouped together, concentrated in usually short time gaps. The result is that an uninterrupted sequence of bits, called *error burst*, is affected by errors. In other words, a burst of length b may be defined as a vector whose non-zero components are grouped together in a sequence of length b preceded and followed by at least one zero component [23].

Examples of error bursts can be found extensively in storage mediums. These errors may be the result of a material deterioration of the physical storage support, like a scrape on a disc surface, or the lack of oxide, or even the presence of dust particles, on a magnetic recording tape. Other examples of error bursts can be found in wireless communications. In this case, they may be the result of a temporary reduction in the power of the received signal (*fading*), leading to a demodulation failure of a certain number of symbols.

A binary erasure channel (BEC) is a natural generalization of the binary symmetric channel (BSC). The fundamental

[1]With an FPGA it is possible to obtain a latency around or below 1 microsecond, whereas with a CPU a latency smaller than 50 microseconds is already very good [20]. Moreover, the latency of an FPGA is much more deterministic. One of the main reasons for this low latency is that FPGAs can be much more specialized: they do not depend on the generic operating system, and communication does not have to go via generic buses (such as universal serial bus (USB) or peripheral component interconnect express (PCIe)).

[2]Recently, Intel bought Altera, one of the largest producers of FPGAs, paying $ 16.7 billion, thus making it their largest acquisition ever [20].

difference is that, in this case, it is assumed that the receiver can also provide the decoder with an additional symbol, in addition to the characters of the alphabet $\mathbb{F}_2$. This symbol, denoted by "?", corresponds to received values considered a priori not intelligible. To implement this type of receiver, it is sufficient to monitor the quality of the incoming signal and decide that when this is below an arbitrarily set threshold, the character is to be discarded. In other words, we consider a digital binary transmitter (that can transmit the two values "0" and "1" on a transmission channel). A decoder that implements the binary erasure channel can be realized by means of the function $\phi : [0, 1] \mapsto \{0, 1, ?\}$.

When such erasures occur in a burst we call this channel a *burst erasure channel*. In the majority of space communication systems, the data frames are usually protected against channel errors using a serial concatenation of a RS outer code followed by a convolutional inner code. When a data frame is incorrectly decoded it is flagged as incorrect and thus erased from the data stream, giving rise to a flow of data including bursts of erasures. Other similar contexts are: deep space communications over noisy channels, where certain packets are not decodable leaving gaps, or bursts of erasures, in the data stream [24]; free space optical links, where climatic phenomena may preclude photons detection during certain time intervals [25]; communication scenarios implying the use of on-the-fly changes of code rate, or modulation scheme [26], or also the use of streaming applications (e.g., Internet video, mobile games, etc.). Since, in this case, cyclic redundancy check (CRC) codes, usually used in this kind of applications, are not sufficient to reduce the delay in the transmission experienced by the receiver, the effect produced by this delay is like as if an erasure had occurred [27].

A. Gilbert-Elliott Channel Model

In the early 1960's, Gilbert [28] and Elliott [29] introduced a channel model, based on a 2-state Markov approach [30], still extensively employed to statistically characterize the transmission channels through their error patterns description [31] and to analyse the efficiency of channel coding schemes [32].

The model considered a good (G) and bad (B) state generating errors as independent events. The two states G and B generate these error events at a rate $1 - k$ and $1 - h$, respectively. The model is depicted in Fig. 1. When considering, for instance, a *burst error channe*l, the good state can be interpreted as the state in which a bit is correctly received, whereas the bad state as the state in which a received bit is in error. When considering, instead, a *burst erasure channel*, the good state can be interpreted as the state in which a packet is correctly received, whereas the bad state as the state in which a packet is erased.

Define the transition matrix T as

$$T = \begin{pmatrix} 1-p & p \\ r & 1-r \end{pmatrix} \tag{1}$$

with the two transitions probabilities p and r given by

$$p = P(s_t = B | s_{t-1} = G) \tag{2}$$

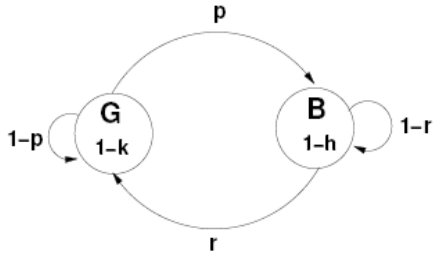

Fig. 1. Gilbert-Elliott channel model.

$$r = P(s_t = G | s_{t-1} = B) \tag{3}$$

being s_t the state at time t.

Given the stationary state probabilities π_G and π_B, existing for $p > 0$ and $r < 1$ [8], the error probability P_E in the steady state can be expressed as:

$$P_E = (1 - k)\pi_G + (1 - h)\pi_B \tag{4}$$

where

$$\pi_G = \frac{r}{p + r} \tag{5}$$

$$\pi_B = \frac{p}{p + r} \tag{6}$$

III. Channel Emulation

The Altera DE1 board, shown in Fig. 2, includes the following hardware:

- Altera Cyclone II 2C20 FPGA device;
- Altera serial configuration device EPCS4;
- universal serial bus (USB) blaster (on board) for programming and user application programming interface (API) control supporting both joint test action group (JTAG) and active serial (AS) programming modes;
- 512-Kbyte static random access memory (SRAM);
- 8-Mbyte synchronous dynamic random access memory (SDRAM);
- 4-Mbyte flash memory;
- secure digital (SD) card socket;
- 4 pushbutton switches;
- 10 toggle switches;
- 10 red user light emitting diodes (LEDs);
- 8 green user LEDs;
- 50-MHz oscillator, 27-MHz oscillator and 24-MHz oscillator for clock sources;
- 24-bit compact disc (CD)-quality audio coder-decoder (CODEC) with line-in, line-out, and microphone-in jacks;
- video graphics array (VGA) digital to analogue converter (DAC) with VGA-out connector;
- electronic industries alliance recommended standard 232 (EIA RS-232) transceiver and 9-pin connector;
- personal system/2 (PS/2) mouse/keyboard connector;
- 2 40-pin expansion headers with resistor protection.

In addition, the DE1 board supports the management via software of standard input/output (I/O) interfaces and the

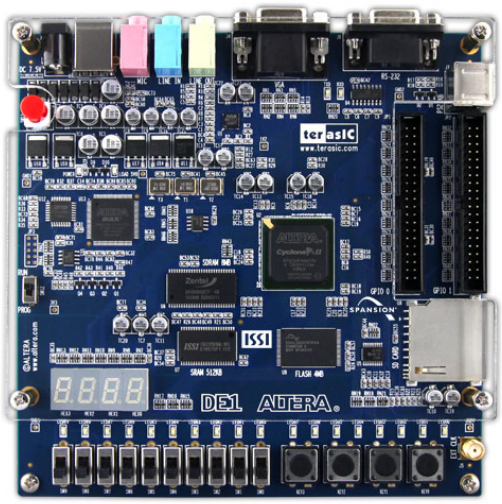

Fig. 2. Altera DE1 board.

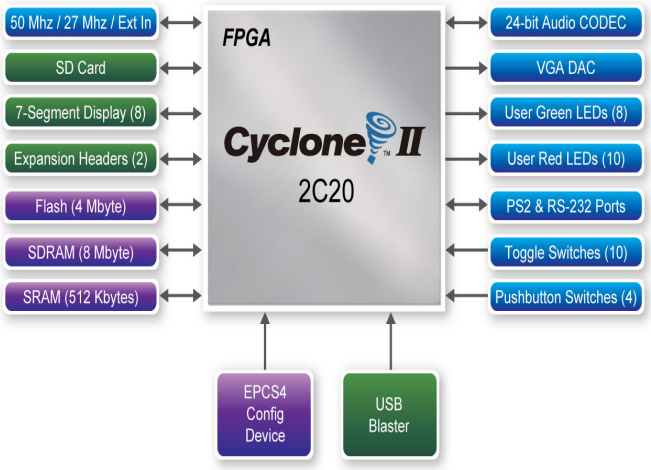

Fig. 3. Altera DE1 block diagram.

access to the various components through a control panel. Its blockdiagram is shown in Fig. 3.

The description language used for programming was the Verilog hardware description language (HDL) using the Altera Quartus II 13.0 as development environment.

Verilog, standardized as IEEE 1364, is a hardware description language used in electronic systems modelling. It is usually employed in the design and validation of analog, digital, and mixed-signal circuits, as well as of genetic circuits.

HDLs (such as, e.g., Verilog) are comparable to software programming languages because they contain procedures allowing to express, e.g., a signal transmission time and its power (sensitiveness), using two kinds of assignment operators: a non-blocking ($<=$) and a blocking ($=$) assignment. The non-blocking one permits, e.g., the description of a state-machine evolution with no need to declare and use non-permanent cache variables. Thanks to these options, that belong to Verilog's HDL semantics, engineers are able to rapidly describe circuits of considerable size in a proportionately condensed and compressed format.

Verilog HDL has a statement structure comparable to the C programming language, universally used in software development for telecommunication systems engineering area. Similar to the C programming language, Verilog is case dependent and has a principal preprocessor (even if not so highly developed as that of ANSI C/C++). The control statements (*if-else* condition, *for* cycle, *while* cycle, etc.) are similar, and the operator priority is consistent with the C programming language, even if there are syntactic differences, such as the bit-widths for variable declarations, delimitation of the procedures structure (Verilog makes use of a *begin-end* structure rather than opening ({) and closing curly brackets (})), and a multiplicity of additional insignificant dissimilarities. Moreover, Verilog necessitates precise dimension variables, whereas in the C programming language these dimensions are deduced from the variable's *type* (for example an integer type may have a dimension of 8 bits).

A design performed through the Verilog HDL involves a ranking in which the modules enclose the prototype hierarchy, and are in communication with further modules via an ensemble of stated ports (input, output, and bidirectional). Inside a module, there is a number of possible combinations of the following statements: sequential and parallel statements, variable declarations, and instances of other modules (sub-hierarchies). The sequential statements are located within a begin-end block and are executed in sequential order, but the blocks are executed in parallel, so that Verilog may be considered a dataflow language.

The channel has been simulated defining a finite state machine through a counter threshold, programmable by means of the switches at disposal on the Altera DE1 board. The module *burst adder* was used to realize the emulation of the channel. This receives 8 bits long input symbols (i.e., bytes), and outputs 8 bits long symbols. This module may work in two modes[3]:

1) the so called *additive mode*, in which a pseudorandom 8-bit word is added, using a linear feedback shift register (LFSR): this mode corresponds to the burst error channel;
2) the so called *multiplicative mode*, returning a zero output word: this mode corresponds to the burst erasure channel.

In this module, the length of the error burst is programmable by means of a 4-bit bus: thus, 16 different values may be selected. The binary configurations of the switches SW [2-5], through which it is possible to choose the duration of the error burst or erasure burst are reported in Table I of [22].

Moreover, in order to emulate an inter-arrival time having a non-zero dispersion, a LFSR has been used, taking its output to determine the inter-burst interval. To determine the beginning of the error burst or of the erasure burst, the *trigger* input signal has been used, within the module *burst adder*, driven

[3]In Table II of [22] are reported the binary configurations of the switches SW [0] and SW [1] through which the channel modes can be enabled.

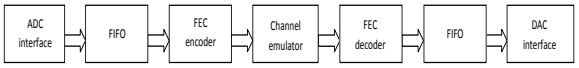

Fig. 4. Block scheme of the implemented system.

by the output of one of the two LFSRs. The trigger signal was used to start the threshold counter.

This emulation corresponds to the Gilbert-Elliott model, described in the previous subsection, with the following parameters:

1) the transition probability from the error-free state (G in Fig. 1) to the burst/erasure state (B in Fig. 1) is determined by the LFSR and is selected as $p \approx 0.5$;
2) the permanence in the burst/erasure state (with probability $1 - r$ in Fig. 1) is selected to be deterministic, i.e., $r = 0$.

The portion of the scheme that emulates the channel is shown in Fig. 1 of [22].

IV. FPGA Architecture Implementation

As information source, the output of a programmable audio codec (Wolfson WM8731) was used (integrated in the Altera DE1 board).

A. Structure of the Implemented System

The block scheme of the implemented system is shown in Fig. 4. It was described in detail in Section III.A of [22].

B. Analog to Digital Converter (ADC) Interface

The WM8731 is a low power stereo codec with an integrated headphone driver. The sample bit depths of the digital audio input words can go from 16 to 32 bits, whereas the sampling frequencies from 8 kHz to 96 kHz. In this paper, the sampling rate was set to 48 kHz in order to represent the entire audible band, while the sample bit depth was fixed in 16 bits. Moreover, the codec presents various options for the representation of the sample. Among them, the configuration master in DSP mode was selected, so that to make the system as simple and robust as possible with respect to the operations to be performed on data. In particular, since the package DSP mode contains a stereo sample, this will have double bit depth, equal to 32 bits. In Fig. 3 of [22] the temporal specification of the DSP mode is shown.

The on-board stereo ADC output is available on the digital audio interface. In the ADC also an optional digital high pass filter is available in order to remove unwanted dc components from the audio signal. The on-board DAC accepts digital audio signals from the digital audio interface. Three de-emphasis digital filers are available at 32 kHz, 44.1 kHz and 48 kHz: these can be applied, by software control, to the digital data. The DAC outputs are available both at line level and through a headphone amplifier. Moreover, the headphone output volume is adjustable in the analogue domain over a range of +6 dB to –73 dB and can be muted, too.

In order to create a robust system, it was decided to guarantee a high insulation between the ADC and DAC interfaces and the internal processing blocks. This goal was achieved by introducing the input and output buffers.

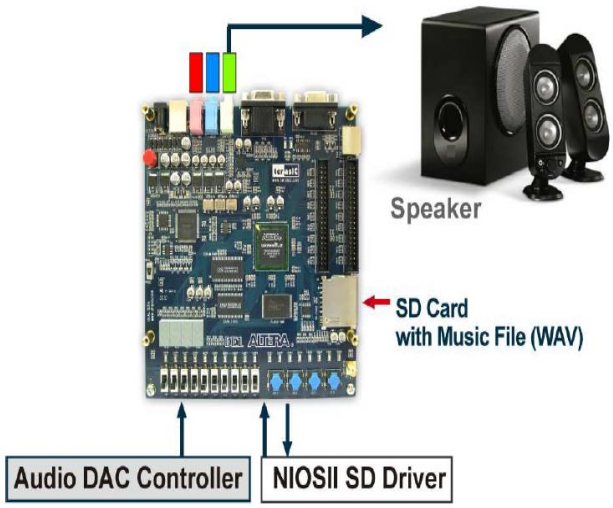

Fig. 5. The setup for the implemented test.

C. Input and Output Buffers

Intel® provides first in first out (FIFO) Intel FPGA intellectual property (IP) core through the parameterizable single-clock FIFO (SC-FIFO) and dual-clock FIFO (DC-FIFO) functions. The FIFO functions are principally applied in data buffering applications that comply with the first-in-first-out data flow, both in synchronous or asynchronous clock domains.

The specific names of the FIFO functions are:

- SC-FIFO: single-clock FIFO;
- DC-FIFO: dual-clock FIFO (supports same port widths for input and output data);
- DC-FIFO_MIXED_WIDTHS: dual-clock FIFO (supports different port widths for input and output data).

To implement the input (and output) FIFO buffer, the parameters of the structure DC-FIFO, available in Quartus II, were set as follows:

- differentiated clock in reading and writing,
- word bit depth of 1 bit in writing (reading) and 8 bit in reading (writing), and
- length of 256 symbols (4096 bits).

Fig. 4 in [22] shows the scheme of the FIFO used (since the input has bit depth 8, this is the output FIFO).

V. Results and Discussion

The implemented system has been tested by connecting an audio source and a speaker via the connectors provided on the Altera DE1 board, as shown in Fig. 5. To read the music data, stored in the SD card, we have used the Nios II processor, whereas, to play the music, we have made use of the Wolfson WM8731 audio CODEC.

The proper configuration for the channel mode has been selected as shown in Table II of [22]. In this way, the impact of the channel on the test signal can be evaluated

1) in the ideal case where, as might be expected, the channel does not affect the original signal;
2) in the burst error channel case, where the original signal is degraded by the superposition of impulse noise, causing a hearing impairment similar to the crackling pop-corn (in fact, the noise burst in the literature is also called *pop-corn noise*);
3) in the burst erasure channel case, where an out of service, simulated by replacing the symbol to be erased with a zero symbol, causes a hearing impairment similar to a sort of crackling in the background.

The duration of the error bursts, that has an evident effect on the disturb affecting the signal, may be varied acting on the control bus. The perceived disturb is much more evident in the "burst error" case with respect to the "burst erasure" one.

As part of the system test, a shortened code RS(204, 188)[4] has been added to the processing chain because, being RS codes non-binary cyclic error-correcting codes, they are very advantageous in burst error and burst erasure correction [33]. Moreover, being the use of the RS(204, 188) code recommended by all DVB standards [34], as such it has been included in many FPGA implementations, like in the already cited [9] and [17].

The shortened code RS(204, 188) is obtained from the code RS(255, 239), limiting to 204 the number of useful bytes transmitted in each packet, while the number of redundancy bytes remains 16, maintaining the ability to correct 8 bytes or 16, in the case of erasure. In fact, by adding t check symbols to the data, an RS code can detect any combination of up to t erroneous symbols, or correct up to $\lfloor t/2 \rfloor$ symbols. As an erasure code, it can correct up to t known erasures, or it can detect and correct combinations of errors and erasures [33]. Furthermore, RS codes are suitable as multiple-burst bit-error correcting codes, since a sequence of $b+1$ consecutive bit errors can affect at most two symbols of size b. Moreover, they have been shown in [24] to be not only a viable but also an optimal solution for the burst erasure channel application.

Both the coding block and the decoding one have been implemented by providing the appropriate parameters, summarized in Table I (rewriting Table III of [22]), to the compiler in the Altera Quartus II environment. Moreover, the Reed Solomon code described above, once added in the processing chain, has been verified to have an error correction capability matching exactly the Reiger's bound [35].

[4]The choice of $k = 188$ was determined by the requirement, considered important at the beginning of the 1990's, to facilitate the transfer of compressed audio and video information on radio bridge and satellite communications systems, based on the asynchronous transfer mode (ATM) standard. This system, designed for telephone switching and transmission, requires data to be organized in cells of 53 bytes, of which 48 bytes of data and 5 bytes of header. The value of 188 bytes (of which the first 4 constitute the heading) was chosen to facilitate the remapping of the useful data, coded according to the moving picture experts group (MPEG) standards, in the ATM cells.

TABLE I
RS CODE SPECIFICATION FOR DVB STANDARDS

Symbol depth (Bits/Symbol)	$m = 8$
Field Generator polynomial	$p(x) = x^8 + x^4 + x^3 + x^2 + 1$
Code Generator polynomial	$g(x) = (x - \alpha^0)(x - \alpha^1) \cdots (x - \alpha^{15})$
Symbols per block (code length)	$n = 204$
Data symbols (message length)	$k = 188$

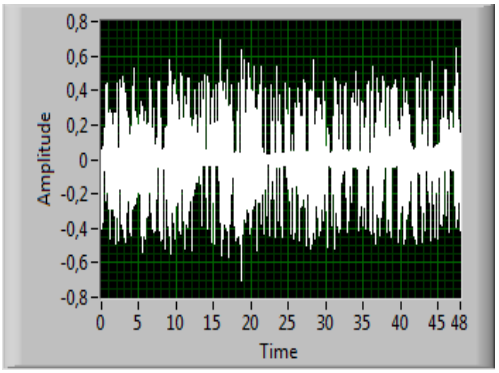

Fig. 6. Amplitude of an audio track affected by error bursts of 1 symbol.

A. Objective Measures

In order to obtain objective measures for the channel emulated in Verilog HDL, a LabVIEW program[5] has been implemented so that to capture the audio signal output from the channel and provide a graphical output.

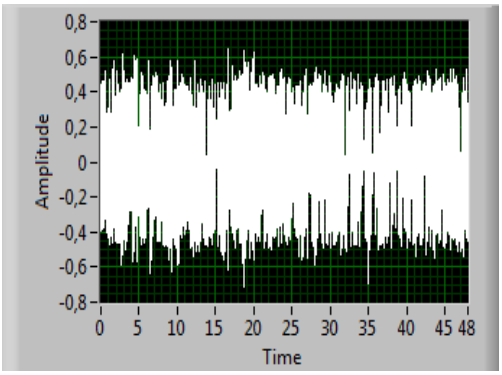

Fig. 7. Amplitude of an audio track affected by error bursts of 15 symbols.

The Virtual Instrument "Compute_audio_degradation.vi" receives in input 32 audio tracks containing the testing track affected by error bursts of duration from 1 to 32 symbols, previously acquired by another VI and saved in a specific folder. The program allows viewing the amplitude and am-

[5]LabVIEW is fundamentally a graphical programming language in which the user can set up the program to control and cache data. It is called virtual instrument (VI) because its aspect and working imitate physical instruments, such as oscilloscopes and multimeters. A user interface, or front panel, can be effortlessly set up, with controls and indicators. Controls are knobs, push buttons, dials, and other input mechanisms. Indicators are graphs, LEDs, and other output displays.

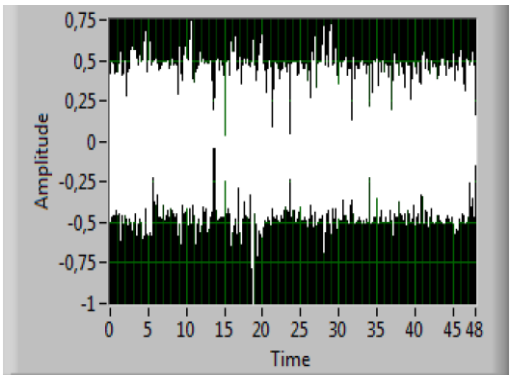

Fig. 8. Amplitude of an audio track affected by error bursts of 30 symbols.

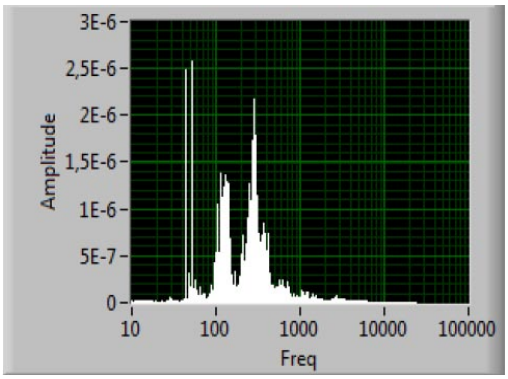

Fig. 11. Spectrum of an audio track affected by error bursts of 30 symbols.

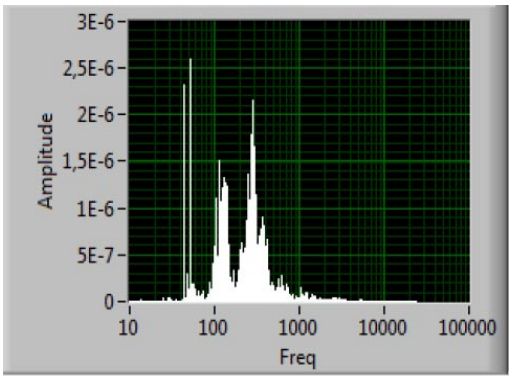

Fig. 9. Spectrum of an audio track affected by error bursts of 1 symbol.

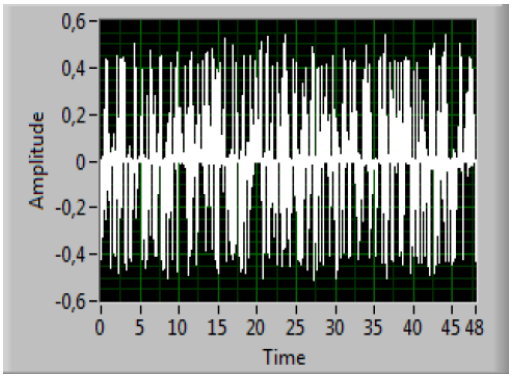

Fig. 12. Amplitude of an error burst of 1 symbol duration.

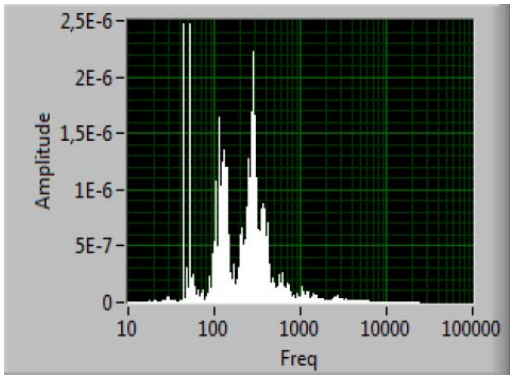

Fig. 10. Spectrum of an audio track affected by error bursts of 15 symbols.

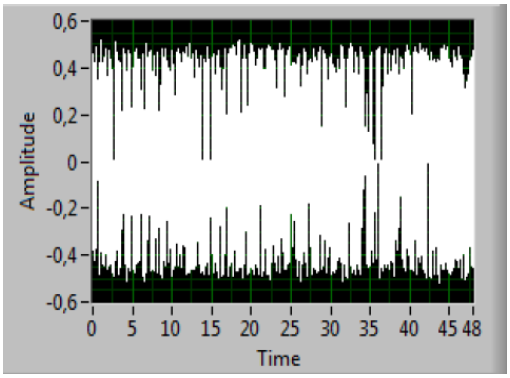

Fig. 13. Amplitude of an error burst of 15 symbols duration.

plitude spectrum of each of these tracks and to perform some processing on them.

In Figs. 6, 7, and 8 we report, as example, the amplitude of an audio track affected by error bursts of duration 1, 15, and 30 symbols, respectively, and in Figs. 9, 10, and 11 their respective amplitude spectra.

The noise can be extracted and processed, too. In Figs. 12, 13, and 14 we report, as example, the amplitude of an error burst of 1, 15, and 30 symbols duration, respectively. Finally, in Fig. 15 we also report the error RMS level with respect to the burst length measured in symbols (of 8 bits each).

In Fig. 15 it is shown that the error RMS level tends towards a horizontal asymptote, i.e., towards a constant, as the duration, in symbols, of the error burst increases. In this sense, as far as the amplitudes of the error bursts are concerned (shown in Figs. 12, 13, and 14), we can appreciate a great difference in the error amplitude comparing Fig. 12, showing the amplitude of an error burst of 1 symbol duration, with Fig. 13, showing the amplitude of an error burst of 15 symbols duration, since the difference between the respective error RMS levels, shown in Fig. 15, is significant. On the other hand, the difference between Fig. 13, showing the amplitude of an error burst of 15 symbols duration, and Fig. 14, showing the amplitude of an error burst of 30 symbols duration, is hard to be appreciated, since the difference between the respective error RMS levels,

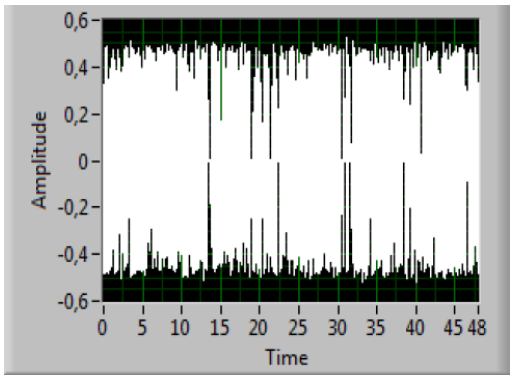

Fig. 14. Amplitude of an error burst of 30 symbols duration.

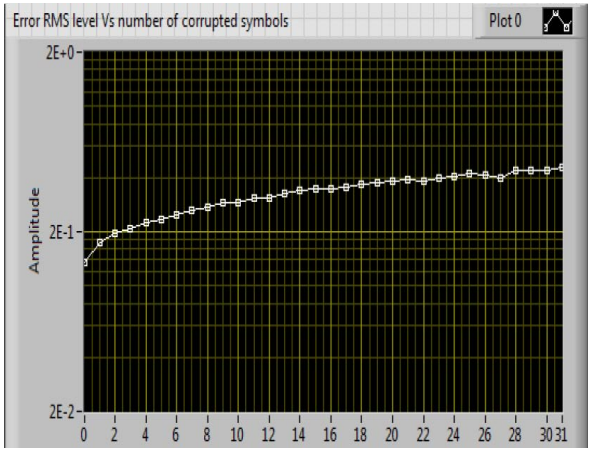

Fig. 15. Error RMS level with respect to the burst length measured in symbols (of 8 bits each).

shown in Fig. 15, is very small. Similarly, as far as the amplitudes of the audio tracks affected by error bursts are concerned (shown in Figs. 6, 7, and 8), we can appreciate a great difference in the audio track amplitudes comparing Fig. 6, showing the amplitude of an audio track affected by an error bursts of 1 symbol duration, with Fig. 7, showing the amplitude of an audio track affected by an error bursts of 15 symbols duration, since the difference between the respective error RMS levels, shown in Fig. 15, is significant. On the other hand, the difference between Fig. 7, showing the amplitude of an audio track affected by an error bursts of 15 symbols duration, and Fig. 8, showing the amplitude of an audio track affected by an error bursts of 30 symbols duration, is hard to be appreciated, since the difference between the respective error RMS levels, shown in Fig. 15, is very small. Finally, as far as the amplitude spectra of the audio tracks affected by error bursts are concerned (shown in Figs. 9, 10, and 11), we can appreciate a certain difference in the areas of the 4 principal Dirac's pulses, visible in the amplitude spectra at about 60, 70, 100, and 500 Hz, respectively, comparing Fig. 9, showing the amplitude spectrum of an audio track affected by an error bursts of 1 symbol duration, Fig. 10, showing the amplitude spectrum of an audio track affected by an error bursts of 15 symbols duration, and Fig. 11, showing the amplitude spectrum of an audio track affected by an error bursts of 30 symbols duration. Again, the difference between Fig. 9 and Fig. 10 is more evident than the difference between Fig. 10 and Fig. 11. On the other hand, there is no appreciable difference in signal distortion, since the bandwidth remains essentially the same in all the three cases examined.

B. Comparison of the FPGA Implementation with a CPU one

As mentioned in Section III, the channel has been simulated defining a finite state machine through a counter threshold. The core module used to realize the emulation of the channel was the so-called *burst adder* implementing an LFSR, which is a *linear recurrent generator* [36]. To implement an LFSR, a sequence of shift registers is needed, generating one bit for each iteration of the algorithm. By means of an *XOR* the contents of some registers are summed up over $\mathbb{F}_2$ to obtain a feedback input to the first register: the feedback connections are defined through a feedback polynomial which is a primitive irreducible polynomial of degree 8 generating a pseudorandom 8-bit word (see Figure 3(a) in [36]).

Being the LFSR the core module used to realize the emulation of the channel, we also compared its FPGA implementation (at disposal on the Altera DE1 board) with a software implementation of the same LFSR, based on [37], running on a workstation with an Intel® Core™ Duo T5870 2.0 GHz CPU. We used the Intel® Math Kernel Library (MKL) and compiled our C code using the Intel® C++ Compiler. The throughputs we obtained, measured in terms of generated billion samples per seconds, were 0.85 at the CPU output and 20.90 at the FPGA output. This means that the FPGA implementation can achieve a ~ 24.59 speedup with respect to the CPU implementation.

We measured the power consumptions of the above mentioned two implementations, too. The idle power for both the machines was the same, i.e., $\sim$ 100 W. When running the LFSR implementation on the FPGA, the power consumption rose up to $\sim$ 120 W, whereas when running it on the CPU the power consumption rose up to $\sim$ 160 W. Thus, the extra power consumptions were $\sim$ 60 W and $\sim$ 20 W for the CPU and the FPGA, respectively. Dividing the throughput by the extra power consumption we obtain the generated billion samples per Joule for each of the two implementations. These two quantities are ~ 0.01 and ~ 1.04 for the CPU and the FPGA, respectively. Thus, the FPGA is not only much more faster than the CPU but also much more energy efficient.

VI. Conclusions and Open Problems

In this paper, a burst error channel and a burst erasure channel simulator have been implemented in a Cyclone II FPGA architecture. In telecommunications, a *burst error channel* is a data transmission channel in which errors are grouped together, concentrated in usually short time gaps. The result is that an uninterrupted sequence of bits, called *error burst*, is affected by errors. A binary erasure channel is instead a natural generalization of the BSC, in which it is assumed that the receiver can also provide the decoder with an additional symbol, in addition to the characters of the alphabet $\mathbb{F}_2$. This

symbol corresponds to received values considered a priori not intelligible, so that to be considered erased. When such erasures occur in a burst, this channel is called *burst erasure channel*.

The software simulation of a transmission channel behaviour is usually easy to be implemented but also very time consuming. In particular, the time needed to perform a software simulation depends on how complicated is the model of the channel to be simulated (see, e.g., [38] and [39]) and also on the bit error rates (BERs) that need to be evaluated (see, e.g., [40] and [41]). Since hardware emulation has the advantage that the evaluation process occurs in hardware, in place of occurring in the virtual setting of a simulator implemented using a standard software, this potentially accelerates the evaluation of the performance characterizing a communication system and the optimization of its parameters. Thus, in this paper a hardware emulation has been proposed and implemented, based on the results of [22]. Furthermore, the fact that a programmable logic FPGA has been chosen is remarkably interesting because it permits the usage of the same hardware for emulation that is really employed in actual telecommunications systems.

The implementation suggested for channel emulation is straightforward and functional and it permits tailoring the model to a variety of real cases. As a matter of fact, the burst duration may be adjusted operating on the switches, as well as the burst inter-arrival time, which can be tuned setting the clock driving the LFSR module. In particular, acting on the LFSR polynomial the transition probability from the G to the B state can also be varied. As far as the LFSR module in particular is concerned, we also compared its FPGA implementation (at disposal on the Altera DE1 board) with a software implementation of the same LFSR, showing a great advantage of the FPGA implementation in terms of latency and energy efficiency.

After surveying the burst error correcting techniques presented in the literature, a Reed Solomon code was chosen (see, e.g., [42]). This choice has opened on to inspiring results. In addition, owing to the employment of an audio signal, it has been feasible to be promptly aware of the advantages emerging from this encoding scheme.

Possible future developments may consider the use of different coding schemes suitable for this application such as iteratively decoded low density parity check (LDPC) codes (see, e.g., [43]-[48]) or related structures, including irregular repeat-accumulate (IRA) codes [49], generalized IRA codes (see, e.g., [50]), tornado codes and protograph-based codes [26], turbo codes (see, e.g., [51]-[53]), serial and hybrid concatenated codes (see, e.g. [54]-[56]), concatenated turbo codes [57], and product codes (see, e.g., [58]-[60]).

REFERENCES

[1] Matlab: The Language of Technical Computing, Mathworks, Inc., 2012. Available at: http://www.mathworks.co.uk/products/datasheets/pdf/matlab.pdf.

[2] T. Adam and U. Hashim, "Simulation of fading channel and burst error behavior of state-3 memoryless Markov model", *Journal of Selected Areas in Telecommunications (JSAT)*, Cyber Journals: Multidisciplinary Journals in Science and Technology, November Edition, 2011, pp. 37-43.

[3] S. Kuchhal, "Simulation of the burst error correction using Matlab", *International Journal of Scientific Research*, vol. 2, no. 9, June 2012, pp. 162-165. DOI: 10.15373/22778179/SEP2013/58.

[4] J. A. Briffa and S. Wesemeyer, "SimCommSys: taking the errors out of error-correcting code simulations", *The Journal of Engineering*, vol. 2014, no. 6, 2014, pp. 332-339. DOI: 10.1049/joe.2014.0055.

[5] Y. Dai, Y. Fang, D. He, and B. Huang, "Parallel design for error-resilient entropy coding algorithm on GPU", *Journal of Parallel and Distributed Computing*, vol. 73, no. 4, April 2013, pp. 411-419. DOI: 10.1016/j.jpdc.2012.12.008.

[6] T. Suzuki, S.-Y. Kim, J.-I. Kani, T. Hanawa, K.-I. Suzuki, and A. Otaka, "Demonstration of 10-Gbps real-time Reed-Solomon decoding using GPU direct transfer and kernel scheduling for flexible access systems", *Journal of Lightwave Technology*, vol. 36, no. 10, 2018, pp.1875-1881. DOI: 10.1109/JLT.2018.2793938.

[7] A. Konrad, B. Y. Zhao, A. D. Joseph, and R. Ludwig, "A Markov-based channel model algorithm for wireless networks", *Wireless Networks*, vol. 9, no. 3, January 2003, pp. 189-199. DOI: 10.1023/A:1022869025953.

[8] G. Hasslinger and O. Hohlfeld, "The Gilbert-Elliott model for packet loss in real time services on the Internet", *Proceedings of the 14th GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems*, Dortmund, Germany, March 31 - April 2, 2008, pp. 269-286.

[9] N. Brandonisio, S. Porto, D. Carey, P. Ossieur, G. Talli, N. Parsons, and P. D. Townsend, "Burst-mode FPGA implementation and error location analysis of forward error correction for passive optical networks", *Journal of Optical Communications and Networking*, vol. 10, no. 4, 2018, pp. 298-308. DOI: 10.1364/JOCN.10.000298.

[10] K. Srinandhini and V. Vaithianathan, "FPGA implementation of MIMO-OFDM transceiver", *Proc. of the 2014 International Conference on Communication and Signal Processing*, Melmaruvathur, India, April 3-5, 2014, pp. 353-357. DOI: 10.1109/ICCSP.2014.6949861.

[11] R. Bassoli, V. N. Talooki, H. Marques, J. Rodriguez, R. Tafazolli, and S. Mumtaz, "Hybrid serial concatenated network codes for burst erasure channels", *Proc. of the 2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, Glasgow, UK, May 11-14, 2015, pp. 1-4. DOI: 10.1109/VTCSpring.2015.7145864.

[12] M. N. Krishnan and P. V. Kumar, "Rate-optimal streaming codes for channels with burst and isolated erasures", *Proc. of the 2018 IEEE International Symposium on Information Theory (ISIT)*, Vail, CO, USA, June 17-22, 2018, pp. 1809-1813. DOI: 10.1109/ISIT.2018.8437570.

[13] F. Casu, "Optimization of protection techniques based on FEC codes for the transmission of multimedia packetized streams", *Ph.D. Dissertation*, Universidad Politécnica de Madrid, Spain, 2017. Available at: http://oa.upm.es/46530/1/FILIPPO_CASU.pdf.

[14] M. Kabát, V. David, P. Holub, and M. Pulec, "High-performance forward error correction: enabling multi-gigabit flows and beyond on commodity GPU and CPU hardware in presence of packet loss", *Future Generation Computer Systems*, vol. 54, January 2016, pp. 326-335. DOI: 10.1016/j.future.2015.04.007.

[15] A. Wells, "Quasi-cyclic LDPC codes based on balanced incomplete block designs", *Ph.D. Dissertation*, Lancaster University, UK, 2011.

[16] X. Mountrouidou and H. G. Perros, "On the traffic modeling of burst aggregation algorithms using video and data traces", *Telecommunication Systems*, vol. 39, no. 3-4, December 2008, pp. 223-234. DOI: 10.1007/s11235-008-9127-8.

[17] B. Tiwari and R. Mehra, "Design and implementation of Reed-Solomon decoder for 802.16 network using FPGA", *Proc. of the 2012 IEEE International Conference on Signal Processing, Computing and Control*, Waknaghat Solan, India, March 15-17, 2012, pp. 1-5. DOI: 10.1109/ISPCC.2012.6224380.

[18] S. Bakale and D. Dabhade, "Reed-Solomon decoder with parallel syndrome computation on FPGA: a review", *International Journal of Science and Research (IJSR)*, vol. 4, no. 3, March 2015, pp. 1131-1134.

[19] C. R. Cullinan, T. R. Frattesi, and C. M. Wyant, "Computing performance benchmarks among CPU, GPU, and FPGA", *Computer Science*, 2012, pp. 1-124. Available at: https://web.wpi.edu/Pubs/E-project/Available/E-project-030212-123508/unrestricted/Benchmarking_Final.pdf.

[20] A. van der Ploeg, "Why use an FPGA instead of a CPU or GPU?", Netherlands eScience Center. Available at: https://blog.esciencecenter.nl/why-use-an-fpga-instead-of-a-cpu-or-gpu-b234cd4f309c.

[21] A. Boscolo, F. Vatta, F. Armani, E. Viviani, and D. Salvalaggio, "Physical AWGN channel emulator for bit error rate test of digital baseband communication", *Applied Mechanics and Materials*, vols. 241-

244, 2013, pp. 2491-2495. DOI: 10.4028/www.scientific.net/AMM.241-244.2491.

[22] M. Rigo, C. Travan, F. Vatta, and F. Babich, "Implementation of a burst error and burst erasure channel emulator using an FPGA architecture", *Proc. of the 2014 Int. Conf. on Software, Telecommunications and Computer Networks, SOFTCOM'14*, Split, Croatia, September 17-19, 2014, pp. 414-418. DOI: 10.1109/SOFTCOM.2014.7039095.

[23] P. K. Das and V. Tyagi, "Codes on s-periodic errors", *Ratio mathematica*, vol. 22, 2012, pp. 61-68. Available at: https://pdfs.semanticscholar.org/2140/c9b34056cb3f52a43af2808e0678086d5f18.pdf.

[24] J. Hamkins, "Optimal codes for the burst erasure channel", *Tech Briefs*, NASA's Jet Propulsion Laboratory, Pasadena, California, September 2010.

[25] Hamid Hemmati (ed.), *Deep-space optical communications*, John Wiley & Sons, New York, 2003.

[26] G. P. Calzolari, M. Chiani, F. Chiaraluce, R. Garello, and E. Paolini, "Channel coding for future space missions: new requirements and trends", *Proceedings of the IEEE*, vol. 95, November 2007, pp. 2157-2170. DOI: 10.1109/JPROC.2007.905134.

[27] J.-J. Climent, D. Napp, and V. Requena, "A novel construction for streaming networks with delays", *Proceedings of the 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on EUropean Transnational Education (ICEUTE 2019)*, Seville, Spain, May 13-15, 2019, pp. 185-194. DOI: 10.1007/978-3-030-20005-3_19.

[28] E. Gilbert, "Capacity of a burst-noise channel", *Bell System Technical Journal*, vol. 39, Sept. 1960, pp. 1253-1265. DOI: 10.1002/j.1538-7305.1960.tb03959.x.

[29] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels", *Bell System Technical Journal*, vol. 42, Sept. 1963, pp. 1977-1997. DOI: 10.1002/j.1538-7305.1963.tb00955.x.

[30] A. A. Markov, "Theory of algorithms", *Works of the Mathematical Institute, Academy of Sciences of the USSR*, vol. 42, 1954.

[31] B. Girod, K. Stuhlmüller, M. Link, and U. Horn, "Packet loss resilient internet video streaming", *Proceedings of SPIE Visual Communications and Image Processing*, San Jose, CA, United States, December 28, 1998, pp. 833-844. DOI: 10.1117/12.334735.

[32] J.-Y. Pyun, J.-J. Shim, S.-J. Ko, and S.-H. Park, "Packet loss resilience for video stream over the Internet", *IEEE Transactions on Consumer Electronics*, vol. 48, no. 3, Aug. 2002, pp. 556-563. DOI: 10.1109/TCE.2002.1037041.

[33] G. C. Clark, Jr. and J. B. Cain, *Error-correction coding for digital communications*, Plenum Press, New York, 1981. DOI: 10.1007/978-1-4899-2174-1_6.

[34] DVB, "Framing structure, channel coding and modulation for digital terrestrial television", *ETSI EN 300 744*, vol. 4.1, January 2001.

[35] S. H. Reiger, "Codes for the correction of 'clustered' errors", *IRE Transactions on Information Theory*, vol. 6, no. 1, March 1960, pp. 16-21. DOI: 10.1109/TIT.1960.1057558.

[36] M. Bakiri, C. Guyeux, J. Couchot, and A. Oudjida, "Survey on hardware implementation of random number generators on FPGA: theory and experimental analyses", *Computer Science Review*, Elsevier, vol. 27, February 2018, pp.135-153. DOI: 10.1016/j.cosrev.2018.01.002.

[37] R. C. Tausworthe, "Random numbers generated by linear recurrence modulo two", *Mathematics of Computation*, vol. 19, no. 90, 1965, pp. 201-209. DOI:10.1090/S0025-5718-1965-0184406-1.

[38] M. Comisso, F. Vatta, G. Buttazzoni, and F. Babich, "3D millimeter-wave peer-to-peer networks with boundary located destination", *IEEE Communications Letters*, vol. 23, no. 7, July 2019, pp. 1227-1230. DOI: 10.1109/LCOMM.2019.2914650.

[39] M. Comisso, G. Palese, F. Babich, F. Vatta, and G. Buttazzoni, "3D multi-beam and null synthesis by phase-only control for 5G antenna arrays", *Electronics (Switzerland)*, vol. 8, no. 6, June 2019, pp. 1-13. DOI: 10.3390/electronics8060656.

[40] M. Comisso, F. Vatta, G. Buttazzoni, and F. Babich, "Estimation of the bit error rate (BER) for uplink millimeter-wave line-of-sight communications", *Proc. of the 2019 International Conference on Software, Telecommunications and Computer Networks*, Split, Croatia, September 19-21, 2019, pp. 1-6. DOI: 10.23919/SOFTCOM.2019.8903692.

[41] A. Soranzo, F. Vatta, M. Comisso, G. Buttazzoni, and F. Babich, "New very simply explicitly invertible approximation of the Gaussian Q-function", *Proc. of the 2019 International Conference on Software, Telecommunications and Computer Networks*, Split, Croatia, September 19-21, 2019, pp. 1-5. DOI: 10.23919/SOFTCOM.2019.8903883.

[42] F. Babich, M. D'Orlando, and F. Vatta, "Distortion estimation algorithms for real-time video streaming: an application scenario", *Proc. of the 2011 International Conference on Software, Telecommunications and Computer Networks*, Split, Croatia, September 15-17, 2011, pp. 291-295.

[43] F. Babich, M. Noschese, A. Soranzo, and F. Vatta, "Low complexity rate compatible puncturing patterns design for LDPC codes", *Journal of Communication Software and Systems*, vol. 14, no. 4, December 2018, pp. 350-358. DOI:10.24138/jcomss.v14i4.639.

[44] F. Vatta, A. Soranzo, M. Comisso, G. Buttazzoni, and F. Babich, "New explicitly invertible approximation of the function involved in LDPC codes density evolution analysis using a Gaussian approximation", *Electronics Letters*, vol. 55, no. 22, October 2019, pp. 1183-1186. DOI: 10.1049/el.2019.1349.

[45] F. Vatta, F. Babich, F. Ellero, M. Noschese, G. Buttazzoni, and M. Comisso, "Performance study of a class of irregular LDPC codes based on their weight distribution analysis", *Proc. of the 2019 International Conference on Software, Telecommunications and Computer Networks*, Split, Croatia, September 19-21, 2019, pp. 1-6. DOI: 10.23919/SOFTCOM.2019.8903633.

[46] F. Vatta, A. Soranzo, M. Comisso, G. Buttazzoni, and F. Babich, "Performance study of a class of irregular LDPC codes through low complexity bounds on their belief-propagation decoding thresholds", *Proc. of the 111th AEIT International Annual Conference, AEIT 2019*, Florence, Italy, September 18-20, 2019, pp. 1-6. DOI: 10.23919/AEIT.2019.8893306.

[47] F. Vatta, A. Soranzo, and F. Babich, "More accurate analysis of sum-product decoding of LDPC codes using a Gaussian approximation", *IEEE Communications Letters*, vol. 23, no. 2, February 2019, pp. 230-233. DOI: 10.1109/LCOMM.2018.2886261.

[48] F. Vatta, F. Babich, F. Ellero, M. Noschese, G. Buttazzoni, and M. Comisso, "Role of the product $\lambda'(0)\rho'(1)$ in determining ldpc code performance", *Electronics (Switzerland)*, vol. 8, no. 12, December 2019, pp. 1-14. DOI: 10.3390/electronics8121515.

[49] A. Brown, M. Luby, and A. Shokrollahi, "Repeat-accumulate codes that approach the Gilbert-Varshamov bound", *Proc. of the IEEE International Symposium on Information Theory*, Adelaide, Australia, September 2005, pp. 169-173. DOI: 10.1109/ISIT.2005.1523316.

[50] F. Fagnani and C. Ravazzi, "Spectra and minimum distances of repeat multiple accumulate codes", *IEEE Transactions on Information Theory*, vol. 55, no. 11, November 2009, pp. 4905-4924. DOI: 10.1109/TIT.2009.2030459.

[51] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo codes", *Proc. of the 1993 IEEE International Conference on Communications - ICC'93*, Geneva, Switzerland, May 1993, pp. 1064-1070. DOI: 10.1109/ICC.1993.397441.

[52] F. Babich and F. Vatta, "Turbo codes construction for robust hybrid multitransmission schemes", *Journal of Communication Software and Systems*, vol. 7, no. 4, December 2011, pp. 128-135. DOI: 10.24138/jcomss.v7i4.174.

[53] F. Vatta, R. Romano, and M. T. D. Alizo, "Turbo codes for quantum key distribution (QKD) applications", *Proc. of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies, ISABEL'11*, Barcelona, Spain, October 26-29, 2011, pp. 1-5. DOI: 10.1145/2093698.2093860.

[54] F. Babich and F. Vatta, "On the error statistics of turbo decoding for hybrid concatenated codes design", *Journal of Communication Software and Systems*, vol. 15, no. 2, June 2019, pp. 202-212. 10.24138/jcomss.v15i2.669.

[55] D. Divsalar and F. Pollara, "Serial and hybrid concatenated codes with applications", *Proc. of the International Symposium on Turbo Codes and Related Topics*, Brest, France, Sept. 1997, pp. 80-87. DOI: 10.1.1.81.5866.

[56] C. Koller, A. Graell i Amat, J. Kliewer, F. Vatta, K. Sh. Zigangirov, and D. J. Costello, Jr., "Analysis and design of tuned turbo codes", *IEEE Transactions on Information Theory*, vol. 58, no. 7, July 2012, pp. 4796-4813. DOI:10.1109/TIT.2012.2195711.

[57] D. J. Costello and G. Meyerhans, "Concatenated turbo codes", *Proc. of the 1996 International Symposium on Information Theory and its Applications*, Victoria B. C., Canada, Sept. 1996.

[58] V. Sidorenko, M. Bossert, and F. Vatta, "Properties and encoding aspects of direct product convolutional codes", *Proc. of the 2012 IEEE International Symposium on Information Theory, ISIT'12*, Boston, USA, July 1-6, 2012, pp. 2351-2355. DOI: 10.1109/ISIT.2012.6283934.

[59] F. Vatta, V. Sidorenko, and M. Bossert, "Termination and tailbiting of rate-k/n direct product convolutional codes", *Proc. of the 2010 IEEE International Symposium on Information Theory, ISIT'10*, Austin, USA, June 13-18, 2010, pp. 2023-2027. DOI: 10.1109/ISIT.2010.5513371.

[60] B. Liu, Y. Li, B. Rong, L. Gui, and Y. Wu, "LDPC-RS product codes for digital terrestrial broadcasting transmission system", *IEEE Transactions on Broadcasting*, vol. 60, no. 1, March 2014, pp. 38-49. DOI: 10.1109/TBC.2013.2291359.

Caterina Travan received her M.Sc. in Electronics and Telecommunications Engineering from University of Trieste, Trieste, Italy, in 2015. She worked then on her Ph.D. Degree with the Technical University of Graz and Infineon Technologies AG, Austria, until 2018, with main focus on novel materials for gas sensing. She is currently employed at Infineon Technologies AG, Munich, Germany, as development engineer for environmental sensors.

Francesca Vatta received the M.Sc. Degree in Electronic Engineering and the Ph.D. Degree in Telecommunications from the University of Trieste, Trieste, Italy, in 1992 and 1998, respectively. She joined the Department of Engineering and Architecture (DIA) of the University of Trieste in 1999, where she is Assistant Professor of Information Theory and Error-Control Coding. Starting in 2002, she spent several months as visiting scholar at the University of Notre Dame, Notre Dame, IN, U.S.A., cooperating with the Coding Theory Research Group under the guidance of Prof. D. J. Costello, Jr. Starting in 2005, she spent several months as visiting scholar at the University of Ulm, Germany, cooperating with the Telecommunications and Applied Information Theory Research Group under the guidance of Prof. M. Bossert. She is an author of more than 80 papers published on international journals and conference proceedings. Her current research interests are in the area of channel coding concerning, in particular, the analysis and design of capacity achieving coding schemes.

Fulvio Babich received the doctoral degree, (Laurea), cum laude, in Electrical Engineering, at the University of Trieste, on July 1984. After graduation he was with Telettra at the Research and Development Laboratories, where he was engaged in optical fiber communications. Then he joined Zeltron, where he was a communication system engineer, responsible of the activities within the ESPRIT program. In 1992 he joined the Department of Electrical Engineering (DEEI) of the University of Trieste, where he is Professor of Signals and Systems and Wireless Networks. Fulvio Babich is vice director of Department of Engineering and Architecture. He is the coordinator of the Ph.D. in Industrial and Information Engineering of the University of Trieste. His current research interests are in the field of wireless networks and personal communications. He is involved in channel modeling, multiple access techniques, channel encoding, error control techniques and cross-layer design. Fulvio Babich serves as reviewer for many international journals, has served as co-chair for the Communication Theory Symposium, ICC 2005, Seul, ICC 2014, Sydney, and ICC 2017, Paris, for the Wireless Communication Symposium, ICC 2011, Kyoto, and for the Wireless Communication Symposium, WCSP 2012, Huangshan, China. He has served as General chair of the 13th IEEE IFIP Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2014 June 2-4, 2014 Piran, Slovenia. Fulvio Babich is Senior Member of IEEE.