IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 September 2023, accepted 23 October 2023, date of publication 26 October 2023, date of current version 2 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3327786

RESEARCH ARTICLE

Design and Implementation of a DVB-S2 Reconfigurable Datapath BCH Encoder for High Data-Rate Payload Data Telemetry

GIOVANNI QUINTARELLI[1], MATTEO BERTOLUCCI[2], AND PIETRO NANNIPIERI[1], (Member, IEEE)
[1]Department of Information Engineering, University of Pisa, 56122 Pisa, Italy
[2]IngeniArs S.r.l., 56121 Pisa, Italy

Corresponding author: Pietro Nannipieri (pietro.nannipieri@unipi.it)

This work was supported in part by the Italian Ministry of Education and Research (MIUR) through the FoReLab Project (Departments of Excellence).

ABSTRACT To ensure the flexibility of Earth Observation satellite missions, it is essential to have highly adaptable communication systems equipped with efficient modulation and coding schemes. The preferred approach for such applications is the DVB-S2 standard, which provides three operational modes: Constant Coding and Modulation, Variable Coding and Modulation, and Adaptive Coding and Modulation. Additionally, this standard incorporates a robust Forward Error Correction system that combines a Low-Density Parity-Check encoder with a Bose-Chaudhuri-Hocquenghem encoder. This combination ensures optimal utilization of channel bandwidth while maintaining an acceptable Bit Error Rate throughout the satellite's orbit. This research is centred on designing and implementing a highly flexible and high-speed parallel Bose-Chaudhuri-Hocquenghem encoder, fully compliant with the DVB-S2 standard, and supporting all Constant Coding and Modulation, Variable Coding and Modulation, and Adaptive Coding and Modulation modes. In contrast to existing solutions, our proposed encoder permits the concurrent processing of a larger number of bits, thereby enhancing parallelism. Furthermore, parallelism is a configurable parameter, allowing seamless system scalability. The proposed encoder can be tailored to accommodate a wide range of throughput requirements, making it suitable for various mission classes. This flexibility enables users to balance factors like complexity, power consumption, and performance. The hardware platform selected for circuit implementation is the space-grade Xilinx XQRKU060 FPGA. Our results demonstrate a maximum achievable throughput ranging from 600 Mbps to 19.2 Gbps and a dynamic power consumption of 10 mW to 79 mW, depending on the chosen parallelism level (ranging from 2 to 96). The optimization of resource utilization is particularly noteworthy, as it reduces the required resources to as low as 760 Look-up Tables and 344 registers for lower parallelism levels.

INDEX TERMS DVB-S2, BCH, satellite, telemetry transmitter, high throughput, FPGA, parallel processing, PDT, reconfigurable datapath.

I. INTRODUCTION

The Digital Video Broadcasting Satellite 2nd generation (DVB-S2) [1] standard is a satellite communication standard proposed by the European Telecommunications Standards Institute (ETSI) in 2005 as the evolution of the DVB-S standard. Later, it was proposed as a recommended standard for high data-rate Payload Data Telemetry (PDT) downlink by the Consultative Committee for Space Data System (CCSDS) organization. DVB-S2 defines the transmission scheme shown in Figure 1. It provides a powerful Forward Error Correction (FEC) system that combines a Low-Density

The associate editor coordinating the review of this manuscript and approving it for publication was Gian Domenico Licciardo.

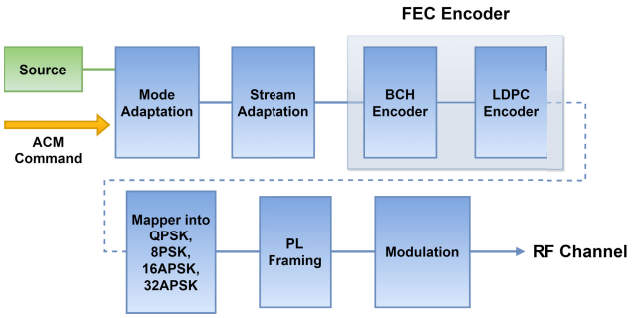

FIGURE 1. Function block diagram of a DVB-S2 transmitter.

Parity-Check (LDPC) encoder [2] with a Bose-Chaudhuri-Hocquenghem (BCH) encoder that enables near-error-free operation, achieving performance levels within 0.7 to 1 dB from the Shannon limit. Additionally, it offers a wide range of possible code rates and modulations and three different modes of operations, the Constant Coding and Modulation (CCM), Variable Coding and Modulation (VCM), and the Adaptive Coding and Modulation (ACM) mode [3]. The last two modes in particular, VCM and ACM, allow the optimization of satellite-to-ground station communication despite to variable link budget due to the satellite's orbit [4].

According to the anticipated relative position between the satellite and the ground station, the VCM mode consists of adjusting both the modulation and coding rate in response to the distance from it. As a result, in scenarios where the distance is shorter so the link budget is more advantageous, a higher code rate and more spectrally efficient modulation will be employed. Conversely, in cases where the distance is longer so the link budget is less favourable, a less spectrally efficient modulation and lower code rate will be employed. While forecasting the distance is generally straightforward, some tropospheric propagation phenomena can be unpredictable and might exert a substantial influence on the link budget, particularly at higher frequencies. To confront this challenge, one potential solution is to employ the ACM mode. The ACM mode entails dynamically adjusting the modulation and coding rate in real-time to ensure peak performance, guided by the receiver's measurement of the signal's Carrier-to-Noise Ratio (C/N) ratio, as outlined in [5]. As shown in Figure 2, an additional communication uplink between the ground station and the satellite is required to send the command regarding the code rate and modulation to adopt.

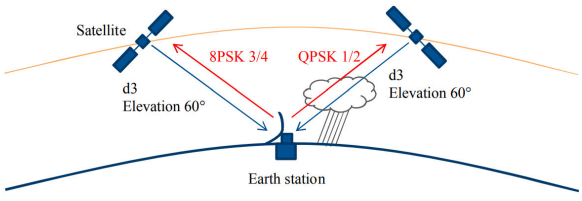

FIGURE 2. Adaptive coding and modulation (ACM) working principle.

Presently, the existing BCH encoders employed in communication systems demonstrate commendable performance regarding both throughput and error correction capabilities. Nevertheless, they grapple with constraints in terms of adaptability and re-programmability across various operational modes. This deficiency in flexibility not only escalates design costs but also impedes the widespread acceptance of this technology, particularly within the burgeoning new space sector [6].

To surmount these limitations and bolster the adaptability of error correction, further investigation is necessary to explore design strategies that can deliver enhanced flexibility while preserving high-performance standards. Economical solutions for data payload satellite transmitters entail the utilization of reconfigurable systems constructed on FPGA devices. Such reconfigurability can be seen at different levels:

- Full support to all the operatives modes foreseen in the DVB-S2 standard thanks to ACM and VCM, so that the satellites can always operate at the maximum available bandwidth.
- If the encoder is developed with configurable parallelism, the same Intellectual Property (IP) core can be configured to operate in a mission with drastically different requirements in terms of complexity, power budget and data-rate requirements, without incurring in additional non-recurring design costs. Such an aspect is of key importance in the new space scenario, where affordable access to space technology can be granted by designing components with broader applicability in terms of mission classes.
- Being Field Programmable Gate Array (FPGA) the technology target, also in-mission re-purposing of the transmitter system can be supported, by updating the FPGA programming file. This may be useful because the satellites experience different communication requirements in the various phases of their operative lifespan, allowing them to save power or logical resources in the phases of the satellite mission that require lower downlink bandwidth.

The design of reconfigurable systems requires the architectures of each block to be highly parametric to optimize the various metrics, such as power, logic resources and performance according to the customer's needs. The presented work focuses on the design and implementation of a full DVB-S2-compliant parallel BCH encoder, which will be integrated into a DVB-S2 compliant transmitter IP core. The key contributions of the proposed innovative BCH encoder are:

- Full support to all CCM, VCM and ACM modes.
- Reconfigurable datapath, which allows the selection of the desired parallelism. Based on the selected parallelism, the proposed system configurability implies variable throughput, clock frequency, power and resource utilization.
- Such configurability provides increased flexibility to potential users, compared to state-of-the-art examples of

full DVB-S2 compliant encoders supporting all CCM, VCM and ACM modes (i.e. [7], [8], and [9]), remaining competitive in terms of performances.

The rest of this paper is organized as follows. Section II will focus on the theory of BCH codes and shows both serial and parallel encoding algorithms and some state-of-the-art implementations of BCH encoders. Section III makes a detailed description of the proposed BCH encoder architecture. Implementation results are presented and analyzed in Section IV, and the conclusions are finally drawn in Section V.

II. BCH ENCODERS

BCH codes form a large class of cyclic codes that are generated using polynomials over a finite field (also called a Galois Field) and are highly effective in correcting random errors. These codes were first introduced as binary BCH codes by Hocquenghem in 1959 [10], and then separately discovered by Bose and Chaudhuri in 1960 [11]. Their cyclic property was demonstrated by Peterson in 1960 [12]. The primary class of BCH codes comprises binary BCH codes.

For any integer

$$m \geqslant 3 \quad and \quad t < 2^{m-1} \tag{1}$$

is defined a binary BCH code with the following properties: [13]

- **Block length:**

$$n = 2^m - 1 \tag{2}$$

- **Number of parity bits:**

$$n - k \leqslant mt \tag{3}$$

- **Minimum Hamming** distance:

$$d_{min} \geqslant 2t + 1 \tag{4}$$

Each information block and codeword can be represented with a polynomial over $GF(2)$ where bits are mapped into the polynomial's coefficients. For example, let's suppose we have a message block to encode described as a k-tuple with the form

$$(a_0, a_1, \cdots, a_{k-1}) \tag{5}$$

It can be described with the polynomial

$$u(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \tag{6}$$

By using this notation, a polynomial code, and therefore also a BCH code, can be defined as a set of polynomials of degree $n-1$ or lower that includes a particular polynomial $g(x)$ as one of its factors. This polynomial $g(x)$ is referred to as the code's *generator polynomial* and for a (n, k) code it is of degree $n-k$. So, all the codeword's polynomials $c(x)$ will be described by (7).

$$c(x) = g(x) \cdot u(x) \tag{7}$$

The DVB-S2 standard specifies the use of different BCH codes for each operating mode and defines the generator polynomial $g(x)$ for each of them.

A. SERIAL BCH ENCODING

Applying (7) for the encoding process will result in a non-systematic codeword, where checkbits are not appended after the information bits. Conversely, the encoding algorithm for a systematic (n, k) BCH code is given by the following steps:

1) Multiply the message polynomial shown in (6) by x^{n-k}. It will result in a right shift of the information bits by $n-k$ positions.

$$u'(x) = u(x) \cdot x^{n-k} \tag{8}$$

2) Divide $u'(x)$ by $g(x)$ and take the reminder $r(x)$. The coefficients of $r(x)$ represent the checkbits.

$$r(x) = r_0 + r_1 x + \cdots + r_{n-k-1} x^{n-k-1} \tag{9}$$

3) Add $r(x)$ to $u'(x)$ to obtain the codeword $c(x)$

$$c(x) = r(x) + u(x) \cdot x^{n-k} \tag{10}$$

This algorithm can be implemented by using the serial Linear Feedback Shift Register (LFSR) architecture shown in Figure 3.

The sum and product operations on the message bits are performed over the Galois Field $GF(2)$, so they are modulo-2. Consequently, the adder and multiplier blocks become XOR and AND logical operations, respectively.

For the first k clock cycles, the message bits enter into the LFSR starting for the MSB. At the same time, input data are sent to the output port to form the systematic part of the codeword. After k cycles the checkbits $r(x)$ are ready to be read, so they have shifted out from LFSR registers and sent to the output port.

Since one bit is processed every cycle, the throughput will be equal to the clock frequency and cannot be varied once it is fixed. Therefore, it is necessary to find another encoding algorithm in which data throughput also depends on another parameter.

B. PARALLEL BCH ENCODING

To increase the throughput for a fixed clock frequency, more than one bit per clock cycle must be processed. As a result, the throughput would be expressed as

$$throughput = parallelism \cdot f_{clk} \tag{11}$$

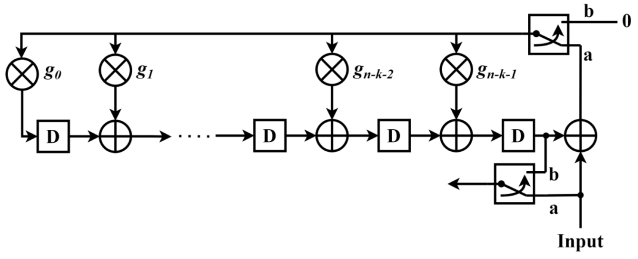

FIGURE 3. LFSR serial architecture.

where parallelism refers to the number of bits that are processed each cycle. This approach has been followed in [14], which proposes a parallel BCH encoding algorithm.

For the LFSR shown in Figure 3, assuming that

$$r = n - k \tag{12}$$

is the number of checkbits. Assuming that the r-tuple $(p_0(T), \cdots, p_{r-1}(T))$ represents the state of all the remainder registers at the time T and $x(T+1)$ represents the input bit at the time $T+1$. We can express the $p_i(T+1)$ state of the i-th register as

$$p_i(T+1) = p_{i-1}(T) + g_{i-1}[p_{r-1}(T) + x(T+1)] \tag{13}$$

So the $\mathbf{p}(T+1)$ vector can be derived

$$\begin{pmatrix} p_{r-1}(T+1) \\ p_{r-2}(T+1) \\ p_{r-3}(T+1) \\ \vdots \\ p_1(T+1) \\ p_0(T+1) \end{pmatrix}_r = \begin{pmatrix} g_{r-1} & 1\,0\,0\cdots 0 \\ g_{r-2} & 0\,1\,0\cdots 0 \\ g_{r-3} & 0\,0\,1\cdots 0 \\ \vdots & \\ g_1 & 0\,0\,0\cdots 1 \\ g_0 & 0\,0\,0\cdots 0 \end{pmatrix}_{r\times r} \begin{pmatrix} p_{r-1}(T) + x(T+1) \\ p_{r-2}(T) \\ p_{r-3}(T) \\ \vdots \\ p_1(T) \\ p_0(T) \end{pmatrix}_r \tag{14}$$

that leads to the equation (15)

$$\mathbf{p}(T+1) = \mathbf{Q}[\mathbf{p}(T) + x(T+1)] \tag{15}$$

where the $\mathbf{Q}$ matrix is called the *state matrix*.

Equation (15) suggests that it is possible to predict the future state of the remainder registers by knowing both the current state and the future input bits. By assuming that $\mathbf{p}(T+m)$ represents the state of the remainder registers at a certain time $T+m$, then based on (15), it is possible to derive a formula that calculates the value of $\mathbf{p}(T+m)$.

$$\mathbf{p}(T+m) = \mathbf{Q}^m[\mathbf{p}(T) + \mathbf{x}_m] \tag{16}$$

where

$$\mathbf{x}_m = \begin{pmatrix} x(T+1) \\ x(T+2) \\ \vdots \\ x(T+m) \\ 0 \\ \vdots \\ 0 \end{pmatrix}_r \tag{17}$$

The value of $\mathbf{Q}^m$ can be found recursively by varying the index i from 2 to m in

$$\mathbf{Q}^i = \left(\mathbf{Q}^{i-1} \begin{pmatrix} g_{r-1} \\ \vdots \\ g_1 \\ g_0 \end{pmatrix} \middle| \text{The first } r-1 \text{ columns of } \mathbf{Q}^{i-1} \right)_{r\times r} \tag{18}$$

Finally, (16) translates into an algorithm that recursively calculates the codeword by processing m bits at a time. By implementing this algorithm in hardware it is possible to increase the output data throughput of the BCH encoder by a m factor over the standard serial solution.

C. STATE OF THE ART

BCH codes are widely used in various fields given their effective random error correction capability. In the context of FEC systems and transmitters, as discussed in the paper, BCH codes are combined with LDPC codes for PDT applications that adopt the DVB-S2 standard.

The DVB-S2 CCSDS telemetry standard has achieved significant success within the space community, leading to its adoption for specific missions [15]. For this kind of application, BCH encoders should support variable code rates and error correction capability to implement ACM and VCM modes.

Previous research works, notably [7] and [16], elucidate the architecture of a DVB-S2 FEC system, comprising a BCH encoder and a LDPC encoder, a Bit Interleaver, and a Constellation Mapper. While [7] implements both VCM and CCM modes and employs a parallel BCH encoder, [16] exclusively supports CCM mode and adopts a serial BCH encoder with a conventional LFSR architecture.

Another instance of a DVB-S2 FEC system is exemplified in [8], featuring an 8-bit parallel BCH and LDPC encoder capable of supporting every Modulation and Code Rate (ModCod) configuration defined by the standard.

Furthermore, a novel system architecture for a DVB-S2 FECencoding system is outlined in [17], suggesting the deployment of a concatenated LDPC encoder with parallel branches of BCH encoders to enhance throughput compared to the conventional serial concatenation approach.

In addition to their prominence in satellite communications, BCH codes find application in Solid State Drive (SSD) controllers, ensuring reliable reading and writing of NAND flash memory. In [18], the authors propose a systolic array-type design for the parallel BCH encoder and syndrome generator at the decoder stage, allowing for flexibility in any parallelization factor without adding complexity.

Within the domain of digital communication and storage, where BCH encoders present extensive usage, [19] introduces an alternative parallel BCH encoder architecture. Based on vector calculation principles rather than the more conventional LFSR implementation, this approach yields a reduction in the utilization of logical resources, all the while preserving

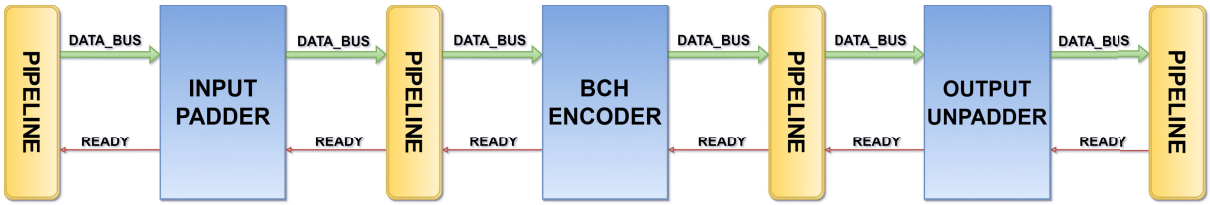

FIGURE 4. IP architecture block scheme.

TABLE 1. State of the art summary on architecture.

Reference	DVB-S2 Compliant	BCH Encoder Architecture	Throughput	Parallelism
[7]	Yes	Parallel	1.19 Gbps	Fixed
[8]	Yes	Parallel	N/A	Fixed
[16]	Only CCM	Serial	122 Mbps	No
[18]	No	Parallel	5.1 Gbps	Reconfigurable
[14]	No	Parallel	999.2 Mbps	Fixed

the throughput capacity. Additionally, a different approach employing vector calculation for parallel BCH encoders is elucidated in [14].

Table 1 provides a summary of the key points and outcomes of each study. Although these advanced encoding techniques offer many benefits, there is a noticeable limitation in the current state of the art. All the architectures compliant with DVB-S2 are not reconfigurable by the user. So, the design metrics are predetermined and cannot be altered, making these types of architectures suitable only for the specific mission requirements for which they are designed. To support different and variable mission requirements, reconfigurable architectures are necessary. Therefore, we propose a new parallel BCH encoder architecture that offers full compliance with the DVB-S2 standard and parallelism reconfigurability.

III. PROPOSED ARCHITECTURE

A pipeline architecture with an AXI-Stream-like interface is proposed. It provides signal synchronization between the stages and allows to increase in the maximum clock frequency. The main innovative aspects of this architecture are:

- Configurable datapath, allowing the use of the same IP core in applications with very different requirements in terms of throughput, power and complexity;
- State-of-the-art performances in terms of throughput;
- Full compliance to the DVB-S2 Standard.

The proposed architecture is illustrated in Figure 4 and will be analyzed in the following three sections.

A. INPUT PADDER

The DVB-S2 standard proposes the BCH encoder as the first stage of the FEC encoding system. Here, the data packet, or BBFRAME, has to be encoded with the expected

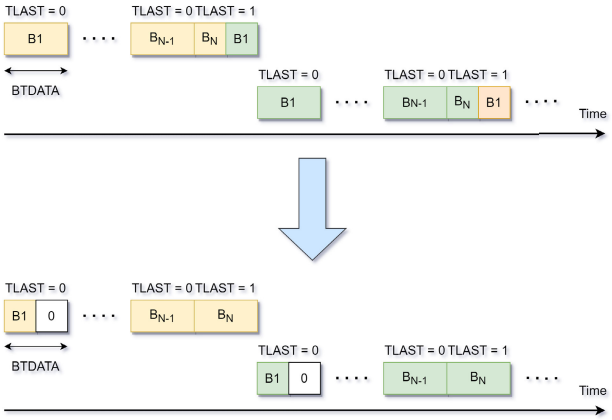

FIGURE 5. Input padding process.

(n_{BCH}, k_{BCH}) BCH code, where k_{BCH} is the uncoded word length while n_{BCH} is the coded word length. The parallel encoder has a generic m-bit parallelism, that is referred to the number of bits that are processed simultaneously, so the data packet will have to be divided into sub-blocks of size m. For a generic set of different parallelisms, it would happen that k_{BCH} is not a multiple of m. As a result, the final block of m data within the packet will contain data from two contiguous packets. This invalidates the last data block, causing a loss of information. To ensure that each data block exclusively belongs to a single packet, the packet length must be made divisible by m. This is achieved by adding padding zeros at the beginning of the packet, as illustrated in Figure 5

Note that the input interface requires the first bits to be processed to be in LSB positions, so padding zeros are added to the LSB positions of the first data block.

B. BCH ENCODER

The DVB-S2 standard defines variable error correction capabilities

$$t = \{8, 10, 12\} \tag{19}$$

for the BCH codes that translate into different generator polynomials $g(x)$ with different degrees.

It provides two useful tables for the generator polynomial's definition, one for short FECFRAME length (16200 bit) and the other for long FECFRAME length (64800 bit).

It also provides the method for the calculation of the generator polynomial given t and the FECFRAME length. The first t polynomials in Table 2 or Table 3 have to be multiplied.

Table 4 report the t associated value to every defined LDPC code rate and FECFRAME length. It can be observed that for each FECFRAME length different LDPC code rates share the same t value, so they will have the same generator polynomial. In conclusion, we will have only four different generator polynomials for all the possible combinations of FECFRAME length and LDPC code rate, whose relative degrees are reported in Table 4.

ACM and VCM modes allow dynamic variations in FECFRAME length, modulation and coding rate, so this type of information must be transmitted throughout the processing pipeline. The PLSCODE is an 8-bit word that defines the ModCod, FECFRAME length and pilot insertion [1].

So, the proposed BCH encoder architecture should handle the dynamic change of PLSCODE for a generic set of different parallelism levels. Equation (16) defines the iterative algorithm for parallel BCH encoding, but the state matrix and the number of checkbits are fixed. Dynamic variations in PLSCODE, on the other hand, require variable state matrices and several checkbits. It is accomplished by the proposed architecture in Figure 6.

The state register holds the current state $\mathbf{p}(T)$ in (16), so the register input and output data width must be equal to the number of checkbits to compute. The number of checkbits has been fixed to the maximum possible one, 192. If fewer checkbits are required, padding zeroes are added to reach the length of 192. The next step, described by (16) is to add the input bits to the first m bit of the current state. The modulo-2 addition is performed with XOR gates, represented in Figure 6 with a two-input bus XOR. Each bus width is BTDATA which is equal to the parallelism m. The computed BTADTA bits are now concatenated with the other $192 - BTDATA$ bits of the present state, to form the $\mathbf{p}(T) + x_m$ term defined in (16). Now to calculate the future state $\mathbf{p}(T + m)$ the vector-matrix product with the state matrix $\mathbf{Q}^m$ is performed by the combinational network in Figure 6. As previously mentioned different PLSCODEs should be handled so different state matrices have to be defined, once for every possible generator polynomial and parallelism. Since parallelism is a parameter that is fixed at compile time, the only other variable that affects the state

TABLE 2. DVB-S2 minimal polynomials for 16200-bit FECFRAME length.

$g_1(x)$	$1 + x + x^3 + x^5 + x^{14}$
$g_2(x)$	$1 + x^6 + x^8 + x^{11} + x^{14}$
$g_3(x)$	$1 + x + x^2 + x^6 + x^9 + x^{10} + x^{14}$
$g_4(x)$	$1 + x^4 + x^7 + x^8 + x^{10} + x^{12} + x^{14}$
$g_5(x)$	$1 + x^2 + x^4 + x^6 + x^8 + x^9 + x^{11} + x^{13} + x^{14}$
$g_6(x)$	$1 + x^3 + x^7 + x^8 + x^9 + x^{13} + x^{14}$
$g_7(x)$	$1 + x^2 + x^5 + x^6 + x^7 + x^{10} + x^{11} + x^{13} + x^{14}$
$g_8(x)$	$1 + x^5 + x^8 + x^9 + x^{10} + x^{11} + x^{14}$
$g_9(x)$	$1 + x + x^2 + x^3 + x^9 + x^{10} + x^{14}$
$g_{10}(x)$	$1 + x^3 + x^6 + x^9 + x^{11} + x^{12} + x^{14}$
$g_{11}(x)$	$1 + x^4 + x^{11} + x^{12} + x^{14}$
$g_{12}(x)$	$1 + x + x^2 + x^3 + x^5 + x^6 + x^7 + x^8 + x^{10} + x^{13} + x^{14}$

TABLE 3. DVB-S2 minimal polynomials for 64800-bit FECFRAME length.

$g_1(x)$	$1 + x^2 + x^3 + x^5 + x^{16}$
$g_2(x)$	$1 + x + x^4 + x^5 + x^6 + x^8 + x^{16}$
$g_3(x)$	$1 + x^2 + x^3 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{16}$
$g_4(x)$	$1 + x^2 + x^4 + x^6 + x^9 + x^{11} + x^{12} + x^{14} + x^{16}$
$g_5(x)$	$1 + x + x^2 + x^3 + x^5 + x^8 + x^9 + x^{10} + x^{11} + x^{12} + x^{16}$
$g_6(x)$	$1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{12} + x^{13} + x^{14} + x^{15} + x^{16}$
$g_7(x)$	$1 + x^2 + x^5 + x^6 + x^8 + x^9 + x^{10} + x11 + x^{13} + x^{15} + x^{16}$
$g_8(x)$	$1 + x + x^2 + x^5 + x^6 + x^8 + x^9 + x^{12} + x^{13} + x^{14} + x^{16}$
$g_9(x)$	$1 + x^5 + x^7 + x^9 + x^{10} + x^{11} + x^{16}$
$g_{10}(x)$	$1 + x + x^2 + x^5 + x^7 + x^8 + x^{10} + x^{12} + x^{13} + x^{14} + x^{16}$
$g_{11}(x)$	$1 + x^2 + x^3 + x^5 + x^9 + x^{11} + x^{12} + x^{13} + x^{16}$
$g_{12}(x)$	$1 + x + x^5 + x^6 + x^7 + x^9 + x^{11} + x^{12} + x^{16}$

TABLE 4. DVB-S2 error correction capability and generator polynomials for BCH codes.

FECFRAME length	LDPC code rate	t	Polynomial degree
normal (64800 bit)	1/4, 1/3, 2/5, 1/2, 3/5, 3/4, 4/5, 5/6	12	192
	2/3, 5/6	10	160
	8/9, 9/10	8	128
short (16200 bit)	1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9	12	168

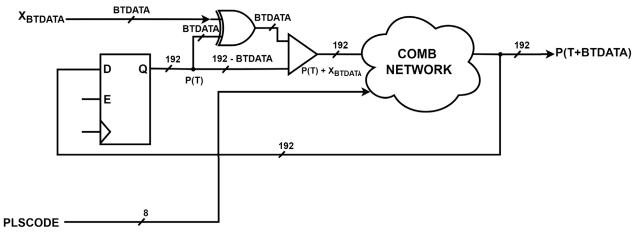

FIGURE 6. Parallel BCH encoder for multiple PLSCODEs.

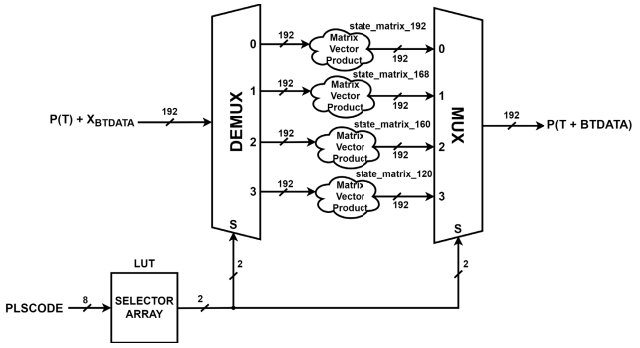

FIGURE 7. Combinational network for multiple PLSCODEs.

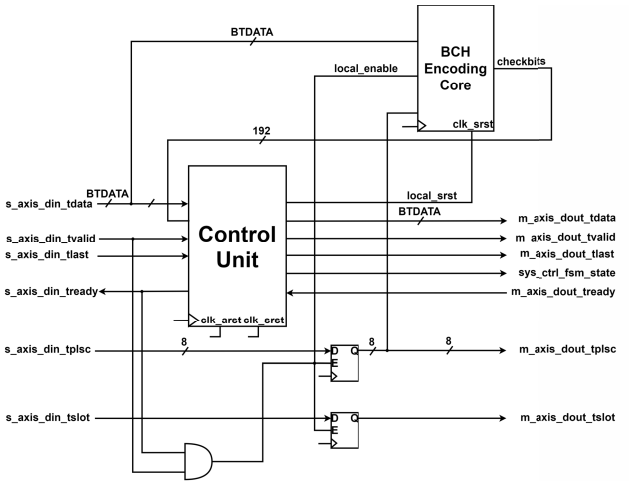

FIGURE 8. BCH encoder block architecture.

matrix is the generator polynomial. Fortunately, the presence of a limited number of possible generator polynomials limits the possible state matrices' number to four and allows for a straightforward implementation of the combinational network, that assumes the form shown in Figure 7.

Four combinational sub-network are defined, that perform a single matrix-vector product operation with a fixed matrix as a parameter. Each matrix is precalculated in MATLAB as the associated state matrix for each of the four possible generator polynomials and it is instantiated at compile time.

Hence, the input $\mathbf{p}(T) + x_m$ is routed in one of the four possible paths by a demultiplexer and the output $\mathbf{p}(T + m)$ is routed to the output of the combinational network by a multiplexer. The path is selected by a decoder depending on the current PLSCODE.

The full BCH encoder block architecture is shown in Figure 8. For the first k_{BCH}/m cycles the input message bits are sent both to the BCH encoding core and to the output by the control unit. Once the checkbits are ready, the control unit switches them to the output port for the remaining cycles. After n_{BCH}/m total cycles, the systematic codeword has been produced. Another task handled by the control unit is the pipeline control signals driving.

C. OUTPUT UNPADDER

This is the last block in the pipeline that will interface with the LDPC encoder. Consequently, it will have to adapt the format of the data output from the BCH encoder to one suitable for processing by the LDPC encoder. Since the processing is LSB-first, the first bits that will be processed are those in LSB positions and should be information bits. So, the output unpadder block should take the zeros that were initially added by the input padder at the beginning of the frame and move them to the end.

A simple implementation of this process consists of sending the input data blocks into a shift register and selecting the output data blocks with a floating window, implemented with a multiplexer. Initially, after a few cycles, the shift register will be full, so the first output data block will be selected by removing the padding zeros. Then, when the last data block enters the shift register, several zeros equal to those initially removed will be loaded inside the shift register and will be sent to the output attached to the last data block.

The final result is that, after an initial latency time due to the shift register filling, the padding zeros at LSB positions will be moved to MSB positions as shown in Figure 9.

IV. RESULTS

A. DESIGN VERIFICATION METHODOLOGY

The design verification methodology consists of three main phases. At first, a high-level reference model is created to validate the correct functionality of the circuit. This reference model makes it easier to generate the expected results because fast and powerful scripting languages like Python or MATLAB can be used to model the system. The second step regards the Register Transfer Level (RTL) functional simulation. To prove its correctness, a last step is necessary: a comparative analysis between the reference model and the RTL model's outputs.

The verification process has been automated through the use of bash scripts, and it has been iterated to ensure that every supported level of parallelism {2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64, 96} in the architecture is thoroughly tested, along with every PLSCODE defined by the standard.

The BCH encoder has been modelled in MATLAB as a class object. It takes as input the parallelism, the PLSCODE and a vector of bits that represents the BBFRAME to encode. The BBFRAME is created using MATLAB's random function, which generates a vector of random 0's and 1's of the desired length. The length of the vector is determined by the BBFRAME length specified by the selected PLSCODE. To produce the reference output data the parallelism is fixed and random BBFRAMEs are produced. Then they are computed by the encoder, one for every defined PLSCODE.

The input BBFRAMEs and the output codewords are sliced into groups of m bits, where m is the parallelism and are written inside two different text files. Similarly, the

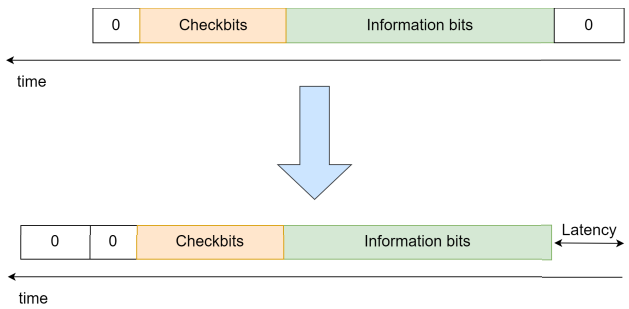

FIGURE 9. Output unpadding process.

TABLE 5. IP design metrics values after implementation.

Parallelism	F_{MAX}	Throughput	LUTs	FFs	Dynamic Power Consumption
2	300 MHz	600 Mbps	760	344	10 mW
3	275 MHz	825 Mbps	808	357	10 mW
4	300 MHz	1.2 Gbps	1132	367	12 mW
6	250 MHz	1.5 Gbps	1434	388	15 mW
8	275 MHz	2.2 Gbps	1313	394	16 mW
12	250 MHz	3 Gbps	1760	441	19 mW
16	300 MHz	4.8 Gbps	1588	474	21 mW
24	225 MHz	5.4 Gbps	2313	553	30 mW
32	275 MHz	8.8 Gbps	2722	631	33 mW
48	225 MHz	10.8 Gbps	3960	784	50 mW
64	225 MHz	14.4 Gbps	4595	950	58 mW
96	200 MHz	19.2 Gbps	6529	1263	79 mW

relative PLSCODE to every input BBFRAME is written inside another text file.

The second step of the flow involves running the functional simulation. In this step, the parallelism remains the same as it was in the previous step, while the input BBFRAME and PLSCODE vary. At every iteration the encoder reads from the text files m bits of the input BBFRAME and the relative PLSCODE and produces m bits of the output codeword. During each iteration, the encoder's output bits are written into a text file, line by line. This file will be later compared to the output reference one produced in the previous step.

After the simulation is completed, the output codewords generated by both models into the text files are compared together and if no differences are found, the flow is deemed complete. This entire procedure has to be repeated for every supported parallelism.

The whole process is illustrated in Figure 10.

B. SYNTHESIS AND IMPLEMENTATION RESULTS

The synthesis and implementation process was performed using Vivado by Xilinx, with the Xilinx Kintex Ultrascale XQRKU060 FPGA as the target device. This FPGA device is optimized for real-time, high-throughput space applications, and it is radiation tolerant.

To determine the maximum clock frequency for each level of parallelism, we followed an incremental implementation flow based on frequency sweeps to determine the maximum supported frequency. Additionally, we selected the synthesis strategy denoted as *Flow_PerfOptimized_high* and the implementation strategy denoted as *Performance_ExtraTimingOpt* provided by Vivado to emphasize the performance of the IP. Since the entire flow was performed only for the IP module, which will later need to be integrated with other transmitter blocks, the synthesis and implementation were conducted in *out-of-context* mode.

For the throughput estimation, we conducted a best-case analysis using QuestaSim, where no dead cycles were forced in the simulation by the external signals on the pipeline. The simulations, at the maximum supported clock frequency for each level of parallelism, showed that the maximum throughput can be well approximated by multiplying the maximum clock frequency with the parallelism.

The post-synthesis simulation was another step in the IP's characterization flow to estimate power consumption. After the synthesis step, the netlist was extracted and simulated on QuestaSim. For the post-synthesis functional simulation, the same input data vectors were used in text format as for the previous RTL simulation. The PLSCODE was set to a fixed default value since it only determines the input frame length and has no impact on the switching activity. The input data bits were randomly generated in MATLAB, as described

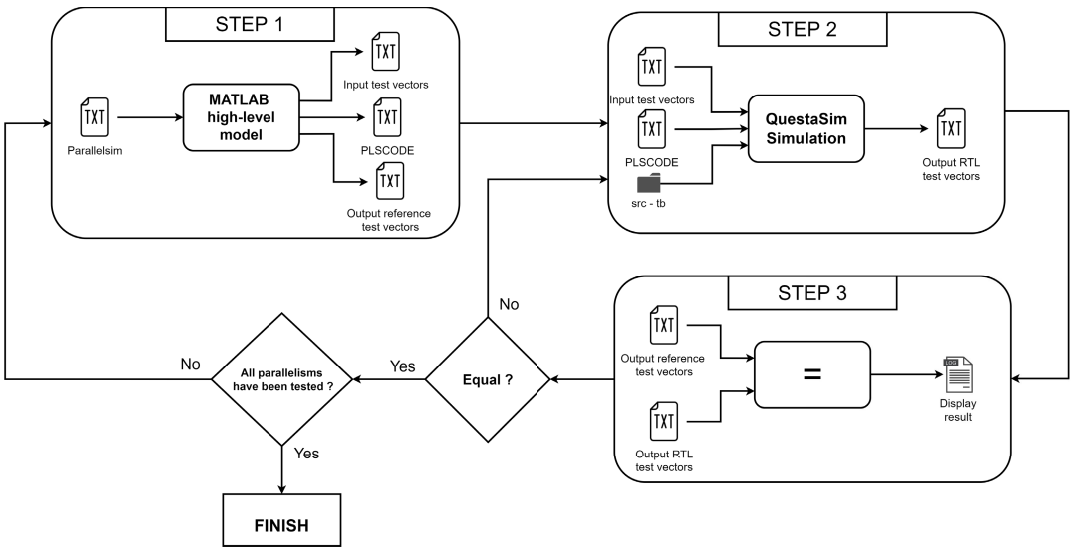

FIGURE 10. Verification flow process diagram.

TABLE 6. State of the art functionality analysis.

Reference	DVB-S2	Mode of Operation	Parallelism	Technology
[7]	Yes	CCM, VCM	5-bit	Xilinx XC7K325t FPGA
[16]	Yes	CCM	Serial	Xilinx Virtex 6 FPGA
[9]	Yes	CCM, VCM, ACM	Serial	Xilinx Virtex2P XC2VP30 FPGA
[18]	No	No	Reconfigurable	Xilinx Virtex-II XC2VP30 FPGA
[14]	No	No	4-bit	Xilinx Spartan 3E FPGA
[17]	No[a]	No	Parallel branches[b]	Spartan 6 XC6SLX45T-FGG484-3C FPGA
[19]	No	No	Parallel	N/A
Proposed Solution	Yes	CCM, VCM, ACM	Reconfigurable	Xilinx Kintex Ultrascale XQRKU060 FPGA

[a] It is DVB-S2 compliant only in concatenated BCH and LDCP FEC encoding structure but it doesn't handle BCH and LDPC codes defined by the standard.
[b] Parallel BCH encoders are concatenated with the LDPC encoder.

in Section IV-A. Several post-synthesis simulations have been conducted, with each one performed for the different levels of parallelism, to estimate the power dependence on parallelism. This allowed us to determine the switching activity on the netlist's nodes, which, when given as input to Vivado, generated a power report that estimated the power consumption of the final IP implementation with a good level of accuracy. It turns out that parallelism only affects the dynamic power consumption, while the static power consumption depends exclusively on the target device and is equal to 625 mW.

Table 5 reports the final results of this activity.

C. COMPARISON WITH THE STATE OF THE ART

In Section II-C, other solutions for BCH encoders have been presented, some of which are combined with other blocks within more complex systems. Consequently, certain metrics in the presented works cannot be directly compared with the proposed IP, as they refer to the entire system rather than just the BCH encoder core. Moreover, certain values are not reported, so in some cases, they have been derived based on assumptions. Table 7 reports these results.

According to Table 6, only [7], [9], and [16] are state-of-the-art solutions that support the DVB-S2 standard.

First, our proposed IP is the only one that allows configurable parallelism. This feature makes it significantly more flexible and scalable compared to the other reference solutions. Specifically, [7] utilizes a parallel BCH encoder architecture with a fixed 5-bit parallelism, whereas [9] and [16] implement a serial architecture. Additionally, our IP stands out for its ability to support all three modes of operation specified by the DVB-S2 standard. Solution [9] is the only other encoder capable of supporting all modes, whereas [7] only supports CCM and VCM, and [16] only supports CCM. Notably, the serial structure of [9] limits its performance, as indicated in Table 7. It can handle a maximum throughput of 182.3 Mbps, whereas our IP can support a maximum throughput of 19.2 Gbps. The extreme scalability and flexibility of our proposed architecture result in higher utilization of FPGA resources for achieving the same sustained throughput. For example, considering that [7] can sustain a maximum throughput of 1.19 Gbps, we observe from Table 7 that it utilizes 545 Flip-FLops (FF) and 646 Look-Up Table (LUT)s. In contrast, our IP, configured to support a similar maximum throughput of 1.2 Gbps with 4-bit parallelism, requires 1132 LUTs and 367 FFs.

An alternative solution is proposed in [17]. This work suggests using a conventional FEC encoding system, which consists of a concatenation of a BCH encoding section and a LDPC encoding section. This approach differs from our proposed solution and other examples found in the literature in that it employs multiple BCH encoders operating in parallel

TABLE 7. State of the art design metrics analysis.

Reference	F_{MAX}	Throughput	Logic Resources	Power Consumption
[7]	389.5 MHz[a]	1.19 Gbps[a]	545 FFs 646 LUTs	N/A
[16]	122 MHz[a]	122 Mbps[c]	N/A	N/A
[9]	182.7 MHz	182.3 Mbps	548 FFs 1233 LUTs	N/A
[18]	657.9 MHz[d]	5.1 Gbps[d]	15 FFs[d] 52 LUTs	N/A
[14]	249.8 MHz	999.2 Mbps[b]	26 FFs 13 LUTs	N/A
[17]	50 MHz[g]	200 Mbps[g]	2568 FFs[g] 2806 LUTs[g]	N/A
[19]	Note[h]	Note[i]	Note[l]	N/A
Proposed Solution	300 MHz[e] 200 MHz[f]	600 Mbps[e] 19.2 Gbps[f]	344 FFs[e] 760 LUTs[e] 1263 FFs[f] 6529 LUTs[f]	635 mW[e] 704 mW[f]

[a] Referred to FEC encoder.
[b] Not specified; derived as the product between F_{MAX} and parallelism.
[c] Not specified; derived by the knowledge of BCH encoder serial architecture.
[d] For 8-bit parallelism.
[e] For 2-bit parallelism (proposed solution).
[f] For 96-bit parallelism (proposed solution).
[g] Referred to both BCH encoder and decoder.
[h] Given as the number of gates in the critical path and is equal to 8.
[i] Given as number of clock cycles per codeword and is equal to 241.
[l] Given as number XORs and is equal to 8952.

branches to increase throughput. This strategy increases throughput by a factor of N over the standard serial solution, where N represents the number of parallel BCH encoder branches.

It is important to note that this approach has a significant drawback - it requires a proportionally higher amount of resources by a factor of N. On the contrary, our proposed solution can increase throughput by a factor of N as well thanks to parallelism expansion, with the advantage of lower logic complexity scaling factor, as depicted in Table 7.

Consider the results shown in Table 5. When we increased the parallelism from 2 to 16, there was an eightfold increase in throughput. However, if we had applied the technique introduced in [17], the number of LUTs and FFs would have also increased by a factor of 8. In contrast, our approach only led to a twofold increase in complexity for LUTs and a 1.4-fold increase for FFs.

Other works, such as [14] and [18], propose examples of parallel BCH encoders where the design is optimized for a single BCH code. This optimization allows for achieving good performance in terms of throughput and clock frequency with minimal resource utilization. Such a design methodology is challenging to implement in our case, where we need to support multiple BCH codes while maintaining reconfigurable parallelism. However, a comparison can still be made. As for the maximum sustainable data throughput [14] and [18] report respectively a value of 5.1 Gbps and 999.2 Mbps. This performance can be reached for the proposed IP as well with a parallelism of respectively 4-bit and 24-bit. In addition, compared with [18] that reaches a maximum clock frequency of 249.8 MHz we can outperform it, by reaching a value of 300 MHz with a 4-bit parallelism configuration. Naturally, the resource utilization is significantly higher, but this is due to the requirement to support all the codes defined by the standard and maintain reconfigurable parallelism by adding dedicated logic for padding and unpadded operations.

The work described in [19] proposes a novel method for reducing the number of logic gates in parallel BCH encoders, based on the mathematical manipulation of equation (15). The derived architecture is similar to the one shown in Figure 6. It incorporates an additional matrix, sparsely populated with ones, alongside an equivalent XOR with two $\mathbf{p}(n-k)$ bits inputs, where $\mathbf{p}(n-k)$ represents the number of checkbits. The function of these additional blocks is to compute the $\mathbf{p}(T)+\mathbf{x}_m$ component outlined in equation (16).

In contrast, our proposed solution employs a two m-bit inputs equivalent XOR gate and a bus concatenation operation, as described in Section III-B. While our approach has the potential to decrease the number of logic gates, it's important to recognize that comparing the two architectures directly poses a challenge because of the substantial differences in their design complexities. Specifically, [19] quantifies the logic gate count within the context of a single (8191, 7684) BCH code with a fixed parallelism, whereas our solution handles multiple BCH codes, with multiple parallelisms.

V. CONCLUSION

In this work, a novel architecture for a BCH encoder with a configurable datapath has been proposed. In particular, the presented circuit is capable of fully supporting the DVB-S2 standard in all its operative modes, with a customizable throughput (and circuit complexity). The circuit indeed can provide a throughput between 600 Mbps and 19.2 Gbps, with power consumption from 10 mW to 79 mW and resource usage from 760 LUTs, 344 FFs to 6529 LUTs, and 1263 FFs. The flexibility of the proposed hardware represents its added value: the same IP can be exploited in dramatically different use case scenarios, allowing future users to exploit the very same IP in low-cost hardware-constrained missions to large high-performance missions with high throughput requirements.

NOTATION

In the following, we report the notation used throughout the document.

- Matrices are represented in bold uppercase letters (e.g., $\mathbf{A}$).
- Vectors are represented in bold lowercase letters (e.g., $\mathbf{x}$).
- Scalars are represented in regular font (e.g., α).
- We treat time as a discrete variable, and we use the term T to represent a generic time point.

REFERENCES

[1] *Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications (DVB-S2)*, document ETSI EN 302 307, 2009.
[2] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[3] A. Goldsmith, *Wireless communications*. Cambridge, U.K.: Cambridge Univ. Press, 2005.
[4] *CCSDS Protocols over DVB-S2—Summary of Definition, Implementation, and Performance*, document CCSDS 130.12-G-1, 2016.
[5] D. Panagiotidis and I. Pitas, "On the capacity increase of digital microwave radio links through the use of adaptive modulation and coding," *IEEE Trans. Commun.*, vol. 48, no. 4, pp. 566–573, Mar. 2000.
[6] S. Ciccarelli and A. Dave, "New space economy drivers and indicators," in *Proc. 32nd IAA Symp. Space Policy, Regulations Econ.*, Washington, DC, USA, Oct. 2019.
[7] J. Kang, J. S. An, and B. Wang, "An efficient FEC encoder core for VCM LEO satellite-ground communications," *IEEE Access*, vol. 8, pp. 125692–125701, 2020.
[8] T. Van Nghia, "Development of the parallel BCH and LDPC encoders architecture for the second generation digital video broadcasting standards with adjustable encoding parameters on FPGA," in *Proc. Int. Conf. Eng. Telecommun. (EnT)*, Nov. 2016, pp. 104–109.
[9] M. Gomes, G. Falcao, V. Silva, V. Ferreira, A. Sengo, L. Silva, N. Marques, and M. Falcao, "Scalable and parallel codec architectures for the DVB-S2 FEC system," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Dec. 2008, pp. 1506–1509.
[10] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, vol. 2, pp. 147–156, Sep. 1959.
[11] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, vol. 3, no. 1, pp. 68–79, Mar. 1960.
[12] W. Peterson, "Encoding and error-correction procedures for the Bose–Chaudhuri codes," *IRE Trans. Inf. Theory*, vol. IT-6, no. 4, pp. 459–470, Sep. 1960.
[13] G. C. Clark and J. B. Cain, *Error-Correction Coding for Digital Communications*. New York, NY, USA: Plenum Press, 1981.

[14] R. Mehra, G. Saini, and S. Singh, "FPGA based high speed BCH encoder for wireless communication applications," in *Proc. Int. Conf. Commun. Syst. Netw. Technol.*, Jun. 2011, pp. 576–579.
[15] J.-L. Issler, *Space Telemetry Use Cases of DVB-S2 CCSDS Standard*, Standard CCSDS 131.3-B-1, Jun. 2016.
[16] D. Digdarsini, D. Mishra, S. Mehta, and T. V. S. Ram, "FPGA implementation of FEC encoder with BCH & LDPC codes for DVB S2 system," in *Proc. 6th Int. Conf. Signal Process. Integr. Netw. (SPIN)*, Mar. 2019, pp. 78–81.
[17] A. Mahdy, M. Helmy, and M. Hassan, "Design and implementation of parallel branches for concatenated BCH and LDPC coding on FPGA," in *Proc. 36th Nat. Radio Sci. Conf. (NRSC)*, Apr. 2019, pp. 324–332.
[18] J.-H. Lee and S. Shakya, "Implementation of parallel BCH encoder employing tree-type systolic array architecture," *Int. J. Sensor Appl. Control Syst.*, vol. 1, no. 1, pp. 1–12, Nov. 2013.
[19] X. Zhang, "High-speed and low-complexity parallel long BCH encoder," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

GIOVANNI QUINTARELLI received the bachelor's and master's degrees in electronic engineering from the University of Pisa, in 2020 and 2023, respectively. He conducted his thesis work with the Department of Information Engineering (DII), Pisa, collaborating with IngeniArs.srl in the area of digital signal processing (DSP) and space communications.

MATTEO BERTOLUCCI received the Ph.D. degree in information engineering from the University of Pisa, in 2022. His study focused on the implementation of digital designs for high-speed satellite communications, such as CCSDS 131.2-B transmitters and receivers on FPGA platforms. He is currently part of IngeniArs, where he works as a DSP and Hardware Engineer for space communications. His latest developments focused on the implementation of a giga-symbol-per-second transmitter for CCSDS 131.2-B, which is capable of achieving over six Gbit/s of actual telemetry downlink, thus enabling the full and spectrally efficient use of Ka-band.

PIETRO NANNIPIERI (Member, IEEE) received the Ph.D. degree (cum laude) in information engineering from the University of Pisa, in 2020. In 2019, he was a Visiting Researcher with the TEC-EDP Section, ESTEC (ESA), where he carried out different qualification tests on the SpaceFibre technology. He is currently an Assistant Professor with the University of Pisa. Indeed, he is working on the European Processor Initiative (EPI) Project. His current research interests include digital and VLSI design, electronics for space applications, cryptography, hardware IPs for satellite onboard data handling, signal processing, and hardware cryptography.

Open Access funding provided by 'Università di Pisa' within the CRUI CARE Agreement