NEXTRACK - Next Generation ESTRACK Uplink Services

R. Abelló[1], M. Baldi[2], F. Chiaraluce[2], R. Fernandes[3], P. Freire da Silva[3], R. Garello[4], D. Gelfusa[5], J. M. Palomo[6], E. Paolini[7], R. Prata[3], L. Santos Ugarte[8], L. Simone[5]

[1]*ESA/ESOC*, Darmstadt, Germany – ricard.abello@esa.int
[2]*Università Politecnica delle Marche* and *CNIT*, Ancona, Italy – email: {m.baldi, f.chiaraluce}@univpm.it
[3]*Deimos Engenharia*, Lisbon, Portugal – email: {ricardo.fernandes, pedro.silva, ricardo.prata}@deimos.com.pt
[4]*Politecnico di Torino and CNIT*, Torino, Italy – email: roberto.garello@polito.it
[5]*Thales Alenia Space*, Rome, Italy – email: {dario.gelfusa, lorenzo.simone}@thalesaleniaspace.com
[6]*Deimos Space S.L.U.*, Madrid, Spain – email: jose-maria.palomo@deimos-space.com
[7]*Università di Bologna* and *CNIT*, Bologna, Italy – email: e.paolini@unibo.it
[8]*WGS Working Solutions working at EUMETSAT*, Darmstadt, Germany – laura.santos@external.eumetsat.int

***Abstract*— The Consultative Committee for Space Data Systems has recently updated its recommendation for uplink communication systems, to cope with new requirements for telecommand and modern profiles and applications. Two short Low-Density Parity-Check (LDPC) codes have been added to the Coding and Synchronization sublayer options, to improve the link performance. In this paper we focus on the real-time implementation of the transmitter for the Ground Station segment. We analyze the critical modules, in particular LDPC encoding, for which two efficient solutions based on a Shift Register Adder Accumulator and on Winograd convolution are considered. We then discuss the selection of a proper hardware or software platform, and we show that a Central Processing Unit-based solution is able to achieve the high data-rates required by the new uplink applications.**

Keywords*— encoders, low-density parity-check codes, real-time implementation, space links, telecommand.

I. Introduction

Uplink digital communication systems from ground station to space were originally designed in the 1970s to fulfil the TeleCommand (TC) control requirements: high reliability, transmission of short messages and receiver simplicity. At that time, the on-board hardware did not allow the implementation of complex decoding algorithms and the flight controllers were simple, requiring few short commands to operate. Since then the Consultative Committee for Space Data Systems (CCSDS) has added a few capabilities to meet new mission requirements. Such capabilities include the extension of the size of command messages, the addition of an optional Cyclic Redundancy Check (CRC), and an Automatic Repeat Request (ARQ) protocol to improve the reliability for larger message sets.

Although TC continues being an important uplink application, future spacecrafts will use uplink communications for a much wider variety of uses. In particular, on-board applications tend to require larger data volumes than in the past and do not need a so high transmission reliability as the TC, for which keeping the undetected bit error rate extremely low (on the order of 10^{-9}) is fundamental.

In order to adapt the current uplink communication protocols to the new user applications, the CCSDS has added two new Low-Density Parity-Check (LDPC) codes with parameters (128, 64) and (512, 256) to its TC Recommendation [1], that already included a simpler Bose–Chaudhuri–Hocquenghem (BCH) (63, 56) code.

The new codes achieve large coding gains and will allow the next generation of Near Earth and Deep Space missions to work with very low signal-to-noise ratio (SNR) operational ranges, and to increase uplink telecommand data rates. This will impose additional constraints to both the Ground Station transmitter and the on-board receiver. An in-depth study on the receiver feasibility and implementation for the new telecommand systems is presented in [2] and [3]. In this paper we focus on the transmitter design and implementation.

The European Space Agency (ESA) has funded a project titled "NEXTRACK - Next Generation ESTRACK (ESA Tracking Stations) Uplink Services" aiming at gaining practical experience in the implementation of the transmitter critical parts. The main scopes of the study are:

- To analyze and design the critical modules for the uplink Coding and Synchronization sublayer, including LDPC encoding, Command Link Transmission Unit (CLTU) generation and randomization, as well as to develop an off-line simulation tool.
- To select a proper platform by comparing hardware and software solutions already available at Telemetry Tracking and Command Processor (TTCP), the ESTRACK's Ground Station Modem, and prototype the critical modules for real-time implementation.
- To test the prototype and validate the results by comparison with an off-line simulation tool.

In this paper we discuss these issues and we present some of the results and conclusions the study has achieved till now. The organization of the paper is as follows. In Section II we introduce the Channel Coding and Synchronization sublayer of the uplink system and we discuss its most important blocks.

Study funded by European Space Agency under the contract 4000124933/18/D/MB.

In Section III we present two efficient solutions for the implementation of the LDPC encoder, which is the most critical block. In Section IV we discuss software and hardware platforms for real-time implementation. In Section V we show that a fully software implementation on a CPU-platform is able to achieve high bitrates. Finally, conclusions are drawn in Section V.

II. Channel Coding and Synchronization Sublayer

The relationship of the Channel Coding and Synchronization Sublayer of the Uplink Communication System to the Open Systems Interconnection (OSI) reference model is shown in Fig. 1.

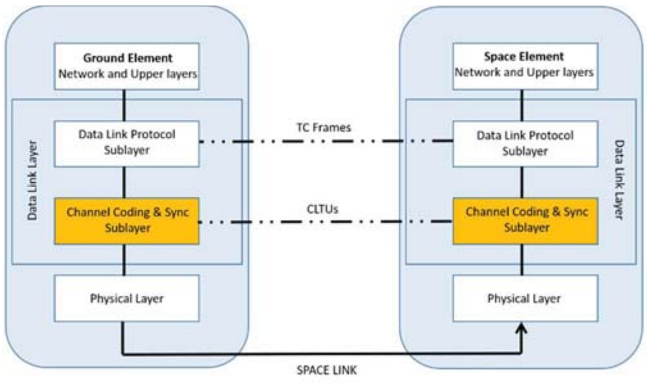

Fig. 1. Channel Coding and Synchronization Sublayer: position on the OSI reference model

As summarized in Fig. 2, if we focus on the Ground Element transmitter, the main blocks of the Channel Coding and Synchronization Sublayer are:

- **Randomizer**, for bit transition generation.
- **Encoder**, for error-control BCH or LDPC coding.
- **CLTU generation**, for delimiting codeblocks.

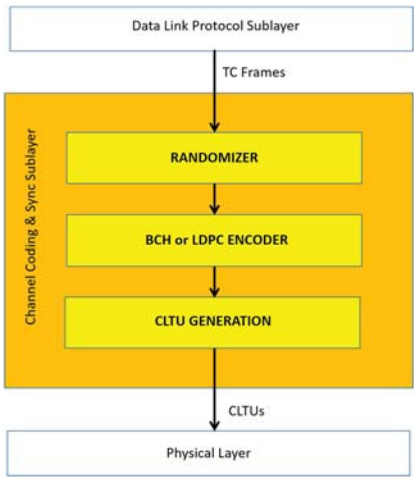

Fig. 2. Channel Coding and Synchronization Sublayer: constituent blocks

A. Randomizer

The randomizer is used to increase the randomness of the binary messages to be transmitted. On-board symbol synchronism circuits need a given transition density, and long runs may prevent a correct working. The randomizer works by summing a pseudo-random maximal length sequence generated by a Linear Feedback Shift Register (LFSR). This guarantees enough transition density and avoids long runs. The CCSDS TC randomizer is made by the 8-cell LFSR with polynomial description $p(D) = 1 + D + D^2 + D^3 + D^4 + D^6 + D^8$ depicted in Fig. 3, which generates a sequence with period $N = 2^8 - 1 = 255$ bits.

This randomizer is optional for BCH coding and mandatory for LDPC coding. When adopted, at the beginning of the Transfer Frame (TF) all the 8 cells are pre-set to 1. Given T the TF length, the first T bits generated by the LFSR are ex-ORed to the TF bits to obtain the T-bit randomized frame to be transmitted, which is then input to the encoder.

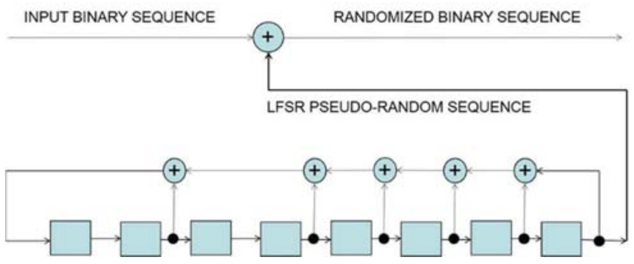

Fig. 3. 8-cell LFSR randomizer

If the TF length is not an integer multiple of the encoder input length, fill bits are appended to obtain an integer multiple, consisting in the repetition of the 01 pattern. According to [1], these fill bits may be added either before or after randomization (since they have already enough bit transition density, then randomization is not strictly required).

Since the LFSR is always preset to the all-one state at the beginning of the TF, the LFSR pseudo-random sequence is always the same, obtained by repeating the main period of $N = 255$ bits.

The real-time implementation of the randomizer can be done by using the logic diagram depicted in Fig. 3 or directly by storing the 255-bit main LFSR period in a memory and summing it when requested, and it is not critical for real-time implementation.

B. BCH encoder

CCSDS TC Recommendation specifies a binary (63, 56) BCH code, where $n = 63$ is the codeword length and $k = 56$ is the information length. The generator polynomial of this BCH code is $g(X) = X^7 + X^6 + X^2 + 1 = (X^6 + X + 1)(X + 1)$. The code may be regarded as an expurgated (63, 57) Hamming code, obtained by allowing even-weight codewords only; its minimum distance is $d_{\min} = 4$, then the code is able to correct single errors and detect up to three errors.

The BCH encoder can be realized by the shift register digital circuits depicted in Fig. 4. This circuit performs systematic encoding of a block $\mathbf{u} = (I_0, I_1, ..., I_{55})$ of 56 bits. At the beginning, the shift register is initialized to all-zero and the two switches are set to the position 1. The encoder is fed serially with the 56 bits $\mathbf{u}$; these bits are propagated to the output and, at the same time, they are processed by the feedback shift register-based digital circuit. When all the 56 bits $\mathbf{u}$ have been input, the two switches are set to position 2: in the seven subsequent clock cycles the complement of the seven BCH parity-check bits $\mathbf{p} = (P_0, P_1, ..., P_6)$ are output by the encoder, and concatenated with $\mathbf{u}$. After these seven clock cycles, the switches are set to position 3, yielding the generation of one bit $F_0 = 0$ (called the filler bit). The binary word $\mathbf{c}$ generated from $\mathbf{u}$ has length 64 bits and is given by $\mathbf{c} = (I_0, I_1, ..., I_{55}, P'_0, P'_1, ..., P'_6, F_0)$ where $P'_i = \mathrm{XOR}(P_i, 1)$ is the complement of bit P_i.

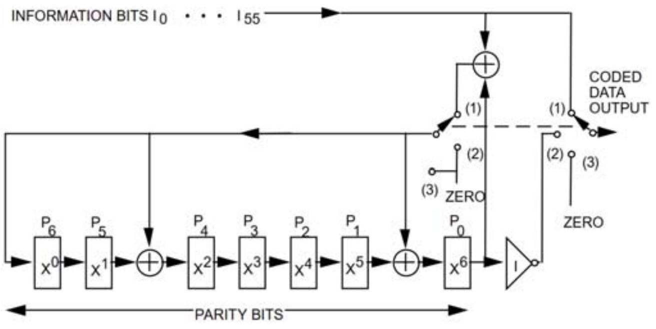

Fig. 4. Encoding circuit for the (63, 56) BCH code

If a block of TFs to be transmitted in one CLTU is not an integer multiple of 56, an appropriate number of fill bits is added to the block. (The fill pattern is a sequence of alternating 0 and 1 bits, starting with a zero.) The BCH encoder real-time implementation is not consider critical.

C. *LDPC encoder*

The LDPC codes introduced in CCSDS TC recommendation [1] have parameters (128, 64) and (512, 256). These codes are defined through their parity check matrix **H**, which can be obtained starting from the structures specified in Fig. 5. According to the figure, the parity-check matrices are constructed from $Q\times Q$ submatrices where $Q = k/4 = n/8$: we have $Q = 16$ for the (128, 64) LDPC code and $Q = 64$ for the (512, 256) LDPC code.

$$\mathbf{H}_{64\times128}=\begin{bmatrix} \mathbf{I}_Q\oplus\mathbf{\Phi}^7 & \mathbf{\Phi}^2 & \mathbf{\Phi}^{14} & \mathbf{\Phi}^6 & \mathbf{0}_Q & \mathbf{\Phi}^0 & \mathbf{\Phi}^{13} & \mathbf{I}_Q \\ \mathbf{\Phi}^6 & \mathbf{I}_Q\oplus\mathbf{\Phi}^{15} & \mathbf{\Phi}^0 & \mathbf{\Phi}^1 & \mathbf{I}_Q & \mathbf{0}_Q & \mathbf{\Phi}^0 & \mathbf{\Phi}^7 \\ \mathbf{\Phi}^4 & \mathbf{\Phi}^1 & \mathbf{I}_Q\oplus\mathbf{\Phi}^{15} & \mathbf{\Phi}^{14} & \mathbf{\Phi}^{11} & \mathbf{I}_Q & \mathbf{0}_Q & \mathbf{\Phi}^3 \\ \mathbf{\Phi}^0 & \mathbf{\Phi}^1 & \mathbf{\Phi}^9 & \mathbf{I}_Q\oplus\mathbf{\Phi}^{13} & \mathbf{\Phi}^{14} & \mathbf{\Phi}^1 & \mathbf{I}_Q & \mathbf{0}_Q \end{bmatrix}$$

$$\mathbf{H}_{256\times512}=\begin{bmatrix} \mathbf{I}_Q\oplus\mathbf{\Phi}^{63} & \mathbf{\Phi}^{30} & \mathbf{\Phi}^{50} & \mathbf{\Phi}^{25} & \mathbf{0}_Q & \mathbf{\Phi}^{43} & \mathbf{\Phi}^{62} & \mathbf{I}_Q \\ \mathbf{\Phi}^{56} & \mathbf{I}_Q\oplus\mathbf{\Phi}^{61} & \mathbf{\Phi}^{50} & \mathbf{\Phi}^{23} & \mathbf{I}_Q & \mathbf{0}_Q & \mathbf{\Phi}^{37} & \mathbf{\Phi}^{26} \\ \mathbf{\Phi}^{16} & \mathbf{\Phi}^0 & \mathbf{I}_Q\oplus\mathbf{\Phi}^{55} & \mathbf{\Phi}^{27} & \mathbf{\Phi}^{56} & \mathbf{I}_Q & \mathbf{0}_Q & \mathbf{\Phi}^{43} \\ \mathbf{\Phi}^{35} & \mathbf{\Phi}^{56} & \mathbf{\Phi}^{62} & \mathbf{I}_Q\oplus\mathbf{\Phi}^{11} & \mathbf{\Phi}^{58} & \mathbf{\Phi}^3 & \mathbf{I}_Q & \mathbf{0}_Q \end{bmatrix}$$

Fig. 5. Parity check matrices of the LDPC codes

With reference to the circulant matrices shown in Fig. 5, $\mathbf{I}_Q$ and $\mathbf{0}_Q$ are the $Q\times Q$ identity and zero matrices, respectively, while $\mathbf{\Phi}$ is the first right circular shift of $\mathbf{I}_Q$. Explicitly, this means that $\mathbf{\Phi}$ has a non-zero entry at row i and column j if and only if $j = i + 1 \bmod Q$. Consequently, $\mathbf{\Phi}^2$ is the second right circular shift of $\mathbf{I}_Q$, that is, $\mathbf{\Phi}^2$ has a non-zero entry at row i and column j if and only if $j = i + 2 \bmod Q$, and so on. Obviously, $\mathbf{\Phi}^0 = \mathbf{I}_Q$. The $\oplus$ operator indicates modulo-2 element-wise matrix addition.

These LDPC codes were designed starting from protographs [4] and achieve very good performance. On-board they can be decoded by usual iterative LDPC decoding algorithm. As an alternative, a non-iterative Most Reliable Basis (MRB) algorithm can be applied, at least for the shorter code, which achieves a higher coding gain [5]-[7].

Starting from the parity check matrix, a generator matrix **G** can be obtained, by using the equation $\mathbf{G}\cdot\mathbf{H}^{\mathrm{T}} = \mathbf{0}_{k\times r}$ where superscript T denotes transposition, $r = n - k$ is the number of parity check symbols, and $\mathbf{0}_{k\times r}$ is the all-zero matrix with the specified size. Since the code is systematic, the matrix **G** can be written as $\mathbf{G} = [\mathbf{I}_{4Q}\ \mathbf{W}]$, where $\mathbf{I}_{4Q}$ is $4Q\times4Q$ identity matrix and **W** is a block-circulant matrix. More in detail, the structure of **G** is shown in Fig. 6, where every $\mathbf{W}_{i,j}$ is a (dense) circulant $Q\times Q$ square matrix.

$$\mathbf{G}=\begin{bmatrix} \mathbf{I}_Q & \mathbf{0}_Q & \mathbf{0}_Q & \mathbf{0}_Q & \mathbf{W}_{1,1} & \mathbf{W}_{1,2} & \mathbf{W}_{1,3} & \mathbf{W}_{1,4} \\ \mathbf{0}_Q & \mathbf{I}_Q & \mathbf{0}_Q & \mathbf{0}_Q & \mathbf{W}_{2,1} & \mathbf{W}_{2,2} & \mathbf{W}_{2,3} & \mathbf{W}_{2,4} \\ \mathbf{0}_Q & \mathbf{0}_Q & \mathbf{I}_Q & \mathbf{0}_Q & \mathbf{W}_{3,1} & \mathbf{W}_{3,2} & \mathbf{W}_{3,3} & \mathbf{W}_{3,4} \\ \mathbf{0}_Q & \mathbf{0}_Q & \mathbf{0}_Q & \mathbf{I}_Q & \mathbf{W}_{4,1} & \mathbf{W}_{4,2} & \mathbf{W}_{4,3} & \mathbf{W}_{4,4} \end{bmatrix}$$

Fig. 6. Generator matrix structure

The scatter chart for the generator matrix of the (128, 64) LDPC code is depicted in Fig. 7, where blue dots represent bits equal to 1 while all remaining bits are 0. We can note that the **G** right side is dense (about $r^2/2$ bits equal to 1).

Given the information vector **u**, we can obtain the corresponding codeword as $\mathbf{c} = \mathbf{uG}$. Anyway, due to the density of **G**, encoding is not optimized. LDPC encoding is then a critical block for real-time implementation. For this reason, in the next section we will consider two efficient alternative encoding solutions.

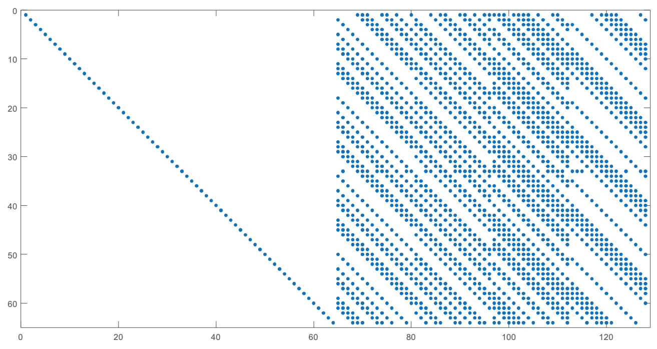

Fig. 7. Scatter chart for the generator matrix of the (128, 64) LDPC code

D. *CLTU generation*

After channel encoding has been realized, the CLTU generation block performs codeword concatenation and prepends to the block of concatenated codewords a CLTU start sequence. The start sequence depends on the adopted channel coding mode. When BCH coding has been selected, the 16-bits start sequence 1110101110010000 is used. On the other hand, when LDPC coding has been selected, the 64-bits start sequence (here expressed in hexadecimal form) 0347 76C7 2728 95B0$_{\mathrm{HEX}}$ is used.

After the start sequence has been prepended, the CLTU generation module appends a tail sequence when the chosen configuration requires it. In case BCH encoding has been selected, the mandatory 64-bit tail sequence (expressed in hexadecimal form) C5C5 C5C5 C5C5 C579$_{\mathrm{HEX}}$ is appended to the block of concatenated codewords. On the other hand, when the system configuration consisting of the (128, 64) LDPC coding with CLTU tail sequence has been selected, the optional 128-bit tail sequence (again here expressed in hexadecimal form) 5555 5556 AAAA AAAA 5555 5555 5555 5555$_{\mathrm{HEX}}$ may be appended. This tail sequence is mandatory if we want to apply decoders based on MRB for LDPC complete decoding [2], [3]. Unlike, the tail sequence is not used at all for the (512, 256) LDPC coding (that, in fact, cannot use MRB, because of the high complexity it implies for, relatively, long codes).

The CLTU generation block is quite simple and is not considered critical for real-time implementation.

III. Efficient LDPC encoding

Both the (128, 64) and the (512, 256) TC LDPC codes belong to the class of quasi-cyclic (QC) LDPC codes, a feature that can be exploited to make their encoding (both in hardware and software implementation) particularly efficient. In this section we present two methods that can reduce the complexity.

A. *SRAA-based encoding*

As explained in the previous section, given the systematic generator matrix $\mathbf{G} = [\mathbf{I}_{4Q}\ \mathbf{W}]$ of the TC LDPC codes, the $\mathbf{W}$ matrix is dense, then the complexity of the straightforward encoding $\mathbf{c} = \mathbf{uG}$ may be problematic on some platforms. A method to circumvent this problem consists of exploiting the QC property possessed by both TC LDPC codes. To do so, let us focus on the structure of the matrix G reported in Fig. 6. As mentioned above, each $\mathbf{W}_{i,j}$ is a square $Q\times Q$ (generally dense) circulant matrix, meaning that any of its rows may be obtained as the right circular shift of the previous row by one position.

To efficiently perform the product $\mathbf{uG}$ (both in software and in hardware), we can write the information block $\mathbf{u}$ as $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4)$, where all $\mathbf{u}_i$, $i = 1, ..., 4$, have length Q bits. Hence, we have $\mathbf{c} = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$, where

$$\mathbf{p}_j = \mathbf{u}_1\mathbf{W}_{1,j} + \mathbf{u}_2\mathbf{W}_{2,j} + \mathbf{u}_3\mathbf{W}_{3,j} + \mathbf{u}_4\mathbf{W}_{4,j}.$$

Next we exploit the fact that all $\mathbf{W}_{i,j}$ matrices are $Q\times Q$ circulant blocks. For such a purpose, we denote by $\mathbf{g}_{i,j}^{(l)}$ the l-th row of $\mathbf{W}_{i,j}$, for $l = 1, ..., Q$. With reference to the generic term involved in the above expressions of $\mathbf{p}_j$, letting $\mathbf{u}_i = (u_{(i-1)Q+1}, u_{(i-1)Q+2}, ..., u_{iQ})$, we have

$$\mathbf{u}_i\mathbf{W}_{i,j} = u_{(i-1)Q+1}\mathbf{g}_{i,j}^{(1)} + u_{(i-1)Q+2}\mathbf{g}_{i,j}^{(2)} + \ldots + u_{iQ}\mathbf{g}_{i,j}^{(Q)}.$$

The calculation of each $\mathbf{p}_j$ requires to perform the above computation four times, one per each $i = 1, ..., 4$. In hardware, the calculation of $\mathbf{p}_j$ may be performed efficiently using a digital circuit called Shift Register Adder Accumulator (SRAA) [8], shown in Fig. 8.

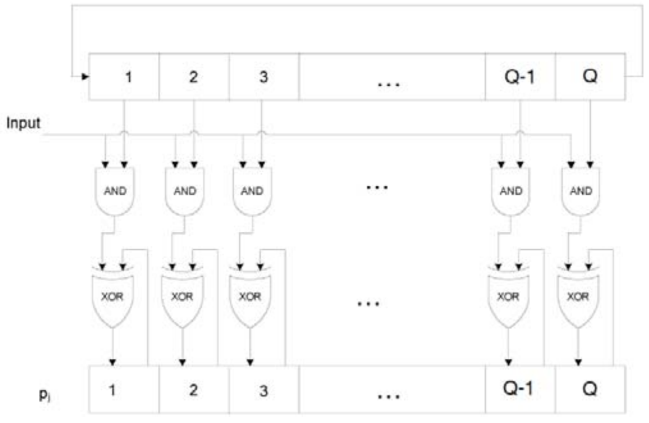

Fig. 8. SRAA encoder

This circuit features two shift registers for Q bits each, as well as Q logic AND and Q logic XOR gates. The calculation of $\mathbf{p}_j$ is performed in four different phases, one per each $i = 1$, ..., 4. At the beginning of phase i, the vector $\mathbf{g}_{i,j}^{(1)}$ is preloaded in the first register, which is in charge of generating all rows of $\mathbf{W}_{i,j}$ in the subsequent $Q - 1$ clock cycles. In each clock cycle of phase i, all elements of the shift register are multiplied (logical AND) by the bit $u_{(i-1)Q+t}$, $t = 1, \ldots, Q$, and the result is accumulated (logical XOR) in the corresponding position of the second Q-bits register. At the end of the four phases, the latter register contains $\mathbf{p}_j$. Overall, the efficient encoder is composed of four SRRA circuits.

Besides being very useful for hardware implementation of the LDPC encoder (for both the length-128 and the length-512 codes), this encoding strategy may be exploited in a software implementation, too, yielding a much more efficient encoding algorithm than the straightforward approach consisting of direct $\mathbf{uG}$ multiplication.

B. *Winograd-based encoding*

In this section we present an alternative encoding technique, based on Winograd convolution [9]. This method exploits the fact that circulant matrices are Toeplitz matrices.

In a Toeplitz matrix, all elements on a descending diagonal from left to right are constant. Any circulant matrix is clearly a Toeplitz matrix, as can be seen in this example of a 4×4 circulant matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{T}_0 & \mathbf{T}_1 \\ \mathbf{T}_2 & \mathbf{T}_0 \end{bmatrix}.$$

Now, any $p\times p$ Toeplitz matrix $\mathbf{T}$ with even p can be decomposed as

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_0 & \mathbf{T}_1 \\ \mathbf{T}_2 & \mathbf{T}_0 \end{bmatrix} =$$

$$= \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 - \mathbf{T}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_2 - \mathbf{T}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{I} \end{bmatrix} = \mathbf{ABC}$$

where:

- $\mathbf{0}$ and $\mathbf{I}$ are $p/2\times p/2$ null and identity matrices, respectively;
- $\mathbf{T}_0$, $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{T}_1 - \mathbf{T}_0$, $\mathbf{T}_2 - \mathbf{T}_0$ are $p/2\times p/2$ Toeplitz matrices.

The above decomposition can be exploited to perform efficiently the product $\mathbf{vT} = \mathbf{vABC}$, where $\mathbf{v} = [\mathbf{v}_0\ \mathbf{v}_1]$ is a vector of length p ($\mathbf{v}_0$ and $\mathbf{v}_1$ are both of length $p/2$).

The product $[\mathbf{v}_0\ \mathbf{v}_1]\mathbf{ABC}$ is developed in three steps as follows:

1. *Evaluation:*

$$[\mathbf{v}_0\ \mathbf{v}_1]\ \mathbf{A} = [\mathbf{v}_0\ \mathbf{v}_1\ \mathbf{v}_0{+}\mathbf{v}_1].$$

This first step requires $p/2$ additions.

2. *Multiplication:*

$$[\mathbf{v}_0\ \mathbf{v}_1\ \mathbf{v}_0 + \mathbf{v}_1]\ \mathbf{B} = [\mathbf{v}_0(\mathbf{T}_1 - \mathbf{T}_0)\ \mathbf{v}_1(\mathbf{T}_2 - \mathbf{T}_0)\ (\mathbf{v}_0 + \mathbf{v}_1)\mathbf{T}_0]$$
$$= [\mathbf{z}_0\ \mathbf{z}_1\ \mathbf{z}_2].$$

This second step requires three products of vectors of length $p/2$ by $(p/2)\times(p/2)$ Toeplitz matrices.

3. *Interpolation*

$$[\mathbf{z}_0\ \mathbf{z}_1\ \mathbf{z}_2]\mathbf{C} = [\mathbf{z}_1 + \mathbf{z}_2\ \ \mathbf{z}_0 + \mathbf{z}_2].$$

This final step requires two additions of $p/2$-bit vectors.

It is important to note that in step 2 we trace back the product of one vector of length p by a Toeplitz matrix of dimension p to 3 products of vectors of length $p/2$ by Toeplitz matrices of dimension $p/2$. This process can be iterated, as far as we get matrices of even dimension. Since the vector-matrix product has quadratic complexity, overall we get considerable savings in terms of elementary operations.

Let us return to our encoding problem and consider again the generation of

$$\mathbf{p}_j = \mathbf{u}_1\mathbf{W}_{1,j} + \mathbf{u}_2\mathbf{W}_{2,j} + \mathbf{u}_3\mathbf{W}_{3,j} + \mathbf{u}_4\mathbf{W}_{4,j} \qquad (j = 1, 2, 3, 4)$$

as stated in Section III.A. To calculate each $\mathbf{p}_j$ we need to perform 4 products of vectors of length Q (= 16 or 64) by $Q\times Q$ Toeplitz matrices. As a consequence, encoding can be performed efficiently by applying the described technique 16 times, one for each $\mathbf{W}_{i,j}$.

We finally note that:

- The 16 vector-matrix products can in principle be performed in parallel.
- For each $j = 1, 2, 3, 4$, we need to perform 3 (binary) sums of vectors of length Q to assemble $\mathbf{p}_j$. In total, we have 12 sums of vectors of length Q.
- Step 1 can be performed only once for each $\mathbf{u}_i$, thus producing further savings in complexity.

This technique looks then very promising for real-time implementation.

IV. Selection of the Platform

One of the initial project goals was to select the platform where the critical modules will be implemented having into account a future portability towards an operational platform available at TTCP. This section describes the possible platforms and the respective selection.

A. *Plaforms available at TTCP*

ESA has the following three kinds of platforms available at the TTCP: CPU, ARM based FPGA and FPGA. These platforms are integrated in two types of units/devices: Data Processing Unit (DPU) and Signal Processing Modules (SPM).

A DPU includes two processors Intel Xeon CPU E5-2637 v4 @ 3.50 GHz while a SPM has inside two FPGAs from Altera/Intel: the FPGA Stratix which is a powerful pure FPGA, and a Cyclone SoC FPGA with dual-core ARM CortexA9.

The Intel Xeon processor E5 v4 is a multi-core enterprise processor built on 14nm process technology designed to have low power and high performance, the processor was designed for a platform consisting of a processor and the Platform Controller Hub (PCH) supporting up to 46 bits of physical address space and 48-bits of virtual address space. Table I addresses the main features of the CPU platform available at TTCP.

TABLE I. CPU PLATFORM SPECIFICATION (FROM ESA)

Features	Values
CPU(s):	16
On-line CPU(s) list:	0-15
Thread(s) per core:	2
Core(s) per socket:	4
Socket(s):	2

Features	Values
Model name:	Intel® Xeon® CPU E5-2637 v4 @ 3.50GHz
L1d cache:	32K
L1i cache:	32K
L2 cache:	256K
L3 cache:	15360K
NUMA node0 CPU(s):	0-3, 8-11
NUMA node1 CPU(s):	4-7,12-15

The Cyclone FPGA built on 28-nm low-power process provides a low cost and low power system, achieving 40 percent lower total power compared with the previous generation. It has efficient logic integration capabilities, integrated transceiver variants, and SoC FPGA with an ARM-based Hard Processor System (HPS).

The capabilities and logic integration were improved thanks to an 8-input Adaptive Logic Module (ALM), with up to 12MB of memory and variable precision digital signal processing (DSP) blocks. As shown in Fig. 9, Cyclone integrates a HPS which includes processors, peripherals, and memory controller with the FPGA fabric using a high-bandwidth interconnect backbone.

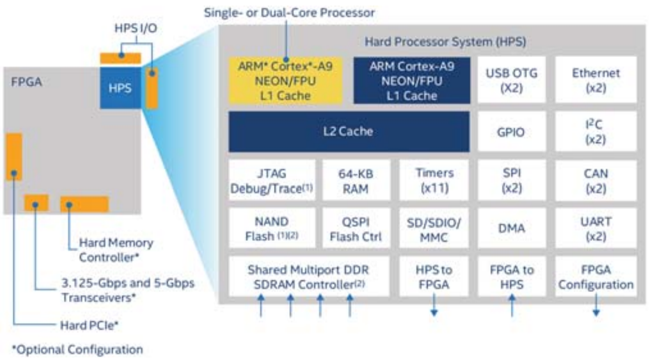

Fig. 9. HPS block diagram of Altera Cyclone SoC (from Altera/Intel)

The Stratix device offers up to 48 integrated transceivers with 14.1 Gbps data rate capability. These transceivers also support backplane and optical interface applications. It has a rich set of high-performance building blocks, including a redesigned Adaptive Logic Module (ALM), 20 Kbit (M20K) embedded memory blocks, variable precision DSP blocks, and fractional Phase-Locked Loops (PLLs). All these building blocks are interconnected by a multi-track routing architecture and comprehensive fabric clocking network. The main features of the FPGA platform are depicted in Table II.

TABLE II. FPGA PLATFORM SPECIFICATION (FROM ALTERA/INTEL)

Features	Stratix
Logic Elements (K)	952
ALMs	359,200
Registers (K)	1,437
M20K Memory Blocks	2,640
M20K Memory (Mbits)	52
MLAB Memory (Mbits)	10.96
Variable Precision DSP blocks (27x27)	352
Variable Precision Multipliers (18x18)	704

Features	Stratix
Global clock networks	16
Regional clocks	92
LVDS channels, 1.4 Gbps (receive/transmit)	210
14.1-Gbps Transceivers	48
PCIe hard IP Blocks	4
Fractional PLLs	28
DDR3 SDRAM x72 DIMM Interfaces	6

B. Requirements

Besides the usage of one of the three platforms available, the NEXTRACK target performance has been taken into account, namely, the critical modules shall support the data rates defined in the CCSDS [10], from 7.8125 bps to 2.048 Mbps.

As it was already mentioned in section II, the LDPC encoding is the most complex and demand module to be implemented. Therefore, in the preliminary design stage of the project, this module was the one considered to evaluate and select the platform. Considering the CPU clock of 3.5 GHz (available at TTCP) which corresponds an instruction cycle of 0.29 ns enables 1709 instruction cycles per each bit. Table III presents the resulting maximum allowed spent time for encoding a single codeword.

TABLE III. TARGET PERFORMANCE CONSIDERING ONLY THE ENCODER

Target output data rate [Mbps]	**2.048**
CPU clock [GHz]	**3.50**
Instruction cycle [ns]	0.29
Number of bits - LDPC(128,64)	128
Number of bits - LDPC(512,256)	512
Number of bits - BCH(63,56)	63
Number of instructions cycles per bit	**Max time for channel encoding [us]**
1709.0	**62.5**
	250.0
	30.8

C. CPU platform

For a first evaluation we compared the CPU performance available at TTCP with several processors available in the consortium. The Intel Xeon CPU E5-1620 v2 @ 3.70GHz was used with similar features and performance to the one available at TTCP, as shown in Fig. 10.

The CPU used in this evaluation has a clock slightly higher than the target (3.7 GHz vs 3.5 GHz), however, it is a Xeon processor from first generation while the CPU available at TTCP is from second generation. Therefore, considering these features we can conclude that both performances should be similar. This can be confirmed in Fig. 10, where the rating of ESA CPU (second bar) is slightly better than the one used by the consortium (first bar). For the platform evaluation, a preliminary/simple code has been developed in C/C++ language where the compiler GCC version 7.3.0 was used and optimization level of "-O3" (maximum performance) was selected.

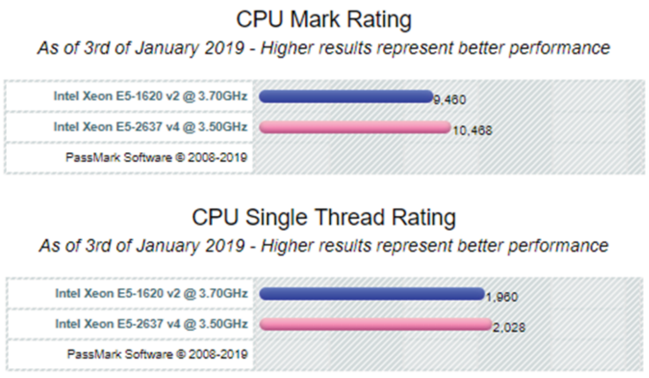

Fig. 10. CPUs comparison (from CPU Benchmark)

As a preliminary assessment, instead of the two efficient algorithms described in section III, the classic **uG** encoder (matrix multiplication) has been used in the assessment. Table IV presents the worst performance results achieved for the CPU platform considering the encoding of a complete codeword (start and tail sequences not included).

TABLE IV. PERFORMANCE ACHIEVED FOR CPU PLATFORM

Channel encoding	Output bit rate [Mbps]
LDPC(128,64)	≥ 22.1
LDPC(512,256)	≥ 4.8
BCH(63,56)	≥ 45

It can be concluded that the CPU performance complies with the target bit rate for both LDPCs and BCH. In the performed tests, the achieved bit rate is not always the same, but it has some fluctuations which can be related with the CPU resources allocation automatic assignment by the operating system. For instance, sometimes prior to the first execution the CPU clock is about 1.5 GHz instead of 3.5 GHz.

If the CPU platform is chosen, the classic **uG** encoder will be implemented since the required performance in principle is achievable. Additionally, the consortium intends to implement at least one of the efficient encoders described in section III to further increase the achievable bit-rates.

Furthermore, it should be highlighted that these results do not include the start and tail sequences, which are easy to add to the codewords with few processing time required and consequently increasing the bit rate. It is expected to increase the performance in about 50% (64/128 bits) or 150% ((64+128)/128 bits) for LDPC(128,64), when only start sequence or start and tail sequences are used, respectively, and 12.5% for LDPC(512,256), (64/512 bits).

D. HW platforms (ARM based FPGA and FPGA)

The SRAA based encoder of Fig. 8 is efficient in case of hardware implementation. The encoder parallel structure presented in Fig. 11 has then been evaluated for both Cyclone and Stratix FPGA devices. The parallel structure has been chosen with respect to the iterative architecture due to the few logic required to implement the SRAA circuit and to take advantage of the higher bit rate achievable with the parallel solution. In Table V and Table VI, the estimated complexity and timing performances achieved by synthesis and place&route performed with Altera Quartus software are reported for both Cyclone and Stratix device and both LDPC codes.

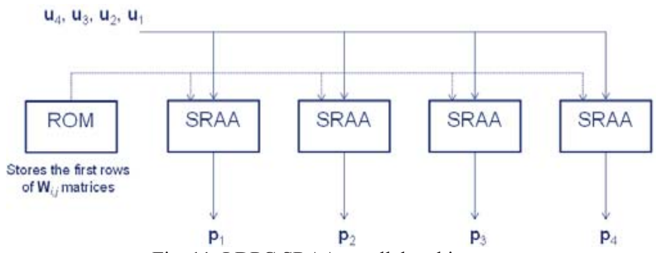

Fig. 11. LDPC SRAA parallel architecture

TABLE V. LDPC(128,64) PERFORMANCE ESTIMATION

LDPC(128,64)	Cyclone		Stratix	
Area report	**Usage**	**%**	**Usage**	**%**
FFs	402	< 1	384	< 1
Combinatorial logic element	234	< 1	347	< 1
Timing report				
Max Frequency	341 MHz		717 MHz	

TABLE VI. LDPC(512,256) PERFORMANCE ESTIMATION

LDPC(512,256)	Cyclone		Stratix	
Area report	**Usage**	**%**	**Usage**	**%**
FFs	1578	< 3	1565	< 1
Combinatorial logic element	1117	< 2	1117	< 1
Timing report				
Max Frequency	264 MHz		627 MHz	

Considering this preliminary implementation on FPGA device and the required design margin (factor of 2.5) to be adopted according to consortium experience in this preliminary phase, a minimum bit rate equal to about 100 Mbps can be considered achievable with hardware implementation on Cyclone device, while 250 Mbps could be achieved with Stratix device in the worst case, for LDPC(512,256) encoder.

The BCH structure (previously shown in Fig. 4) is simple and includes few Flip-Flops (FFs) and combinatorial logic elements as indicated in Table VII.

TABLE VII. BCH ENCODER PERFORMANCE ESTIMATION

BCH encoder	Cyclone		Stratix	
Area report	**Usage**	**%**	**Usage**	**%**
FFs	18	< 1	19	< 1
Combinatorial logic element	13	< 1	12	< 1
Timing report				
Max Frequency	368 MHz		717 MHz	

As per the LDPC encoders, the estimation regarding the maximum frequency is rough due the unconstrained synthesis and place&route: the FPGA device contains only the BCH encoder without other circuits and without any constraints regarding pinout.

This preliminary synthesis and place&route is able to indicate a rough order of magnitude estimation regarding the maximum bit rate allowable with BCH encoder implemented on FPGA. The real maximum bit rate will be lower depending on the physical constraint that will be applied.

Considering the required conservative design margin to be adopted in this preliminary phase (factor of about 2.5 according to consortium experience), a minimum bit rate equal to 150 Mbps can be considerable achievable with hardware implementation on Cyclone device, while 280 Mbps could be achieved with Stratix device.

E. Platform selection

Considering the platforms available at TTCP and the main factors for its choice, namely: the performance evaluation performed in the initial stage of the project and the future portability towards an operational platform, the consortium considers that the CPU platform is the best choice for the NEXTRACK project. It complies with the target bit rate and additionally it guaranties an easy portability thanks to its software approach. Moreover, the CPU platform has margin to be improved in terms of performance since none of the efficient encoders described in section III has been considered in the evaluation.

V. CONSIDERATIONS ON REAL-TIME IMPLEMENTATION

The requirements of the breadboard include the functionality, interface and configuration requirements. The high-level architecture of the breadboard to the CPU platform is presented in Figure 12. The software will be developed in C/C++ language enabling in the future an easier portability for an operational platform when comparing with the hardware options available, ARM based FPGA and FPGA platforms.

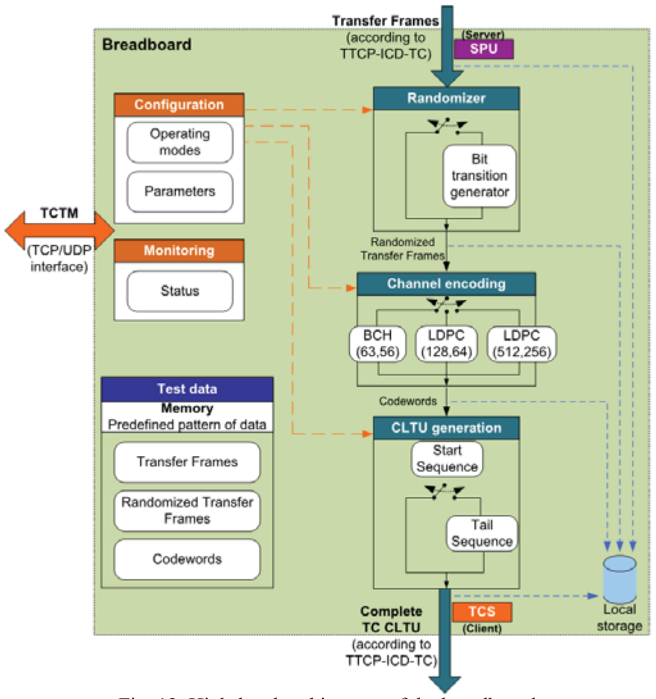

Fig. 12. High-level architecture of the breadboard

The breadboard is composed by a communication link composed of the following 2 interfaces:

- An input to receive Information bits according to TTCP-ICD-TC [10], acting as a server.
- An output to send complete TC CLTU according to TTCP-ICD-TC [10], acting as a client.

Both interfaces will implement the communication diagram described in [10], based on a Transmission Control Protocol (TCP) connection where the SPU designates the system in charge of the TC radiation in the TTCP and it is considered the server, while the TeleCommand System (TCS) designates the Telemetry and Telecommand System (TMTCS) sub-system requesting the TC radiation is considered the client.

In addition, the platform will contain a link for configuration and monitoring, also called TeleCommand and TeleMetry (TCTM) interface. This is an Ethernet interface enabling the breadboard configuration through the reception of parameters sequence selecting the intended operating mode. Besides the nominal/target operating mode, it will also allow to perform Unit Tests (UTs) to evaluate each breadboard module.

The breadboard is composed by a total of seven main modules, as shown in Fig. 12, including the three critical modules for Coding and Synchronization sublayer implementation discussed in the previous sections:

- Randomizer: mandatory or optional depending on the code choice.
- Channel encoding: with different BCH(63,56), LDPC(128,64) or LDPC(512,256) options
- CLTU generation: responsible to generate the breadboard output frames.

The remaining four modules of the breadboard are:

- Configuration: manage the behavior of the entire breadboard. The operation modes are deployed from the configuration module according to the received configuration.
- Monitoring: is in charge of gathering test data since last received configuration, providing telemetry information about the breadboard status which are sent periodically through the TCTM interface. The status includes the number of received transfer frames and the number of generated CLTU frames.
- Test data: in order to allow Unit Tests, this block includes some data patterns for transfer frames, randomized transfer frames and codewords enabling a dedicated test of each module, as well as the entire chain.
- Local Storage: this block saves the useful data like information words, randomized transfer frames, codewords and complete TC CLTU, and also additional data useful for testing and performance assessment and improving, such as processing times of each module.

Finally, the work flow for the breadboard software is presented in Fig. 13. It is composed by three main process: configuration, monitoring and the nominal operating mode, which perform the main encoder tasks.

VI. Conclusions

In this paper we have presented a study on the real-time implementation of the Channel Coding and Synchronization sublayer for the Ground Station segment of the uplink system, taking into account the recent update of the CCSDS Recommendation which included two new LDPC codes to improve the link performance. Two efficient solutions have been presented for LDPC encoding, which is the most critical module. The platform selection has been discussed, by comparing different hardware and software choices. The analysis shows that a Central Processing Unit-based solution is able to achieve the target data-rates and additionally guarantees an easy portability thanks to its software approach.

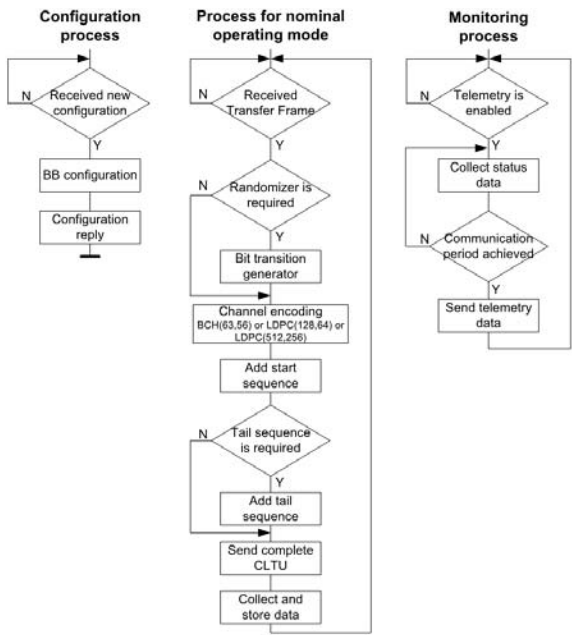

Fig. 13. Breadboard software work flow

References

[1] CCSDS, "TC Synchronization and Channel Coding," Blue Book, CCSDS 231.0-B-3 (Sep. 2017).

[2] M. Baldi, M. Bertinelli, F. Chiaraluce, P. Closas, P. Dhakal, R. Garello, N. Maturo, M. Navarro, J. M. Palomo, E. Paolini, S. Pfletschinger, P. F. Silva, L. Simone, and J. Vilà-Valls. "State-of-the-art space mission telecommand receivers," IEEE Aerospace and Electronic Systems Magazine, vol. 32, no. 6, pp. 4-15, Jun. 2017.

[3] M. Baldi, M. Bertinelli, F. Chiaraluce, P. Freire de Silva, R. Garello, N. Maturo, M. Navarro, J. M. Palomo, E. Paolini, R. Prata, L. Simone, and C. Urrutia, "Theoretical analysis and implementation of effective receivers for telecommand space links," Proc. TTC 2019 International Workshop on Tracking, Telemetry and Command Systems for Space Applications, Darmstadt, Germany, Sep. 2019.

[4] Y. Fang, G. Bi, Y. L. Guan, and F. C. M. Lau, "A survey on protograph LDPC codes and their applications," IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 1989–2016, 4th Quart. 2015.

[5] M. P. C. Fossorier, "Iterative reliability-based decoding of low-density parity check codes," IEEE Journal Selected Areas in Communications, vol. 19, no. 5, pp. 908–917, May 2001.

[6] M. Baldi, N. Maturo, E. Paolini, and F. Chiaraluce. "On the use of ordered statistics decoders for low-density parity-check codes in space telecommand links," EURASIP Journal on Wireless Communications and Networking, 2016:272, 15 pages, 2016.

[7] M. Baldi, F. Chiaraluce, N. Maturo, G. Liva, and E. Paolini. "A hybrid decoding scheme for short non-binary LDPC codes," IEEE Communications Letters, vol. 18, no. 12, pp. 2093-2096, Dec. 2014.

[8] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," IEEE Transactions on Communications, vol. 54, no. 1, pp. 71-81, Jan. 2006.

[9] S. Winograd, "Arithmetic Complexity of Computations", CBMS-NSF Regional Conference Series in Mathematics, vol 33, 1980.

[10] CCSDS, "Radio Frequency and Modulation Systems". Blue Book CCSDS 401.0-B-27 (Oct. 2017).

[11] ESA, "Interface Control Document – TC". TTCP-ICD-TC Issue 1.6, (Nov. 2014).