Received January 26, 2022, accepted February 8, 2022, date of publication February 11, 2022, date of current version February 22, 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3151086*

# An LDPC Encoder Architecture With Up to 47.5 Gbps Throughput for DVB-S2/S2X Standards

**DECAI LIU, YANFEI LUO, YUNFENG LI, ZHIJIE WANG, ZHENGXUAN LI, QIANWU ZHANG, JUNJIE ZHANG, AND YINGCHUN LI**
Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Shanghai 200444, China
Joint International Research Laboratory of Specialty Fiber Optics and Advanced Communication, Shanghai 200444, China
School of Communication and Information, Shanghai University, Shanghai 200444, China

Corresponding author: Yingchun Li (liyingchun@shu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB2900800, in part by the Science and Technology Commission of Shanghai Municipality under Project 20511102400 and Project 20ZR1420900, and in part by the 111 Project under Grant D20031.

**ABSTRACT** Low-Density Parity-Check (LDPC) code is a type of forward error-correction code with excellent performance, and has been widely used in many modern communication standards. The second-generation satellite broadcasting standard (DVB-S2) and its extension (DVB-S2X) adopt a special Irregular Repeated Accumulate (IRA) LDPC code as inner coding scheme. However, due to the large block size, most of the architectures proposed so far use Random Access Memory (RAM) to store and update the encoding results, and the delay caused by address-controlled read and write operations and barrel shift during computation inevitably limits the upper bound of encoder throughput. In this paper, by extracting the periodicity of the parity-check matrix, we introduce a fast encoding algorithm that can efficiently process the multiplication of the information sequence and a large-dimensional sparse matrix, and propose an encoder architecture with low encoding delay and high throughput. The proposed architecture has been implemented and tested on a Xilinx Kintex-7 FPGA, and the result show that the encoder architecture can achieve the highest throughput of 47.5 Gbps at a clock frequency of 280 MHz.

**INDEX TERMS** Low-density parity-check code, DVB-S2/S2X, encoder architecture, high throughput, FPGA.

## I. INTRODUCTION

LOW-DENSITY Parity-Check codes, proposed by Gallager [1], are linear block codes with high encoding gain. Due to the sparsity of the parity-check matrix, LDPC codes have lower decoding complexity than other codes, and have been used in various communication field to approach the Shannon limit better.

To reduce hardware implementation complexity, all LDPC codes in practical applications are structured, i.e., their parity-check matrix consists of an array of cyclic submatrices. These structured LDPC codes are collectively referred to as Quasi-Cyclic LDPC (QC-LDPC) codes. So far, many methods have been proposed to encode QC-LDPC codes, such as the direct method, R-U method [2], Partitioned H method [3], and the hybrid method [4].

In the field of satellite communication, QC-LDPC codes have also been adopted by two standardization organizations, the Consultative Committee for Space Data Systems (CCSDS) and the European Telecommunications Standards Institute (ETSI), and have become the common channel coding scheme. With the integration of space and earth, satellite communication rates will increase to tens or even hundreds of Gbps in the future, bringing new challenges to codec design. Under these circumstances, a channel codec with higher throughput performance is desired.

Two classes of LDPC codes are defined in CCSDS standard [5]: one class of *Accumulate, Repeat by 4, and Jagged Accumulate* (AR4JA) codes is optimized for Deep-Space communication, and another C2 code is intended for Near-Earth communications. Theodoropoulos *et al.* introduced a parallel algorithm for multiple encoding methods [3], achieving state-of-the-art throughput performance for AR4JA codes. However, for C2 code, the direct method is the best choice, which is implemented by multiplying the information

The associate editor coordinating the review of this manuscript and approving it for publication was Luca Barletta.

sequence and the generator matrix. Since the generator matrix of C2 code is in quasi-cyclic form, the Shift-Register-Adder-Accumulator (SRAA) circuit proposed in [6] can be used directly for encoding. However, it leads to a resource burden in the presence of high parallelism, whereas the Recursive Convolutional Encoder (RCE) circuit mentioned in [7] is more suitable for encoding C2 code. By using RCE and ping-pang buffer, the throughput of the encoder architecture for C2 code in [8] reached 3.12 Gbps. Based on this, reference [3] replaced the ping-pang buffer at the input with the PISO registers and achieved a higher throughput of 4.16 Gbps. The work in [9] focuses on optimizing the repeated computation logic in RCE and realizes an encoder with a throughput of 4.69 Gbps while occupying only 1658 Look-Up Tables (LUTs) and 1038 Flip-Flops (FFs).

The second-generation satellite broadcasting standard (DVB-S2) [10] and its extension standard (DVB-S2X) [11] released by ETSI also adopt a Forward Error Correction (FEC) scheme based on a cascaded BCH and LDPC code. For better error correction performance [12], the irregular LDPC code [13] is used in this scheme, and the right-most part of the parity-check matrix is constructed in a dual-diagonal form so that the encoding process has linear complexity. In this paper, we focus on this particular class of structured LDPC codes.

In terms of encoding, the Partitioned H method is particularly well suited for this type of LDPC code, but the multiplication of the information sequence and a large-dimensional sparse matrix involved in the method is related to the complexity of the encoder implementation and is also the main source of encoding delay. To accomplish this step, the encoder architectures proposed in [14]–[19] use the processing method of sequentially computing and updating submatrix, Random Access Memories (RAMs) are used for storage of computation results and parameters of the parity-check matrix. In addition, some logic is required for state control and barrel shift. The advantage of this structure is that it is compatible with a variety of frame lengths and code rates by loading parameters of different parity-check matrices. However, the read-write operation and barrel shift cause a large encoding delay at each update, and the complex control logic causes timing closure difficulties and reduces the device's operating clock frequency. All these factors limited the throughput performance of the encoder and resulted in a maximum throughput of only 10.78 Gbps [19]. To further speed up the encoding process, Lee *et al.* increased parallelism by shifting multiple groups of information sequences simultaneously [20], but this requires that all input information bits be buffered for simultaneous shift operations, which occupies a large number of register resources at high code rate.

To sum up, encoder architectures proposed so far are not efficient enough for high-throughput hardware implementations targeting the specific DVB-S2/S2X code. Although there are some high-throughput encoder architectures for codes with similar characteristics, such as the encoder proposed in [21] for 802.11n/ac. However, this is done by a large number of parallel operations when the block length is short, which is not suitable for DVB-S2/S2X code because the difference in block length is too large.

We notice that the RCE also exhibits high efficiency in processing the multiplication of the information sequence and a large-dimensional sparse matrix, and propose a new LDPC encoder architecture for the DVB-S2/S2X standards that can provide higher throughput performance. The contributions of this paper are as follows:

1) By extracting the periodic structure of the parity-check matrix H, we appropriately transform the form of H and introduce a fast coding algorithm that efficiently solves the multiplication of the information sequence and a large-dimensional sparse matrix.
2) A new LDPC encoder architecture is proposed, which is suitable for high-throughput applications of DVB-S2/S2X code or similar codes.
3) The high performance of the encoder is verified on-chip, and the performance and efficiency at various parameters are analyzed and evaluated, which is of great importance for practical application scenarios.

The rest of this paper is organized as follows: In Section II, we briefly introduce the LDPC code used in DVB-S2/S2X standards. In Section III, we describe the fast encoding algorithm in detail that supports the encoder architecture proposed in Section IV. Section IV gives the overall architecture of the encoder and presents the encoding process. In Section V, the implementation results on Xilinx Kintex-7 FPGA are given, and compared with other methods, the throughput performance and resource utilization efficiency are analyzed. Finally, the conclusion is presented in Section VI.

## II. STANDARDIZED LDPC CODE

The LDPC code can be defined by the parity-check matrix. Normally, it is necessary to obtain a generator matrix for encoding. However, contrary to the sparsity of the parity-check matrix, the generator matrix is dense, which may cause some problems, such as the storage or encoding complexity. The LDPC code adopted in DVB-S2/S2X standards eliminates this problem.

This standardized LDPC code is an Irregular Repeat Accumulate (IRA) code and also a systematic code, that is, the $K-$bit information sequence $\boldsymbol{i} = [i_0, i_1, \ldots, i_{K-1}]$ will be encoded into an $N-$bit codeword $\boldsymbol{c} = [i_0, i_1, \ldots, i_{K-1}, p_0, p_1, \ldots, p_{N-K-1}]$. This LDPC code is defined by a special parity-check matrix $\boldsymbol{H}$, as shown in (1), as shown at the bottom of the next page.

The submatrix $\boldsymbol{B}$ is constrained as a staircase lower triangular matrix. With $\boldsymbol{Hc}^T = \mathbf{0}$, the $N-K$ parity-check bits can be solved from the row of $\boldsymbol{H}$ from top to bottom:

$$\begin{aligned}
p_0 &= a_{0,0}i_0 \oplus a_{0,1}i_1 \oplus \cdots \oplus a_{0,K-1}i_{K-1}\\
p_1 &= a_{1,0}i_0 \oplus a_{1,1}i_1 \oplus \cdots \oplus a_{1,K-1}i_{K-1} \oplus p_0\\
&\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots\\
p_{N-K-1} &= a_{N-K-1,0}i_0 \oplus a_{N-K-1,1}i_1\\
&\quad \oplus \cdots \oplus a_{N-K-1,K-1}i_{K-1} \oplus p_{N-K-2} \qquad (2)
\end{aligned}$$

where $\oplus$ is the addition operator in the Galois Field, i.e., the bitwise XOR. Moreover, the operator $\sum$ appearing later also refers to the sum operation in GF(2). Therefore, the LDPC code can be encoded by (2) without deriving the generator matrix G.

The submatrix $\boldsymbol{A}$ is sparse, and another periodic constraint is imposed to reduce the storage requirements of non-zero elements of the matrix: $K$ *Variable Nodes* (VNs) corresponding to the information sequence are divided into $t$ groups, and each group contains $M$ continuous VNs, i.e. $M \times t = K$, $M = 360$ is a fixed factor in DVB-S2/S2X standards. VNs of one group have the same degree, denoted as $d$. Then as long as the location indices $(l_1, l_2, \ldots, l_d)$ of the $d$ *Check Nodes* (CNs) connected to the first VN in this group are determined, the location indices of the $d$ CNs connected to the $i$-th VN ($i \in \{0, 1, \ldots, M-1\}$) can be obtained by the following formula:

$$\begin{cases} [a_1 + i \times q] \bmod (N-K) \\ [a_2 + i \times q] \bmod (N-K) \\ \vdots \\ [a_d + i \times q] \bmod (N-K), \end{cases} \tag{3}$$

where $q = (N-K)/M$. To improve the performance, submatrix $\boldsymbol{A}$ uses the construction scheme of irregular LDPC code, so that the degree of VNs may vary between groups, but the above constraint is always satisfied in each group. This constraint reduces the storage requirements of the matrix description by a factor of $M$, and there is negligible loss in encoding performance.

## III. FAST ENCODING ALGORITHM

The LDPC code introduced in the previous section is a systematic code, so the input information sequence $\boldsymbol{i}$ mainly participates in the encoding process through the sub-matrix $\boldsymbol{A}$, which can be expressed as:

$$\begin{aligned} \boldsymbol{S}_{1\times(N-K)} &= \boldsymbol{i}_{1\times K} \cdot \boldsymbol{A}^T_{K\times(N-K)} \\ &= [s_0, s_1, \ldots, s_{N-K-1}], \end{aligned} \tag{4}$$

where the operator "$\cdot$" represents matrix multiplication. For $j = 0, 1, \ldots, (N-K-1)$, $s_j$ is calculated as follows:

$$s_j = \sum_{r=0}^{K-1} a_{j,r} i_r = a_{j,0} i_0 \oplus a_{j,1} i_1 \oplus \cdots \oplus a_{j,K-1} i_{K-1}, \tag{5}$$

and using (5), equation (2) can be rewritten as follows:

$$\begin{aligned} p_0 &= s_0, \\ p_1 &= s_0 \oplus s_1, \\ p_2 &= s_0 \oplus s_1 \oplus s_2, \\ &\vdots \\ p_{N-K-1} &= s_0 \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_{N-K-1}. \end{aligned} \tag{6}$$

Thus, the whole encoding process can be divided into two stages: firstly, the intermediate result $\boldsymbol{S}$ of (4) is obtained by multiplying the information sequence $\boldsymbol{i}$ and matrix $\boldsymbol{A}$, and then $N-K$ parity-check bits are obtained by accumulative calculation according to (6). It can also be seen from the formulas that the complexity of encoding is mainly contributed by the multiplication of large-dimensional matrix in the first stage.

Therefore, in the rest of this section, we analyze the structural characteristics of matrix $\boldsymbol{H}$, and introduce a fast encoding algorithm suitable for different code rates and frame lengths through appropriate reshaping operation, which can complete these two processes quickly and efficiently.

### A. COMPUTE THE INTERMEDIATE RESULT S

The constraint introduced in the section II also divides matrix $\boldsymbol{A}$ into $t$ groups. In the M = 360 columns of each group, the indices of non-zero elements of the first column can be found in the appendix of two standards. According to (3), each subsequent column is given by the $q$-bit cyclic shift of the previous column. This means that there is periodicity between rows with an interval of $q$, we can extract these periodic rows and integrate them together in the following ways:

Extract $\boldsymbol{A}'_r$ from $\boldsymbol{A}(r = 0, 1, \ldots, q-1)$:

$$\boldsymbol{A}'_r = \begin{bmatrix} a_{r,0} & a_{r,1} & \cdots & a_{r,K-1} \\ a_{r+q,0} & a_{r+q,1} & \cdots & a_{r+q,K-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r+359q,0} & a_{r+359q,1} & \cdots & a_{r+359q,K-1} \end{bmatrix}$$

Reorganize $\boldsymbol{A}'_r$ into $\boldsymbol{C}$ :

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{A}'_0 \\ \boldsymbol{A}'_1 \\ \vdots \\ \boldsymbol{A}'_{q-1} \end{bmatrix} \tag{7}$$

Through extraction and reorganization, every $M$ associated rows in matrix $\boldsymbol{A}$ are gathered together in matrix $\boldsymbol{C}$.

$$\begin{aligned} \boldsymbol{H}_{(N-K)\times N} &= \left[\boldsymbol{A}_{(N-K)\times K} \boldsymbol{B}_{(N-K)\times(N-K)}\right] \\ &= \left[\begin{array}{cccc:cccccc} a_{0,0} & a_{0,1} & \cdots & a_{0,K-1} & 1 & 0 & 0 & \cdots & 0 & 0 \\ a_{1,0} & a_{1,1} & \cdots & a_{1,K-1} & 1 & 1 & 0 & \cdots & 0 & 0 \\ a_{2,0} & a_{2,1} & \cdots & a_{2,K-1} & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{N-K-2,0} & a_{N-K-2,1} & \cdots & a_{N-K-2,K-1} & 0 & 0 & 0 & \cdots & 1 & 0 \\ a_{N-K-1,0} & a_{N-K-1,1} & \cdots & a_{N-K-1,K-1} & 0 & 0 & 0 & \cdots & 1 & 1 \end{array}\right] \end{aligned} \tag{1}$$

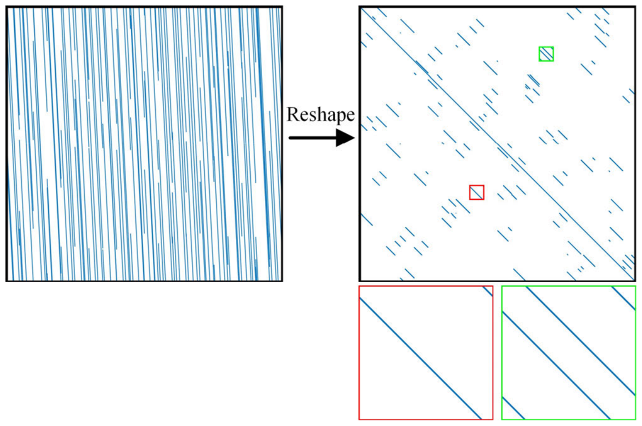


**FIGURE 1. Reshape process of part of matrix $A$. Size of $A$ is 7200×57600 in the case of frame length of 64800 and code rate of 8/9, only the first 7200 columns are displayed in the figure.**

It can be found that the periodicity in $\boldsymbol{A}$ is transformed into the Quasi-Cyclic (QC) characteristic of $\boldsymbol{C}$. Take a part of matrix $\boldsymbol{A}$ (frame length of 64800 and code rate of 8/9) as an example, Fig. 1 shows this reshaping process and the QC characteristics.

The reshaped matrix $\boldsymbol{C}$ is composed of $q\times t$ cyclic matrices:

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_{0,0} & \boldsymbol{C}_{0,1} & \cdots & \boldsymbol{C}_{0,t-1} \\ \boldsymbol{C}_{1,0} & \boldsymbol{C}_{1,1} & \cdots & \boldsymbol{C}_{1,t-1} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}_{q-1,0} & \boldsymbol{C}_{q-1,1} & \cdots & \boldsymbol{C}_{q-1,t-1} \end{bmatrix}, \tag{8}$$

where $\boldsymbol{C}_{i,j}$ is an $M \times M$ cyclic matrix, $i \in \{0, 1, \ldots, q-1\}$, $j \in \{0, 1, \ldots, t-1\}$, that is, each row of $\boldsymbol{C}_{i,j}$ is the cyclic right shift of the previous row, the first row is the cyclic right shift of the last row, each column is the cyclic down shift of the left column, and the first column is the cyclic down shift of the last column.

According to the block properties of $\boldsymbol{C}$, the information sequence can be divided into $t$ segments: $\boldsymbol{i} = [\boldsymbol{i}_0, \boldsymbol{i}_1, \ldots, \boldsymbol{i}_{t-1}]$, where $\boldsymbol{i}_m = [i_{360m}, i_{360m+1}, \ldots, i_{360m+359}]$, $m\in \{0, 1, \ldots, t-1\}$. The row adjustment of (7) has changed the order of the intermediate result $\boldsymbol{S}$. The new result is denoted as $\boldsymbol{S}'$, and is divided into $q$ segments: $\boldsymbol{S}' = [\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_{q-1}]$, where $\boldsymbol{S}_j = [s_j, s_{q+j}, s_{2q+j}, \ldots, s_{359q+j}]$, $j \in \{0, 1, \ldots, q-1\}$. Their relation is expressed as (9).

$$\begin{aligned} \boldsymbol{S}' &= [\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_{q-1}] \\ &= [\boldsymbol{i}_0, \boldsymbol{i}_1, \ldots, \boldsymbol{i}_{t-1}] \cdot \boldsymbol{C}^T \\ &= [\boldsymbol{i}_0, \boldsymbol{i}_1, \ldots, \boldsymbol{i}_{t-1}] \\ &\quad \cdot \begin{bmatrix} \boldsymbol{C}^T_{0,0} & \boldsymbol{C}^T_{1,0} & \cdots & \boldsymbol{C}^T_{q-1,0} \\ \boldsymbol{C}^T_{0,1} & \boldsymbol{C}^T_{1,1} & \cdots & \boldsymbol{C}^T_{q-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}^T_{0,t-1} & \boldsymbol{C}^T_{1,t-1} & \cdots & \boldsymbol{C}^T_{q-1,t-1} \end{bmatrix} \end{aligned} \tag{9}$$

Further, we use $\boldsymbol{c}_{j,m}$ to represent the first row of the matrix $\boldsymbol{C}^T_{j,m}$, whose size is $1 \times 360$. Then according to the cyclic characteristics of $\boldsymbol{C}^T_{j,m}$ (transpose does not change the cyclic characteristics), the solution of each segment $\boldsymbol{S}_j$ in $\boldsymbol{S}'$ can be represented by (10).

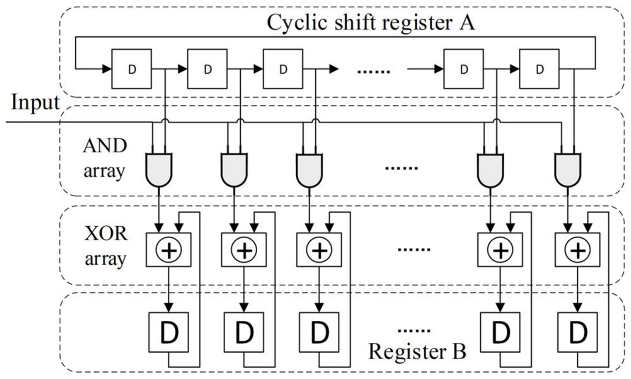


**FIGURE 2. Cyclic shift-register-adder-accumulator structure.**

$$\boldsymbol{S}_j = \sum_{m=0}^{t-1} \boldsymbol{i}_m \cdot \boldsymbol{C}^T_{j,m} \tag{10}$$

To simplify the expression, we denote by $\boldsymbol{v}^{(j)}$ the vector of $j$-bit cyclic right shift of any row vector $\boldsymbol{v}$, then the $\boldsymbol{i}_m \cdot \boldsymbol{C}^T_{j,m}$ in the above equation can be expanded into the following form.

$$\boldsymbol{i}_m \cdot \boldsymbol{C}^T_{j,m} = \sum_{n=0}^{359} i_{360m+n} \cdot \boldsymbol{c}^{(n)}_{j,m} \tag{11}$$

Fig. 2 shows a Cyclic Shift-Register-Adder-Accumulator (CSRAA) circuit that can quickly implement this operation: $\boldsymbol{c}_{j,m}$ is stored in the cyclic shift register B, and then $i_{360m}, i_{360m+1}, \ldots, i_{360m+359}$ are successively entered, register B is cyclic shifted once for each entry. After the input is completed, the accumulation register A obtains the result of (11).

However, it is easy to find that this structure is not suitable for the generally large matrix $\boldsymbol{C}^T$ under the DVB standard. Although, most cyclic submatrices $\boldsymbol{C}^T_{j,m}$ zero matrices $\boldsymbol{0}$ because of sparsity, there will still be tens to hundreds of submatrices whose weight is not 0. If the circuit structure of Fig. 2 is constructed for each of them, the resource overhead of the encoder is unacceptable.

So we further expand (10) and substitute equation (11) to obtain:

$$\boldsymbol{S}_j = \sum_{m=0}^{t-1} \sum_{n=0}^{359} i_{360m+n} \cdot \boldsymbol{c}^{(n)}_{j,m}, \tag{12}$$

obviously the order of multiplication and cyclic shift does not affect the result and can be exchanged:

$$\boldsymbol{S}_j = \sum_{m=0}^{t-1} \sum_{n=0}^{359} \left(i_{360m+n} \cdot \boldsymbol{c}_{j,m}\right)^{(n)}, \tag{13}$$

variables $m$ and $n$ are not related to each other, exchange the order of the summation operators:

$$\boldsymbol{S}_j = \sum_{n=0}^{359} \sum_{m=0}^{t-1} \left(i_{360m+n} \cdot \boldsymbol{c}_{j,m}\right)^{(n)}. \tag{14}$$

At this point, for the inner summation, $n$ is a fixed value, and the cyclic shift can be extracted outside the summation operator and expanded as follows:

$$\begin{aligned}\boldsymbol{S}_j &= \sum_{n=0}^{359}\left(\sum_{m=0}^{t-1} i_{360m+n}\cdot \boldsymbol{c}_{j,m}\right)^{(n)}\\ &= \left(\sum_{m=0}^{t-1} i_{360m}\cdot \boldsymbol{c}_{j,m}\right)\oplus\left(\sum_{m=0}^{t-1} i_{360m+1}\cdot \boldsymbol{c}_{j,m}\right)^{(1)}\\ &\oplus\cdots\oplus\left(\sum_{m=0}^{t-1} i_{360m+359}\cdot \boldsymbol{c}_{j,m}\right)^{(359)}.\end{aligned}\qquad(15)$$

The expanded form of (15) can be rewritten in a recursive manner as (16).

$$\boldsymbol{S}_j = \left(\left(\left(\left(\sum_{m=0}^{t-1} i_{360m+359}\cdot \boldsymbol{c}_{j,m}\right)^{(1)} \oplus \sum_{m=0}^{t-1} i_{360m+358}\cdot \boldsymbol{c}_{j,m}\right)^{(1)} \oplus \sum_{m=0}^{t-1} i_{360m+357}\cdot \boldsymbol{c}_{j,m}\right)^{(1)} \oplus\cdots\cdots\right)^{(1)} \oplus \sum_{m=0}^{t-1} i_{360m}\cdot \boldsymbol{c}_{j,m}\qquad(16)$$

If for any $\boldsymbol{i}_m$, we enter in the reverse order of $i_{360m+359}, i_{360m+358}, \ldots, i_{360m}$, then the cyclic shift operation can be applied to $\boldsymbol{S}_j$ in the following way.

```
for $r = 359 : 0$ do
    $\cdot\boldsymbol{S}_j = \boldsymbol{S}_j^{(1)} \oplus \sum_{m=0}^{t-1} i_{360m+r}\cdot \boldsymbol{c}_{j,m}$
end                                                        (17)
```

The initial value of $\boldsymbol{S}_j$ is $\mathbf{0}_{1\times 360}$. In this way, only the result of $\boldsymbol{S}_j$ needs to be stored, and it does not take up too much resources. The input information sequence is shared by all $\boldsymbol{S}_j$, which means that $\boldsymbol{S}_0 - \boldsymbol{S}_{q-1}$ can be computed simultaneously, so the process of computing $\boldsymbol{S}'$ is summarized as Algorithm 1. In this algorithm, the information sequence is input in $t$-bit in parallel, and each bit of the intermediate result $\boldsymbol{S}$ can be obtained after 360 clock cycles.

## B. ACCUMULATE TO OBTAIN PARITY-CHECK BITS

After $\boldsymbol{S}'$ is calculated, it is a good method to accumulate one by one according to (6) to obtain the parity-check bits. However, to achieve higher throughput, we can fully exploit the order feature of $\boldsymbol{S}_j$ (that is, the order of adjacent bits is separated by $q$) to obtain a parallel output of 360 bits.

**Algorithm 1** Recursive Operation of $S'$

```
Initialize $\boldsymbol{S}'$ with $\mathbf{0}$:
$\boldsymbol{S}' = [\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_{q-1}] = \mathbf{0}_{1\times 360q}$
Recursive computation:
for $r = 359 : 0$ do
    $\boldsymbol{S}_0 = \boldsymbol{S}_0^{(1)} \oplus \sum_{m=0}^{t-1} (i_{360m+r}\cdot \boldsymbol{c}_{0,m})$
    $\boldsymbol{S}_1 = \boldsymbol{S}_1^{(1)} \oplus \sum_{m=0}^{t-1} (i_{360m+r}\cdot \boldsymbol{c}_{1,m})$
    $\vdots \quad \vdots$
    $\boldsymbol{S}_{q-1} = \boldsymbol{S}_{q-1}^{(1)} \oplus \sum_{m=0}^{t-1} (i_{360m+r}\cdot \boldsymbol{c}_{q-1,m})$
end
```

First, rearrange the $q$ segments of $\boldsymbol{S}'$ by column:

$$\begin{bmatrix}\boldsymbol{S}_0\\ \boldsymbol{S}_1\\ \vdots\\ \boldsymbol{S}_{q-1}\end{bmatrix} = \begin{bmatrix} s_0 & s_q & \cdots & s_{359q}\\ s_1 & s_{q+1} & \cdots & s_{359q+1}\\ \vdots & \vdots & \ddots & \vdots\\ s_{q-1} & s_{2q-1} & \cdots & s_{360q-1}\end{bmatrix}_{q\times 360},\qquad(18)$$

sum the matrix in (18) by column to get the following row vector:

$$\left[\sum_{i=0}^{q-1} s_i \quad \sum_{i=q}^{2q-1} s_i \quad \sum_{i=2q}^{3q-1} s_i \quad \cdots \quad \sum_{i=359q}^{360q-1} s_i\right],\qquad(19)$$

then perform another summation: replace the $j$-th element in the row vector with the sum of the first $j$-1 elements, $j\in\{1, 2, \ldots, 360$. The resulting vector is denoted as $\boldsymbol{P}_0$.

$$\boldsymbol{P}_0 = \left[0 \quad \sum_{i=0}^{q-1} s_i \quad \sum_{i=0}^{2q-1} s_i \quad \cdots \quad \sum_{i=0}^{359q-1} s_i\right]\qquad(20)$$

According to (6), the above equation is actually equivalent to $[0, p_{q-1}, p_{2q-1}, \ldots, p_{359q-1}]$. Then add $\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_{q-1}$ one by one to get all parity-check bits:

$$\begin{aligned}[p_0, p_q, \ldots, p_{359q}] &= \boldsymbol{P}_0 \oplus \boldsymbol{S}_0,\\ [p_1, p_{q+1}, \ldots, p_{359q+1}] &= [p_0, p_q, \ldots, p_{359q}] \oplus \boldsymbol{S}_1,\\ &\vdots\\ [p_{q-1}, p_{2q-1}, \ldots, p_{360q-1}] &= [p_{q-2}, p_{2q-2}, \ldots, p_{360q-2}]\\ &\oplus \boldsymbol{S}_{q-1}.\end{aligned}\qquad(21)$$

Adopting this method, all parity-check bits can be generated in 360-bit parallel within $q$ clock cycles. Use vector $\boldsymbol{P}$ to represent the continuously updated 360-bit result, and the accumulation process is summarized as Algorithm 2.

**Algorithm 2** Parallel Processing of Parity-Check Bits

```
Compute $\boldsymbol{P}_0$:
$\boldsymbol{P}_0 = [0, p_{q-1}, p_{2q-1}, \ldots, p_{359q-1}]$
Initialize $\boldsymbol{P}$:
$\boldsymbol{P} = \boldsymbol{P}_0$
Update:
for $j = 0 : q-1$ do
    $\boldsymbol{P} = \boldsymbol{P} \oplus \boldsymbol{S}_j$
end
```

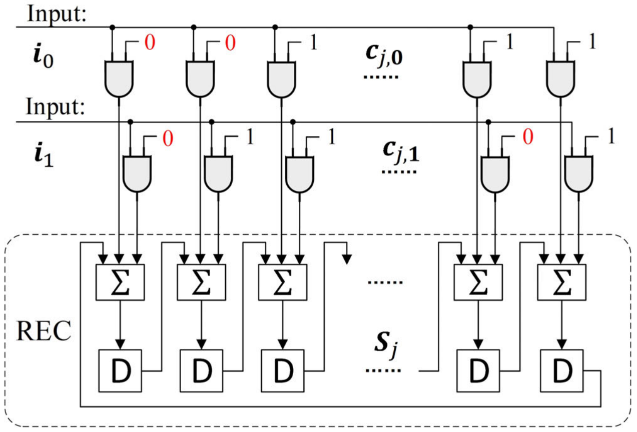


**FIGURE 3.** Recursive encoding core of $S_j$, with $t = 2$.

## IV. PROPOSED ENCODER ARCHITECTURE

Based on the encoding algorithm described in the previous section, a new LDPC encoder architecture suitable for DVB-S2/S2X standards is proposed. This architecture mainly adopts a kind of component composed of registers and adders, which we call *Recursive Encoding Core* (REC). It is extracted from RCE and can quickly complete the relevant computations of the input information sequence. Then, these calculation results are accumulated and the required parity-check bits are output in parallel.

### A. RECURSIVE ENCODING CORE

For a more convenient expression, we can take an example and assume that $t = 2$ in (17), that is, the input information sequence is divided into two segments: $\boldsymbol{i} = [\boldsymbol{i}_0, \boldsymbol{i}_1]$, and the corresponding cyclic submatrix first row $\boldsymbol{c}_{j,0}$ and $\boldsymbol{c}_{j,1}$ are respectively set as:

$$\begin{aligned} \boldsymbol{c}_{j,0} &= [0, 0, 1, \ldots, 1, 1], \\ \boldsymbol{c}_{j,1} &= [0, 1, 1, \ldots, 0, 1] \end{aligned} \tag{22}$$

And the structure shown in Fig. 3 is used to perform the recursive computation of $\boldsymbol{S}_j$.

Fig. 3 contains 360 registers that store the results of $\boldsymbol{S}_j$, above each register is an adder with multiple inputs in GF(2), which is responsible for completing the summation calculation on the right side of the equal sign. At this point, the adder has three summation objects: one is the cyclic right shift of the last result $\boldsymbol{S}_j^{(1)}$, which corresponds to the connection between the register output and the adder input, and the other two are the products of the input information bit and the first row of submatrix $\boldsymbol{C}_{j,m}^T$, denoted as $(i_r \cdot \boldsymbol{c}_{j,0})$ and $(i_{360+r} \cdot \boldsymbol{c}_{j,1})$, and $r$ is from 359 to 0. The AND gate in the figure reflects this product relationship.

It can be seen from Fig. 3 that the register group and adder group enclosed by the dotted line are responsible for the cyclic shift and summation in the algorithm. We call them as a whole the Recursive Encoding Core (REC), which is an important component in the architecture of the encoder.

In addition, we noticed that a large number of AND gates are connected between the REC and the input information bits, which is actually unnecessary. This is because once the parity-check matrix $\boldsymbol{H}$ is determined, the parameter $\boldsymbol{c}_{j,m}$ is also fixed: if one bit of the parameter is 0 (e.g., the four "0"s marked in red in Fig. 3), then the corresponding AND gate and the AND gate output can be directly cleared. On the contrary, if one bit is 1, the AND gate can also be optimized away so that the input bit directly participates in the operation of the adder.

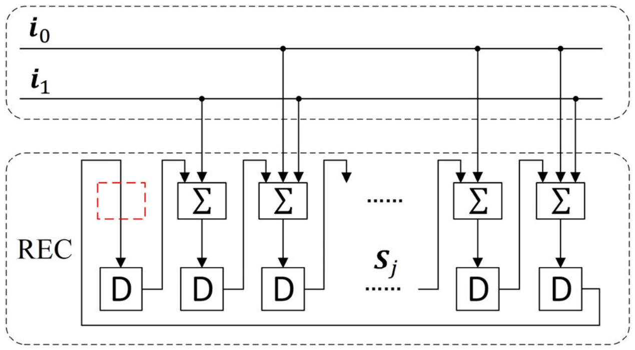


**FIGURE 4.** REC optimized with fixed parameter $c_{j,m}$.

**TABLE 1.** Statistical results of associated input bits of adder.

| Associated Input Bits | 0 | 1 | 2 | 9 |
|---|---|---|---|---|
| Number of Adders | 6128 | 326 | 8 | 18 |

Fig. 4 shows the optimized REC, where the connection between it and the input is very simple. It is worth noting that the adder has also been optimized: Since the input corresponding to parameter "0" is truncated, the number of input ports of some adders is reduced. There is also a special case: when the input port of the adder is not connected to any input bit, the adder can be removed directly (e.g., the adder at the position of the red box in Fig. 4), and the corresponding register is used only as a shift register participating in the operations.

The above properties of REC can greatly simplify the encoder structure and reduce the complexity of its implementation. An adder in REC, if not optimized, must theoretically add $t+1$ terms (including $t$ input bits and 1 cyclic shift result). A large $t$ makes the summation logic extremely complicated, leading to the problem of difficult timing closure, so that the maximum operating frequency and throughput are reduced. After optimization, the number of input ports of the adder of REC depends only on the number of non-zero elements of the

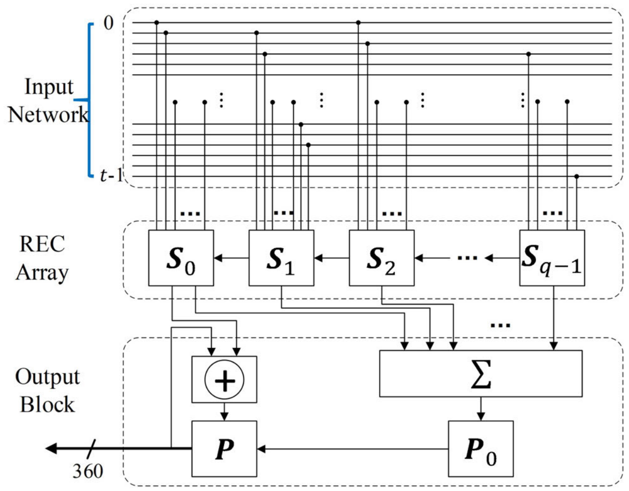


**FIGURE 5.** Overall architecture of LDPC Encoder based on algorithm 1 and 2.

parameter $c_{j,m}$, and the sparsity of the matrix $C^T$ (inherited from matrix $A$) makes the non-zero elements in parameter $c_{j,m}$ also extremely rare. Next, we take the worst case in the standards as an example to illustrate it.

In DVB-S2/S2X standards, $t$ reaches the maximum value of 162 in the case that frame length is 64800 and code rate is 9/10. At this time $q = 18$, i.e., 18 RECs participate in the encoding operation. According to the appendix of the standards, we obtained all $c_{j,m}$ parameters, and sorted out the number of input bits associated with each adder. The results are shown in Table 1.

As can be seen from Table 1: most adders are not associated with any input bit and can be removed directly. The remaining adders are mostly connected to 1 or 2 input bits, while the number of adders connected to the highest 9 input bits is only 18. Therefore, within the REC, most registers can be simplified to shift registers, and a few adders only need to perform an XOR operation of no more than 10 bits. These features make REC very conducive to hardware implementation.

### B. OVERALL ARCHITECTURE

Fig. 5 shows the overall architecture of the encoder, which is mainly composed of three parts. The first part is a parallel $t$-bit input network, which inputs each sub-information sequence $i_m = [i_{360m}, i_{360m+1}, \ldots, i_{360m+359}]$ bit by bit in reverse order. The second part is an array composed of $q$ RECs, which is the core component of algorithm 1. The array and the input network are sparsely connected, and the computation of $S$ is completed within 360 clock cycles. The bottom output block corresponds to Algorithm 2: after all $S_j$ calculations are completed, an initial result obtained by summation is stored in the $P_0$ register group. In the subsequent accumulation stage, we move each $S_j$ in turn to the position of $S_0$ by a simple shift operation to avoid the complex selection logic between different $S_j$. Using $P_0$ and each $S_j$, the register group $P$ accumulates and outputs all parity-check bits in $q$ clock cycles, and the encoding process is finished.

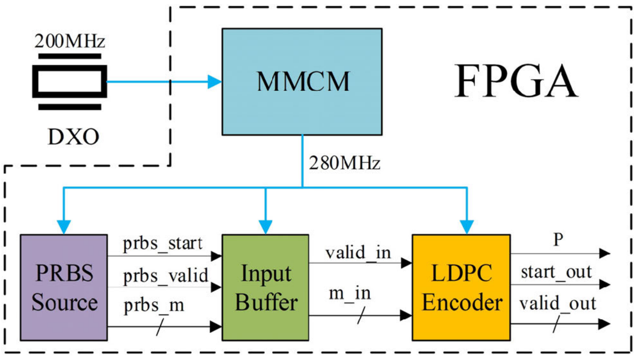


**FIGURE 6.** Hardware test framework.

## V. IMPLEMENTATION RESULT

### A. FUNCTIONAL VERIFICATION AND HARDWARE IMPLEMENTATION

In order to validate the function and performance of the designed encoder, we performed the hardware test as shown in Fig. 6: an external differential crystal oscillator (DXO) supplies the FPGA with a 200 MHz source clock, and the Mixed Mode Clock Manager (MMCM) generates a 280 MHz clock for each module. Pseudo-Noise (PN) code is stored in the "PRBS Source" module as the Pseudo-Random Binary Sequence (PRBS) to be encoded, and starts to be output continuously after power-on. The "Input Buffer" module consists of Block RAM and simple control logic. It buffers the information sequence and outputs it when the encoder is idle, which plays the role of rate matching between "PRBS Source" and "LDPC Encoder". The encoder receives the information sequence and performs encoding.

The Integrated Logic Analyzer (ILA) is an IP core used to monitor the internal signals of a design, and we use it to capture the corresponding signals. Fig. 7 shows the test result of a case. In this case, frame length is 64800 and code rate is 1/4, which corresponds to $t = 45$ and $q = 135$. The management between multiple codewords is shown in Fig. 7(a): When codeword 1 is input, the encoder is idle, so the encoding of codeword 1 is started directly; codeword 2 is input immediately after codeword 1, but codeword 1 has not finished encoding at this time, the "Input Buffer" module buffers codeword 2, and codeword 2 will not be provided to the encoder until the valid_out signal corresponding to the end of encoding is pulled down; codeword 3 and codeword 4 processed similarly to the above process. The entire encoding process of codeword 1 is shown in Fig. 7(b): the information sequence m_in is input for 360 clock cycles, which lasts from position 3002 to position 3362 on the time axis, and S is continuously updated during this period. After the input ends, P0 sums S in 4 clock cycles, then P starts accumulating, and the valid_out signal is asserted, indicating that the output parity-check bits are valid. This output period lasts $q = 135$ clock cycles from position 3366 to position 3501 on the time axis, followed by the encoding of codeword 2.

The generator polynomial of PN code used in the above test is "$X^{15} + X^{14} + 1$", with initial states [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0]. The length of the generated sequence

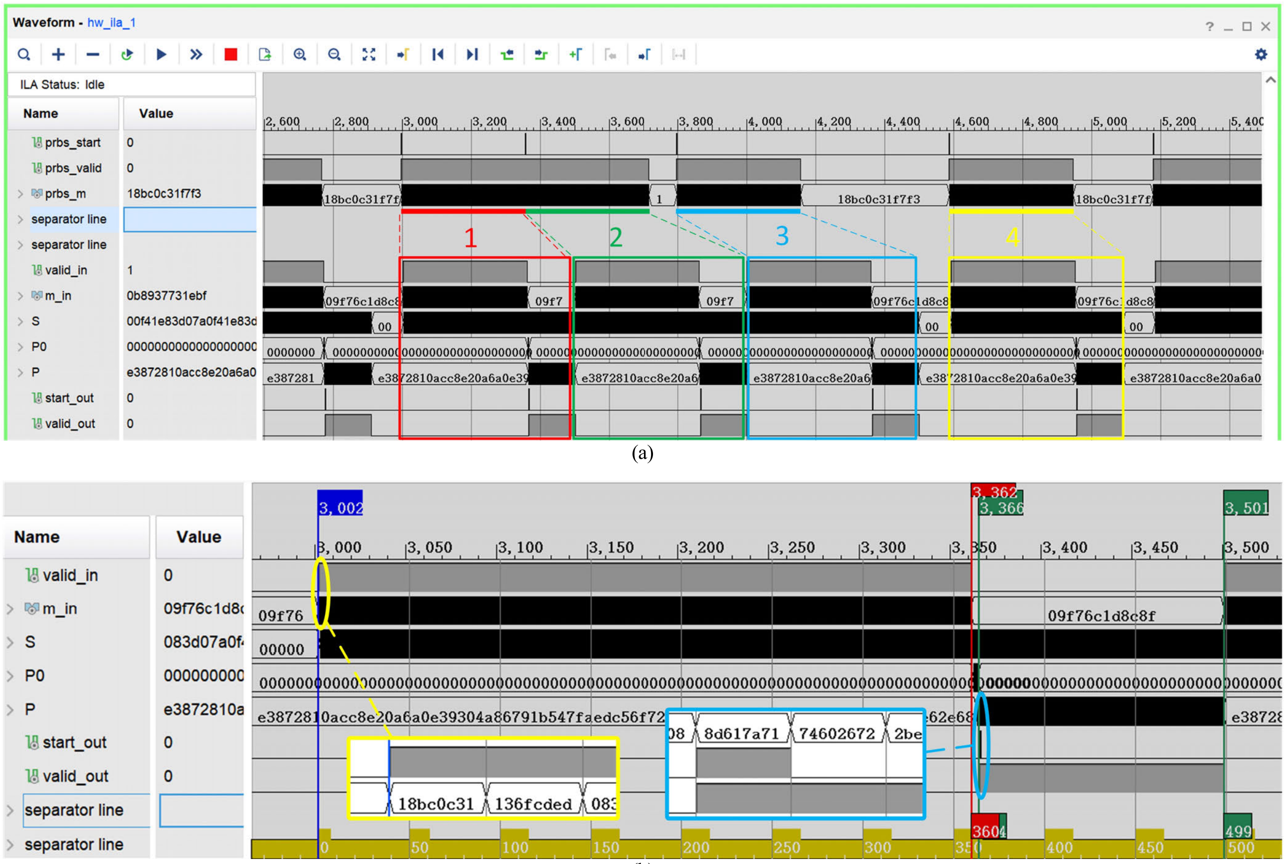


**FIGURE 7.** Hardware test result with frame length of 64800 and code rate of 1/4. (a) Management of multi-codeword encoding; (b) expanded view of codeword 1 encoding process.

to be encoded is 16200, and the first input data in 45-bit parallel format is "18BC0C31F7F3", which corresponds to the initial data marked by the yellow box in Fig. 7(b). The blue box also marks the first 360-bit result of the parity-check bits, namely "8D617A71……". After capturing the parity-check bits, we compared them with the encoding results of MATLAB. The comparison result shows that the two match, indicating that the encoder function is correct.

The encoding process in Fig. 7(b) also shows that the number of clock cycles for a single encoding process of this encoder is 360+$q$+4, of which 360 cycles are used to compute S, $q$ cycles are used to compute the parity-check bits, and the other 4 cycles are used for the initialization of S/P and the summation of P0. This means that the throughput of the encoder, denoted as $T$, is actually determined by the following formula:

$$T = \frac{N \times f_{clk}}{360 + q + 4}, \tag{23}$$

where $f_{clk}$ and $N$ represent the operating clock frequency of the encoder and frame length, respectively.

Then consider the hardware resources consumed by the encoder. According to the overall architecture described in the previous section, the main resource consumption is $q + 2$ register groups, each containing 360 registers. In other words: the smaller the $q$ is, the lower the resource consumption will be, and vice versa.

The DVB-S2/S2X standards include LDPC codes with 3 types of frame length and 48 types of code rate. The $q$ ranges from the lowest 5 to the highest 140. We implemented the LDPC encoders with different frame length and different code rate on Xilinx FPGA device XC7K325T.

Table 2 shows the implementation results for the best ($q = 5$) and worst ($q = 140$) cases. Based on previous estimates, the worst and best cases should theoretically consume $(140 + 2) \times 360 = 51120$ and $(5 + 2) \times 360 = 2520$ registers

**TABLE 2.** Implementation results on Xilinx FPGA XC7K325T.

| | | Worst Case | Best Case |
|---|---|---|---|
| Parameters | Standard | DVB-S2X | DVB-S2 |
| | $N$ | 64800 | 16200 |
| | Code Rate | 2/9 | 8/9 |
| | $q$ | 140 | 5 |
| Resources Report | LUTs | 37542 | 3202 |
| | Flip-Flops | 51139 | 2540 |
| | BRAMs | 1 | 1 |
| Timing Report | Clock Period (ns) | 3.57 | 3 |
| | Clock Frequency (MHz) | >280 | >333 |
| | Worst Negative Slack (ns) | 0.033 | 0.206 |
| | Worst Hold Slack (ns) | 0.075 | 0.111 |

**TABLE 3.** Encoder throughput performance at 280 MHz clock frequency.

| Normal FECFRAME($N$=64800) | | | | | |
|---|---|---|---|---|---|
| Code Rate | $q$ | Throughput (Gbps) | Code Rate | $q$ | Throughput (Gbps) |
| 9/10 | 18 | 47.50 | 23/36 | 65 | 42.29 |
| 8/9 | 20 | 47.25 | 28/45 | 68 | 42.00 |
| 77/90 | 26 | 46.52 | 3/5 | 72 | 41.61 |
| 5/6 | 30 | 46.05 | 26/45 | 76 | 41.24 |
| 4/5 | 36 | 45.36 | 5/9 | 80 | 40.86 |
| 7/9 | 40 | 44.91 | 11/20 | 81 | 40.77 |
| 3/4 | 45 | 44.36 | 24/45 | 84 | 40.50 |
| 22/30 | 48 | 44.04 | 1/2 | 90 | 39.96 |
| 13/18 | 50 | 43.83 | 9/20 | 99 | 39.19 |
| 32/45 | 52 | 43.62 | 2/5 | 108 | 38.44 |
| 25/36 | 55 | 43.30 | 1/3 | 120 | 37.49 |
| 31/45 | 56 | 43.20 | 13/45 | 128 | 36.88 |
| 2/3 | 60 | 42.79 | 1/4 | 135 | 36.36 |
| 29/45 | 64 | 42.39 | 2/9 | 140 | 36.00 |
| Medium FECFRAME($N$=32400) | | | | | |
| Code Rate | $q$ | Throughput (Gbps) | Code Rate | $q$ | Throughput (Gbps) |
| 1/3 | 60 | 21.40 | 1/5 | 72 | 20.81 |
| 11/45 | 68 | 21.00 | | | |
| Short FECFRAME($N$=16200) | | | | | |
| Code Rate | $q$ | Throughput (Gbps) | Code Rate | $q$ | Throughput (Gbps) |
| 8/9 | 5 | 12.29 | 7/15 | 24 | 11.69 |
| 37/45 | 8 | 12.19 | 4/9 | 25 | 11.66 |
| 7/9 | 10 | 12.13 | 2/5 | 27 | 11.60 |
| 11/15 | 12 | 12.06 | 1/3 | 30 | 11.51 |
| 32/45 | 13 | 12.03 | 14/45 | 31 | 11.48 |
| 2/3 | 15 | 11.97 | 4/15 | 33 | 11.43 |
| 3/5 | 18 | 11.87 | 11/45 | 34 | 11.40 |
| 26/45 | 19 | 11.84 | 1/5 | 36 | 11.34 |
| 8/15 | 21 | 11.78 | | | |

**TABLE 4.** Performance comparison of encoders with frame length of 64800 and code rate of 1/2.

| Work | Encoding Cycles | Clock Frequency (MHz) |
|---|---|---|
| [19] | 1080 | 131.7 |
| [20] | >630 | 100 |
| Proposed | 454 | 280 |

respectively, which are basically consistent with the results in the resources report, indicating that the implementation results meet the design expectations. Then pay attention to the timing report: even in the worst case, the operating clock frequency of the encoder can reach 280 MHz, and in the best case, it can exceed 330 MHz.

Therefore, we can uniformly use $f_{clk} = 280$ MHz to measure the maximum throughput of the encoder. According to (23), the throughput performance of the encoder with different parameters is given in Table 3.

As shown in the table, in the case of Normal FECFRAME (frame length 64800), the proposed encoder has a throughput of over 36 Gbps at various code rates, and can reach up to 47.5 Gbps when the code rate is the highest 9/10. For the other two frame lengths, the throughput of the encoder decreases somewhat with shorter frame length, but a throughput of 11.34 Gbps can be achieved even with the lowest frame length.

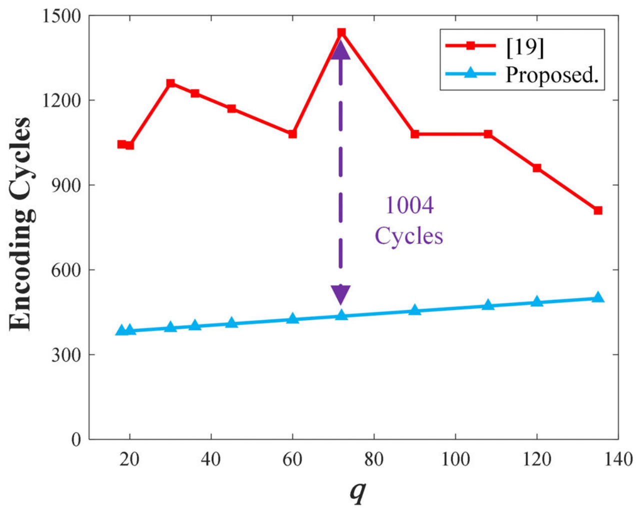


**FIGURE 8.** Comparison of encoding cycles with frame length of 64800.

## B. PERFORMANCE AND EFFICIENCY ANALYSIS

According to (23), the throughput performance of the encoder mainly depends on two factors: first, the number of clock cycles required for a single encoding process, and second, the operating clock frequency of the encoder.

Table 4 shows the comparison result of three encoders with different architectures when the frame length is 64800 and the code rate is 1/2. The proposed encoder is superior to the encoders of [19] and [20] in both aspects: not only the number of encoding cycles is lower, but also the operating clock frequency is also higher.

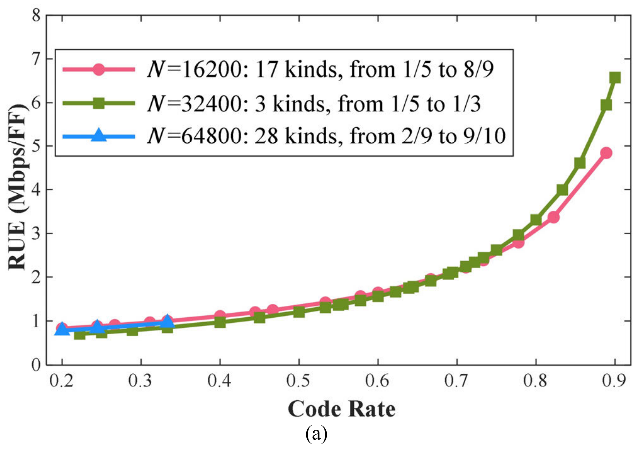
Besides, the number of encoding cycles at different code rates are also given in [19], so we compare them with the results of this work, as shown in Fig. 8. It can be seen that the number of encoding cycles of the proposed encoder only increases linearly with $q$ and is lower than the result of [13] at all code rates, which means that the proposed coder has higher throughput performance even at the same operating clock frequency. At a code rate of 3/5 ($q = 72$), the difference between the two reaches the maximum value of 1004 cycles, and the difference in throughput is more than 3 times.

In addition, the throughput and occupied resources of the proposed encoder architecture are different at different code rates. To make a better and fair comparison between encoders with different code rates, we use the *Resource Utilization Efficiency* (RUE) to measure the comprehensive performance of encoders, which is expressed as the obtained throughput divided with the number of used resources:

$$RUE = \frac{T}{Resources}. \tag{24}$$

The RUE represents the average throughput carried by a unit of resource. For encoders with different code rates, the higher the value, the higher the throughput that can be achieved with the same amount of hardware resources.

In DVB-S2/S2X Standards, there are 17 kinds of code rates when the frame length is 16200, 3 kinds when the frame length is 32400, and 28 kinds when the frame length is 64800.



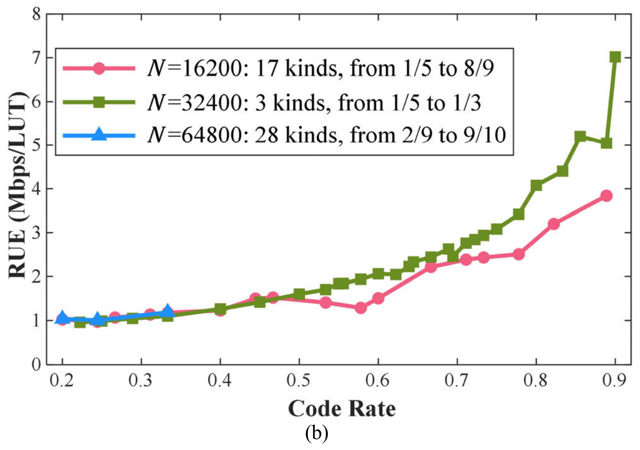


**FIGURE 9. Relationship curve between RUE and code rate for (a) Flip-flop and (b) Look-up table.**

We obtain the relationship curve between RUE and code rate of the proposed encoder architecture under three different frame lengths, as shown in Fig. 9.

It can be seen that at fixed frame length with increasing code rate increases, the RUE of FF shows an exponential growth trend, while the RUE of LUT also increases, although there are some fluctuations. The reason for the fluctuations is that the number and distribution of non-zero elements in the parameter $\boldsymbol{c}_{j,m}$ vary greatly at different code rates, which causes the amount of LUT consumed during implementation does not grow linearly like FF.

On the whole, the RUE of LUT and FF basically increases with the increase of the code rate, which is mainly due to the two effects caused by the decrease of $q$: On the one hand, the number of encoding cycles is reduced, increasing the throughput; and on the other hand, the resource usage is reduced. The numerator of (24) is increased and the denominator is decreased, which significantly improves the RUE.

In summary, we recommend that the encoder architecture proposed in this paper be used in scenarios with longer frame length and higher code rate, so that higher throughput performance than other architectures can be achieved with lower resource consumption.

## VI. CONCLUSION

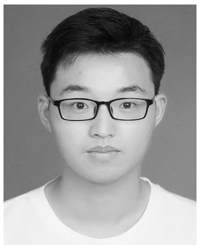
In this paper, we present an encoder architecture suitable for LDPC code in DVB-S2/S2X standards. By exploring and reshaping the parity-check matrix, a kind of component called recursive encoding core is used for fast computations in the encoding process. Compared to other methods, this architecture significantly reduces the encoding time in the case of longer frame length, thereby achieving extremely high throughput performance. The implementation results on FPGA also show that the encoder architecture has higher resource utilization efficiency at higher code rates and can achieve a higher performance index with fewer resources than in lower code rate scenarios. However, the input of the proposed encoder often comes from BCH encoder, and how to effectively convert its output into the required $t$-bit parallel format needs to be further explored, and this is also the problem we want to solve in the next step.

## REFERENCES

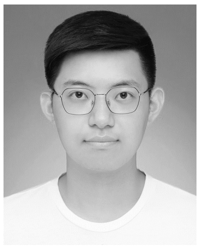
[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962, doi: 10.1109/TIT.1962.1057683.
[2] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001, doi: 10.1109/18.910579.
[3] D. Theodoropoulos, N. Kranitis, A. Tsigkanos, and A. Paschalis, "Efficient architectures for multigigabit CCSDS LDPC encoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 5, pp. 1118–1127, May 2020, doi: 10.1109/TVLSI.2020.2975050.
[4] A. E. Cohen and K. K. Parhi, "A low-complexity hybrid LDPC code encoder for IEEE 802.3an (10GBase-T) Ethernet," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 4085–4094, Oct. 2009.
[5] *TM Synchronization and Channel Coding*, Standard 131.0-B-3, CCSDS, Sep. 2017.
[6] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 53, no. 11, p. 1973, Nov. 2005.
[7] K. Andrews, S. Dolinar, and J. Thorpe, "Encoders for block-circulant LDPC codes," in *Proc. Int. Symp. Inf. Theory (ISIT)*, Sep. 2005, pp. 2300–2304.
[8] D. Theodoropoulos, N. Kranitis, and A. Paschalis, "An efficient LDPC encoder architecture for space applications," in *Proc. IOLTS*, Sant Feliu de Guixols, Spain, 2016, pp. 149–154.
[9] J. Liu and Q. Feng, "A miniaturized LDPC encoder: Two-layer architecture for CCSDS near-Earth standard," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 7, pp. 2384–2388, Jul. 2021, doi: 10.1109/TCSII.2021.3054334.
[10] *Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications; Part 1: DVB-S2*, Standard ETSI EN 302 307-1 V1.4.1, 2014.
[11] *Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications; Part 2: DVB-S2 Extensions (DVB-S2X)*, Standard ETSI EN 302 307-2 V1.2.1, 2020.
[12] G. Y. Mihaylov, T. B. Iliev, E. P. Ivanova, I. S. Stoyanov, and L. Iliev, "Performance analysis of low density parity check codes implemented in second generations of digital video broadcasting standards," *Proc. 39th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2016, pp. 499–502, doi: 10.1109/MIPRO.2016.7522195.
[13] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001, doi: 10.1109/18.910578.
[14] 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, doi: 10.1109/ACCESS.2020.3007923.

[15] A. V. Lazarenko, "FPGA design and implementation of DVB-S2/S2X LDPC encoder," in *Proc. IEEE Int. Conf. Electr. Eng. Photon. (EExPolytech)*, Oct. 2019, pp. 98–102, doi: 10.1109/EexPolytech.2019.8906811.
[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] 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.
[18] N. Kumar, C. Prakash, S. N. Satashia, V. Kumar, and K. S. Parikh, "Efficient implementation of low density parity check codes for satellite ground terminals," in *Proc. Conf. ICACCI, New Delhi, India, Sep. 2014*, pp. 689–695.
[19] M. Gomes, G. Falcao, A. Sengo, V. Ferreira, V. Silva, and M. Falcao, "High throughput encoder architecture for DVB-S2 LDPC-IRA codes," in *Proc. Internatonal Conf. Microelectron.*, Cairo, Egypt, Dec. 2007, pp. 271–274.
[20] I. Lee, M. Kim, D. Oh, and J. Jung, "High-speed LDPC encoder architecture for digital video broadcasting systems," in *Proc. Int. Conf. ICT Converg. (ICTC)*, Jeju, South Korea, Oct. 2013, pp. 606–607.
[21] Y. Jung, C. Chung, J. Kim, and Y. Jung, "7.7 Gbps encoder design for IEEE 802.11n/AC QC-LDPC codes," in *Proc. Int. SoC Design Conf. (ISOCC)*, Nov. 2012, pp. 215–218, doi: 10.1109/ISOCC.2012.6407078.

**DECAI LIU** received the B.S. degree in communication engineering from Shanghai University, Shanghai, China, in 2020, where he is currently pursuing the master's degree. His research interests include digital signal processing, channel coding, and FPGA implementation.



**YANFEI LUO** received the B.S. degree in communication engineering from Shanghai University, Shanghai, China, in 2020, where he is currently pursuing the master's degree. His research interests include digital signal processing, receiver synchronization, channel equalization, and FPGA implementation.



**YUNFENG LI** received the B.S. degree in communication engineering from Shanghai University, Shanghai, China, in 2019, where he is currently pursuing the master's degree in communication and information system. His research interests include LDPC decoder and FPGA implementation.

**ZHIJIE WANG** received the B.S. degree in communication engineering from Shanghai University, Shanghai, China, in 2021, where he is currently pursuing the M.S. degree in communication and information system. His current research interests include digital signal processing, satellite communication, and FPGA.

**ZHENGXUAN LI** received the Ph.D. degree in communication and information engineering from Shanghai Jiaotong University, in 2016. She is currently an Associate Professor with the School of Communication and Information Engineering, Shanghai University. Her primary research interests include broadband optical access and digital signal processing in short reach optical interconnects.

**QIANWU ZHANG** received the Ph.D. degree in communication and information engineering from Shanghai University, in 2015. Since 2015, he has been an Associate Professor with Shanghai University. His research interests include optical amplifiers, optical transceivers, optical and electrical signal processing, high-speed optical signal transmission in long-haul, metropolitan, and access networks.

**JUNJIE ZHANG** received the Ph.D. degree in physical electronics from USTC, Hefei, China, in 2005. He is currently a Professor with the School of Communications and Engineering, Shanghai University, Shanghai, China. His primary research interests include channel coding, satellite communications, and optical communications.

**YINGCHUN LI** received the B.S. degree from the Shanghai University of Science and Technology, Shanghai, China, in 1984, and the Ph.D. degree from Shanghai University, Shanghai, in 2008.

From 1984 to 1999, he worked with the Advanced Communication Group, Shanghai University of Science and Technology, where he is engaged in optical fiber and digital communication. Since 2000, he has been working with the Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Shanghai University, and since 2010, he has been a Professor with the School of Communication and Information, Shanghai University. His current research interests include free space optical and radio over fiber communications.

• • •