```
Initialize S_c = 0
for m = 0 to (t − 1) do
    for each r ∈ CN(c = 360m) do
        x = ⌊r/q⌋
        Read old s_c value from s_c register
        s_c = s_c + i_m^(x)
        Write new s_c value to s_c register
    end for
end for
```

Moving on, let $L_{L\times L}$ be a lower triangular matrix of ones:

$$L=\begin{bmatrix}1&0&0&\dots&0&0\\1&1&0&\dots&0&0\\1&1&1&\dots&0&0\\\vdots&\vdots&\vdots&\ddots&\vdots&\vdots\\1&1&1&\dots&1&0\\1&1&1&\dots&1&1\end{bmatrix}_{L\times L}\tag{22}$$

By applying the linear transformation specified by $\mathbf{L}$ to the checksum vector, we obtain the following result:

$$L\cdot s_c^T = \left[\sum_{i=0}^{q-1}s_i\ \sum_{i=0}^{2q-1}s_i\ \sum_{i=0}^{3q-1}s_i\cdots\sum_{i=0}^{360q-1}s_i\right]^T\tag{23}$$

Next, the vector s_c is logically shifted left of one bit to obtain the following vector, referred to as the parity initialization vector:

$$p_{init}=\left[0\ \sum_{i=0}^{q-1}s_i\ \sum_{i=0}^{2q-1}s_i\cdots\sum_{i=0}^{359q-1}s_i\right] = \left[0\ p_{q-1}\ p_{2q-1}\cdots p_{359q-1}\right]\tag{24}$$

In this identity, we have applied Eq. 14. Furthermore, we notice that p_{init} can readily be obtained from the checksum vector s_c using a simple combinatorial lesson. Finally, we can now calculate L parity bits at a time by using the following procedure:

$$\begin{cases}[p_0p_q\cdots p_{359q}]=\\=[0p_{(q-1)}\cdots p_{(359q-1)}]+[s_0s_q\cdots s_{359q}]\\ [p_1p_{(q+1)}\cdots p_{(359q+1)}]=\\=[p_0p_q\cdots p_{359q}]+[s_1s_{(1+q)}\cdots s_{(1+359q)}]\\ [p_2p_{(q+2)}\cdots p_{(359q+2)}]=\\=[p_1p_{(q+1)}\cdots p_{(359q+1)}]+[s_2s_{(2+q)}\cdots s_{(2+359q)}]\\ \vdots\\ [p_{(q-1)}p_{(2q-1)}\cdots p_{(n-k-1)}]=\\=[p_{(q-2)}p_{(2q-2)}\cdots p_{(n-k-2)}]+[s_{(q-1)}s_{(2q-1)}\cdots s_{(n-k-1)}]\end{cases}\tag{25}$$

Similarly to S_j, we define P_j as:

$$P_j=\left[p_j\ p_{j+q}\ p_{j+2q}\cdots p_{j+359q}\right]\tag{26}$$

Which can be rewritten recursively as:

$$\begin{cases}P_0=p_{init}=[0p_{q-1}p_{2q-1}\cdots p_{359q-1}]\\P_j=S_j+P_{j-1}\end{cases}\tag{27}$$

Similarly to S_M, we shall define a $q\times L$ matrix containing the parity bits as follows:

$$P_M=\begin{bmatrix}P_0\\P_1\\P_2\\\vdots\\P_{q-1}\end{bmatrix}=\begin{bmatrix}p_0&p_q&p_{2q}&\cdots&p_{359q}\\p_1&p_{q+1}&p_{2q+1}&\cdots&p_{359q+1}\\p_2&p_{q+2}&p_{2q+2}&\cdots&p_{359q+2}\\\vdots&\vdots&\vdots&\ddots&\vdots\\p_{q-1}&p_{2q-1}&p_{3q-1}&\cdots&p_{N-K-1}\end{bmatrix}_{q\times M}\tag{28}$$

Finally, we can define the computation algorithm of the P_j.

```
Initialize P_j = p_initi
for j = 0 to (q − 1) do
    Read S_j from memory at address j
    P_j = S_j + P_{j−1}
    Write P_j to memory at address j
end for
```

Two crucial observations must be made. The first one is that p_{init}, being obtained from the checksum vector s_c, can be computed only after all S_j have been calculated. Secondly, this algorithm computes L=360 parity check bits at a time, which is desirable for high throughput architectures; however, these bits are not output in the natural order. Therefore, a reordering process is necessary. Although this is an important topic, much of the relevant literature underrates this problem: we shall propose a solution to this issue further on. Finally, other 2× q cycles are needed for accumulation, for $2\times(W+q)$ under the assumption that Simple One Port RAM is utilized.

D. VECTORIZED QUASI-CYCLIC ENCODING

We shall now present an alternative approach to the encoding algorithm based on performing a row permutation on $\mathbf{A}$. Extract a $q\times k$ matrix, denoted as A'_r, from $\mathbf{A}$, with $r\in 0,\ 1,\ \dots q-1$

$$A'_r=\begin{bmatrix}a_{r,0}&a_{r,1}&\cdots&a_{r,k-1}\\a_{r+q,0}&a_{r+q,1}&\cdots&a_{r+q,k-1}\\\vdots&\vdots&\ddots&\vdots\\a_{r+359q,0}&a_{r+359q,1}&\cdots&a_{r+359q,k-1}\end{bmatrix}\tag{29}$$

Reorganize the A'_r submatrices into $\mathbf{C}$. $\mathbf{C}$ is a row-wise permutation of $\mathbf{A}$.

$$C=\begin{bmatrix}A_0{}'\\A'_1\\\vdots\\A'_{q-1}\end{bmatrix}\tag{30}$$

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

$$C=\begin{bmatrix}C_{0,0}&C_{0,1}&\cdots&C_{0,t-1}\\C_{1,0}&C_{1,1}&\cdots&C_{1,t-1}\\\vdots&\vdots&\ddots&\vdots\\C_{q-1,0}&C_{q-1,1}&\cdots&C_{q-1,t-1}\end{bmatrix}\tag{31}$$

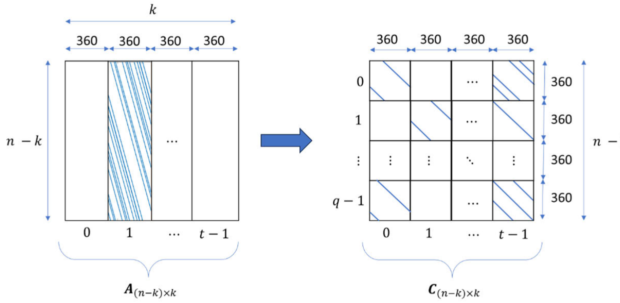

FIGURE 4. Reshaping of A matrix.

Each submatrix $C_{j,m}$ is a square $L\times L$, i.e. 360×360, cyclic matrix (with $j\in\{0,\ 1,\ \dots q-1)$, $m\in\{0,\ 1,\ \dots t-1\}$), that is each row (or column) is the right logic rotation of the previous row (or column). The reshaping operation of the **A** matrix into a quasi-cyclic matrix **C** is represented in Figure 4:

The accumulation vector $S_{1\times(n-k)}$ in Eq. 10 is consequently rearranged as $S\prime_{1\times(n-k)}$:

$$S\prime_{1\times(N-K)}=\left[S_0,\ S_1,\ \dots,\ S_{q-1}\right] \tag{32}$$

where each S_j is a L=360 bits vector, defined as:

$$\begin{aligned} S_j&=[s_j, s_{j+q}, s_{j+2q}, \dots s_{j+359q}]\\ &\quad withj\ \in 0,\ 1,\ 2,\ \dots q-1 \end{aligned} \tag{33}$$

We shall also divide the input message $i=[i_0,\ i_1,\ \dots i_m,\ \dots i_{t-1}]$ into t groups comprised of consecutive 360 input bits, denoted by i_m. Eq. 10 can be rewritten as:

$$\begin{aligned} S'_{1\times(N-K)}&=i\cdot C^T\\ &=[i_0,\ i_1,\ \dots,i_{t-1}]\\ &\quad\cdot\begin{bmatrix} C^T_{0,0} & C^T_{1,0} & \cdots & C^T_{q-1,0}\\ C^T_{0,1} & C^T_{1,1} & \cdots & C^T_{q-1,1}\\ \vdots & \vdots & \ddots & \vdots\\ C^T_{0,t-1} & C^T_{1,t-1} & \cdots & C^T_{q-1,t-1}\end{bmatrix} \end{aligned} \tag{34}$$

Thus, we deduce that each S_j can be computed as:

$$\begin{aligned} S_j&=\left[s_j, s_{j+q}, s_{j+2q}, \dots s_{j+359q}\right]\\ &=[i_0,\ i_1,\ \dots,i_{t-1}]\cdot\begin{bmatrix} C^T_{j,0}\\ C^T_{j,1}\\ \vdots\\ C^T_{j,t-1}\end{bmatrix} \end{aligned} \tag{35}$$

If the entire input message $i=[i_0,\ i_1,\ \dots i_m,\ \dots i_{t-1}]$ is stored in a memory, then it is possible to compute 360 bits in parallel and store them in a $q\times 360$ RAM as follows:

$$S_M=\begin{bmatrix} S_0\\ S_1\\ \vdots\\ S_{q-1}\end{bmatrix}=\begin{bmatrix} s_0 & s_q & \cdots & s_{359q}\\ s_1 & s_{1+q} & \cdots & s_{359q+1}\\ \vdots & \vdots & \ddots & \vdots\\ s_{q-1} & s_{2q-1} & \cdots & s_{360q-1}\end{bmatrix} \tag{36}$$

Furthermore, it can be shown that the computation of the S_j a simple task because of the sparseness of the matrix **C**, inherited by the parity check matrix, and because of the quasi-cyclic structure of the square sub-matrices $C_{i,j}$. By expanding Eq. 35 we get:

$$S_j=i_0\cdot C^T_{j,0}+i_1\cdot C^T_{j,1}+\dots+i_{t-1}\cdot C^T_{j,t-1}=\sum_{m=0}^{t-1}i_m\cdot C^T_{j,m} \tag{37}$$

The first observation that can be made is that not all t transformations of the i_m vectors need to be executed, as the vast majority of the $C_{j,m}$ submatrices have all zero elements. Secondly, it is trivial to compute the product $i_m\cdot C_{j,m}$. Let us consider a quasi-cyclic $L\times L$ submatrix **D** and let $\alpha\ \in 0,\ 1,\cdots,L-1$ be the index of the only non-zero element in the first row of **D**. For example, if $\alpha=0$, then **D** is simply an $L\times L$ identity matrix **I**. The rows of **D** are L-vectors over GF(2) denoted by $[d_0, d_1, d_2,\cdots, d_{L-1}]$. Let **u** and **v** be two L-vectors over GF(2) and consider the following identity:

$$D\cdot u^T=v^T \tag{38}$$

This equation can be expressed equivalently by the system of equations:

$$\begin{cases} d_0\cdot u^T=v_0\\ d_1\cdot u^T=v_1\\ \vdots\\ d_{L-1}\cdot u^T=v_{L-1}\end{cases} \tag{39}$$

Since the rows of **D** are rotated replicas of the first row, we can indicate a logical left rotation by the bracketed apex $x^{(\ )}$ and rewrite the equation above as follows:

$$\begin{cases} d_0\cdot u^T=v_0\\ d_0^{(1)}\cdot u^T=v_1\\ \vdots\\ d_0^{(L-1)}\cdot u^T=v_{L-1}\end{cases} \tag{40}$$

Since d_0 only has one non-zero element in position α, then $d_0\cdot u^T=v_0=u_\alpha$. Consequently, it is clear that $d_0^{(1)}\cdot u^T=v_1=u_{|\alpha+1|_L}$ where the modulo-L is a mathematical description of the fact that the circular non-zero element in **D** rotates over to the 0^{th} position after it has reached the $(L-1)^{th}$ position. By repeating the same process for all the other terms, we obtain the following result:

$$v^T=D\cdot u^T=\begin{bmatrix} u_\alpha\\ u_{|\alpha+1|_L}\\ u_{|\alpha+2|_L}\\ \vdots\\ u_{|\alpha+359|_L}\end{bmatrix}\Rightarrow v=u^{(-\alpha)} \tag{41}$$

In simpler terms, Eq. 41 means that the result of the multiplication between an L-vector **u** and a quasi-cyclic matrix **D** is an L-vector **v** that is the logical proper rotation of **u** by α, i.e. the position of the only non-zero bit in the first row of **D**. Returning to the discussion of Eq. 3.34, we note

TABLE 5. Index-Alfa Table (frame length = 16200, code rate =1/2).

m	α	m	α	m	α	m	α	m	α
5	0	9	26	19	222	-	-	-	-
1	125	2	132	2	323	6	0	-	-
3	217	3	248	4	112	7	0	14	45
1	107	4	280	8	0	17	239	-	-
0	106	9	0	-	-	-	-	-	-
6	246	10	0	13	237	-	-	-	-
11	0	13	176	-	-	-	-	-	-
2	220	12	0	18	318	-	-	-	-
0	154	8	314	13	0	14	175	-	-
5	83	14	0	15	205	-	-	-	-
4	313	15	0	16	3	-	-	-	-
0	198	0	265	16	0	19	64	-	-
0	318	0	332	7	352	17	0	-	-
2	263	4	310	18	0	18	121	-	-
1	237	8	223	17	330	19	0	-	-
2	233	4	155	10	349	-	-	-	-
3	317	6	358	-	-	-	-	-	-
3	174	4	171	11	302	12	271	-	-
1	259	2	213	15	86	-	-	-	-
2	350	7	93	-	-	-	-	-	-
0	0	0	159	3	180	12	48	-	-
1	168	2	0	4	101	9	184	-	-
1	131	1	267	3	0	-	-	-	-
3	148	3	183	4	0	10	124	11	199

that some of the non-zero $C_{j,m}$ are, in fact, quasi-cyclic $L \times L$ submatrix with only one circulating '1', whereas others may have more than one circulating '1'. In the first case, all we need to know to compute the product $i_m \cdot C_{j,m}$ is m, i.e. the index identifying what group of 360 input bits is currently selected, and the α value relative to $C_{j,m}$. Thus, we have:

$$i_m \cdot C_{j,m} = i_m^{(-\alpha)} \tag{42}$$

In the latter case, we can keep the index value m constant and express $C_{j,m}$ as a sum of multiple quasi-cyclic $L \times L$ submatrix with only one circulating '1', each characterized by a distinct α value. For example, if there are three circulating '1's, we shall define three αvalues, i.e. α_1, α_2, α_3, and compute $i_m \cdot C_{j,m}$ as follows:

$$i_m \cdot C_{j,m} = i_m^{(-\alpha_1)} + i_m^{(-\alpha_2)} + i_m^{(-\alpha_3)} \tag{43}$$

Each S_j can be computed by an accumulation of 360-bit entries selected from the input message, i.e. i_m, rotated by an amount α. Therefore, we must extrapolate all the valid (m, α) couplets for any q rows of $L \times L$ quasi-cyclic matrices. These couplets are a complete and equivalent description of the **C** matrix. One example of a Look-Up Table (LUT) for $framelength = 16200$ and $nominalcoderate = 1/2(actualcoderate = 4/9)$ containing the (m, α) couplets are given in Table 5: it is composed of q=25 rows and a varying number of columns, which is equal to the row-weight w_r of each of the selected row of $L \times L$ quasi-cyclic matrices: in this case, w_r assumes every value between 2 and 5.

The total number of elements (with two entries) in the Index-Alfa Tables could be computed by summing all the row-weights w_r of **C**. However, there is a more straightforward and more meaningful method. Let us compare the first 360 columns of **A** with the first 360 columns of **C**, as exemplified in Figure 5 for $framelength = 16200$ and $nominalcoderate = 8/9(q = 5, \;\; n - k = 1800)$.

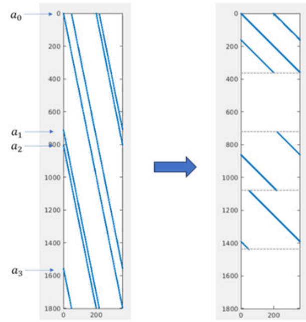

FIGURE 5. Comparing the first 360 columns of A (left) and C (right).

We can quickly identify four $L \times L$ quasi-cyclic matrices with only one rotating '1' and their (m, α) couplets. They correspond to the original four "rotating diagonals" specified by the four elements $[a_0, a_1, a_2, a_3]$ from the first row of the Standard Table. Therefore, by repeating the same observation for all the 360-column groups, we may deduce that there are as many (m, α) couplets in the Index-Alfa Table as there are elements in the Standard Tables, i.e. the Index-Alfa table contains $W(m, \alpha)$ couplets (for the W values, see Table 4). Finally, we may define the following algorithm.

Initialize $P_j = p_{initi}$
for $j = 0$ to $(q - 1)$ **do**
  Initialize $S_j = 0$
  for each $(m, \alpha) \in j - throwoftheIndex - AlfaTable$ **do**
    $S_j = S_j + i_m^{(-\alpha)}$ (now we can store S_j in j^{th} row of S_M)
  end for
end for

Finally, the second step of the algorithm, computing the parity check bits, is identical to Vectorized IRA Encoding. From a timing standpoint, this algorithm requires only W+q cycles, which is better than the previous one. We also showed that the Standard Table and the Index-Alfa tables are equivalent. This approach is more suitable for unidirectional data flow as it does not rely on a complex loop involving RAM as a computational building block. These benefits come at the cost of an increased memory footprint as the entire input message needs to be stored. As already discussed, latency also increases, but it is not an issue in practical applications.

III. REGISTER TRANSFER LEVEL DESIGN

The proposed encoder architecture introduces the following key innovations. **I) Full Compliance with DVB-S2 Standard**. The encoder will be designed to comply with the DVB-S2 Standard for all MODCODs. This will enable the utilization of ACM and VCM techniques, optimizing transmission efficiency by adapting to varying channel conditions. **II) Highly Reconfigurable I/O Interfaces**. The input and