Encoder Writeup:

Specification for Deep space

For deep space, the LDPC specification is AR4JA LDPC code. It has code size, N =1280 bits = Nb x Z, where Nb=40, Z=circulant size=32. A systematic form of G-matrix is followed for encoding. CCSDS document[1] provides the parity part of Punctured G matrix, which consists of Kb=32 block rows with each block row having (Nb-Kb)=8 circulants.

$$G = \left[\begin{array}{cccc|cccc} I_{11} & I_{12} & \dots & I_{1,32} & dm_{11} & dm_{12} & \dots & dm_{18} \\ \dots & \dots & \dots & \dots & dm_{21} & dm_{22} & \dots & dm_{28} \\ I_{32,1} & I_{32,2} & \dots & I_{32,32} & \dots & \dots & \dots & \dots \\ & & & & dm_{32,1} & dm_{32,2} & \dots & dm_{32,8} \end{array}\right]$$

Here, I is 32x32 identity matrix and dm is a 32x32 dense circulant matrix. Each dense circulant is characterized by their first rows. By cyclic shifting operation on this row, other rows can be generated.

Specifications for Near Earth:

The generator matrix (G) for Near earth standard is of size 7154 x 1022. Having 14x2 array of 511x511 circulants.

The message vector to be multiplied with G is { 18 zeros, 7136 information bits} = 7154 bits.

The generated parity bits 1022. The transmitted Code is assumed to be **8160** = {7136 Systematic, 1022 Parity, 2zeros}, so that effectively, a shortened code (8160, 7136) is transmitted, as mentioned in [1].

The parity check (H) matrix for the Near earth Decoder is for a code standard (8176, 7154). It contains Mb=2 block row or Layers with each layer consisting of block columns Nb=16 corresponding to 16 Circulants of weight 2[1]. The row weight of the H-matrix is constant at Wc=32 for all layers. The circulant size, Z=511.

At the decoder, this codeword is expanded to 8176 codeword = {18 Max Values, 7136 Systematic symbols, 1022 parity symbols}, last 2 zeros are discarded.

Architecture of encoder

Recursive Convolution Encoder (RCE) and Shift-Register-Adder-Accumulate (SRAA) are 2 types of encoder circuits, to perform multiplication of dense circulants of G-matrix and systematic bits to estimate parity bits. Compared to SRAA structure, the shifting of resultant parity bits is performed here for convolution operation, instead of shifting of circulant bits.

For the deep space LDPC, we found SRAA encoder[9] and Parallel Recursive convolution encoder (Parallel RCE) architectures[6] suitable. Out of this, parallel RCE offered better speed and flexible control over area. So we chose parallel RCE over SRAA.

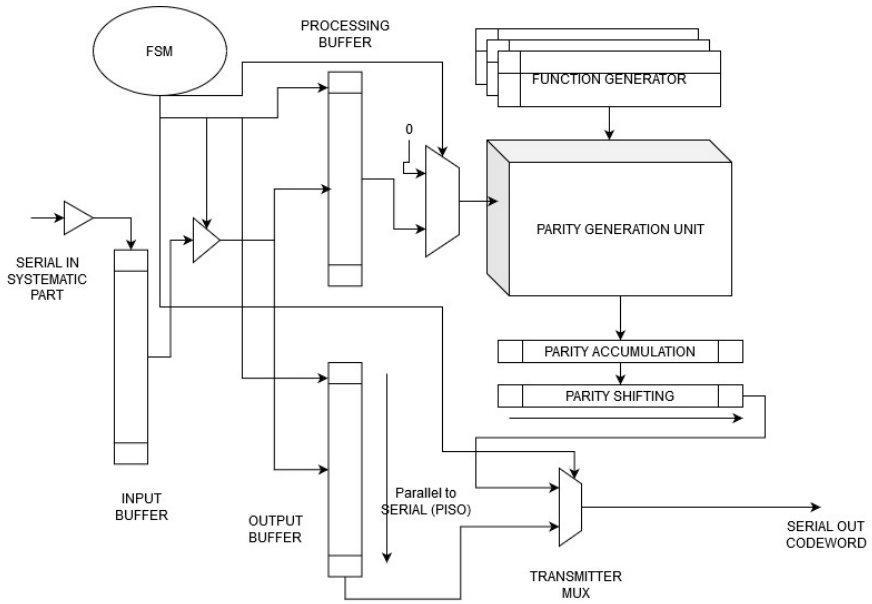


Fig. Parallel RCE architecture overall block diagram.

The parameter for performance is Lm, the systematic bits processed in parallel, and La, circulants of G-matrix processed in parallel, respectively. Based on the area and performance curves in [6], we chose La=8 and Lm=16, to attain smaller latency of 8 cycles.

In SRAA[9], by implementing all circulant rows using wiring, we can eliminate the shift register. For a comparison we had also implemented this parallel version from SRAA type encoder with Lm=Z, La=8. In this parallel version of SRAA:

- Parity accumulators is required, parity shifting operation is only used for output transmission.
- As LDPC code standard changes, Z changes, area of circuit changes. To adjust area by making Lm < Z, additional align/shift register circuit is necessary to get rotated rows of circulant similar to conventional SRAA in [9].
- Number of cycles is less.

For parallel RCE encoder

- Already existing parity shifting operation (for output transmission), is used for convolution, with slight modification.
- As LDPC code standard changes, Z changes, area of circuit can be adjusted by changing value of Lm.
- Number of cycles is more.

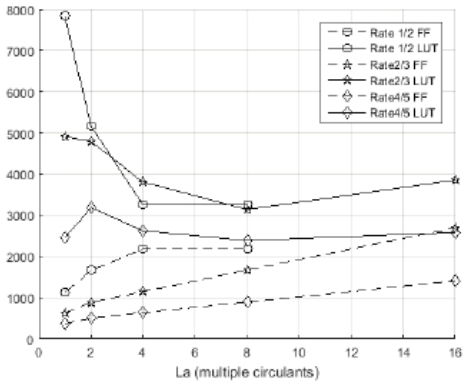


Fig. 7 Resource utilisation as a function of *La* parameter (AR4JA).

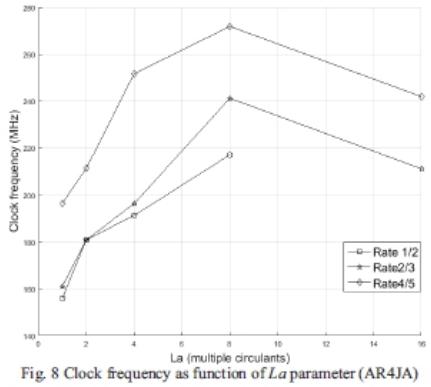


Fig. 8 Clock frequency as function of *La* parameter (AR4JA)

Fig: Area and Performance curves from [6]: The maximum performance is attained for La=8, where Lm x La=16.

We chose Lm=16, La=8, to reduce the latency.

Therefore,

Total systematic bits processed in parallel = Lm x La = 16 x 8 = 128

Total circulants processed in parallel = La x (Nb-Kb)= 8 x 8 =64

Each circulant multiplication completes in (Z/Lm) = 32/16 = 2 cycles

Since 8 circulants are processed In parallel, in 2 cycles 8 circulant multiplication is completed.

Total cycles of computation= (Kb / La) x (Z/Lm) = (32/8) x (32/16) = 4 x 2 = 8 cycles.

Pre-processing steps

For Parallel RCE, there are no pre-processing steps involved. The circuit for calculating parity is readily designed according to conventional matrix multiplication.

Here we analyse the intermediate product expressions obtained from normal matrix multiplication and from the parallel RCE network. This is done to identify any modifications to the order of input vector bits and circulant bits that are fed to the network.

Consider the case of 4x4 dense circulant being multiplied by a 4-bit vector to get a 4-bit parity result (r) :

$$[v_1 \quad v_2 \quad v_3 \quad v_4] * \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} = [r_1 \quad r_2 \quad r_3 \quad r_4] --- Eq(1)$$

Where vector variable **v**= {v1, v2, v3, v4} where v1, v2, v3, v4 are 32 bit vectors, and matrix m is a dense circulant of size 32x32, characterized by its first row, and product vector **r** = {r1, r2, r3, r4}, where r1, r2, r3, r4 are 32 bit vectors.

For the multiplication with 1 dense circulant:

$$Let\ \boldsymbol{f} = [f0, f1, f2, f3] = row\ vector\ of\ dense\ circulant\ of\ G - matrix$$

$$Let\ \boldsymbol{s} = [s0, s1, s2, s3] = systematic\ bits$$

The required expression for parity bits by normal matrix multiplication ($m * G = p_{actual}$):

$$p_{actual} = [p_1 \quad p_2 \quad p_3 \quad p_4]$$

$$p_1 = s0 * f0 + s1 * f3 + s2 * f2 + s3 * f1$$

$$p_2 = s0 * f1 + s1 * f0 + s2 * f3 + s3 * f2$$

$$p_3 = s0 * f2 + s1 * f1 + s2 * f0 + s3 * f3$$

$$p_4 = s0 * f3 + s1 * f2 + s2 * f1 + s3 * f0$$

$$p_{actual} = s0 * \boldsymbol{f} + s1 * \boldsymbol{f^{r(1)}} + s2 * \boldsymbol{f^{r(2)}} + s3 * \boldsymbol{f^{r(3)}} --- Eq(2)$$

In the (Fig.4) of [6], diagram of parallel RCE network is shown below,

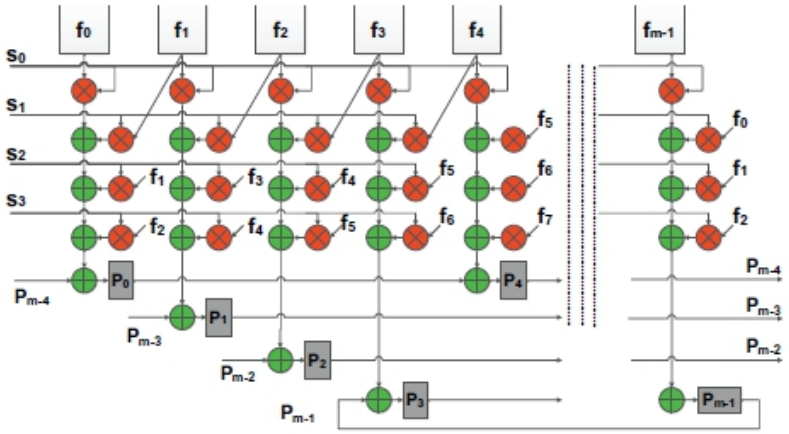


Fig. 4 Parallel RCE implementation (*Lm* parallelism)

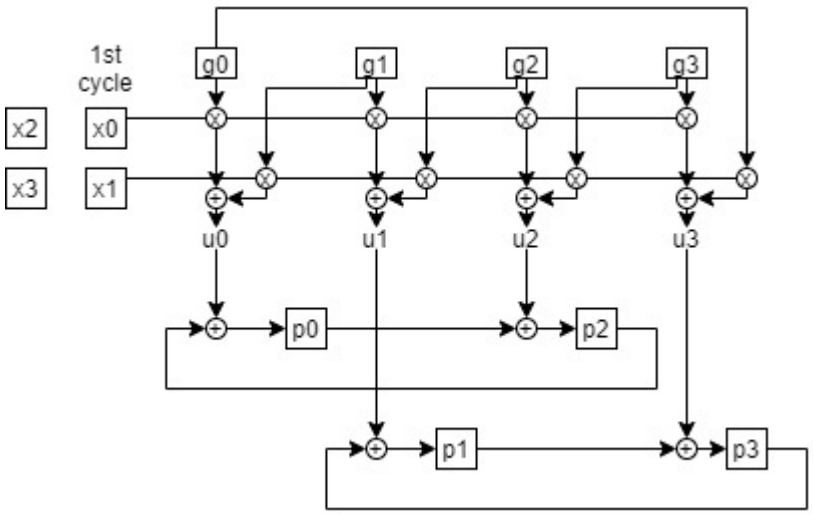


Fig. For Lm=2, circulant bits as g0,g1,g2,g3 and systematic bits as x0,x1,x2,x3 and parity registers p0,p1,p2,p3

Taking Lm=2, an arbitrary input vector $\boldsymbol{x} = [x0, x1, x2, x3]$ and the first row vector of an arbitrary dense circulant as $\boldsymbol{g} = [\, g0, g1, g2, g3]$

According to the parallel RCE network, the row vector is left shifted by 1 place as shown below.

$$\begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{g}^{l(1)} \end{bmatrix}$$

Each row should be multiplied with Lm (=2) vector bits in the following form in each cycle:

$$x_{cycle1} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

$$x_{cycle2} = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}$$

The result of each cycle (u0,u1,u2,u3) are accumulated with previous results in convolution manner to generate parity result.

From the network, parity register values at final cycle (cycle2) can be expressed as follows. (Note: the super script represents cycle number).

$$p0^2 = u0^2 + u3^1$$

$$p1^2 = u1^2 + u4^1$$

$$p2^2 = u2^2 + u0^1$$

$$p3^2 = u3^2 + u1^1$$

Let the parity result be denoted as ($\boldsymbol{r_{obtained}} = [r0, r1, r2, r3]$ = value of the parity registers at cycle 2) . Substituting u0,u1,u2,u3 in the above equations, the obtained mathematical expression of parity bits ($\boldsymbol{r_{obtained}}$) from parallel RCE network:

$$r_0 = x_2 * g0 + x_3 * g1 + x_0 * g2 + x_1 * g3$$

$$r_1 = x_2 * g1 + x_3 * g2 + x_0 * g3 + x_1 * g0$$

$$r_2 = x_2 * g2 + x_3 * g3 + x_0 * g0 + x_1 * g1$$

$$r_3 = x_2 * g3 + x_3 * g0 + x_0 * g1 + x_1 * g2$$

Compared with Eq(2):

$$\boldsymbol{r_{obtained}} = x_0 * \boldsymbol{g^{r(2)}} + x_1 * \boldsymbol{g^{r(1)}} + x_2 * \ \boldsymbol{g} + x_3 * \boldsymbol{g^{r(3)}} \neq \boldsymbol{p}_{actual}\ iff\ \boldsymbol{g} = \boldsymbol{f}$$

1) Correcting the order in which the systematic bits are fed into the network:

$$x_{cycle1} = \begin{bmatrix} x_3 \\ x_2 \end{bmatrix}$$

$$x_{cycle2} = \begin{bmatrix} x_1 \\ x_0 \end{bmatrix}$$

2) By feeding the circulant row vector after right shifting by 1 place:

$$\boldsymbol{g} = [g0, g1, g2, g3] = \boldsymbol{f^{r(1)}} = [f3, f0, f1, f2]$$

$$r_1 = x_1 * g0 + x_0 * g1 + x_3 * g2 + x_2 * g3$$

$$r_2 = x_1 * g1 + x_0 * g2 + x_3 * g3 + x_2 * g0$$

$$r_3 = x_1 * g2 + x_0 * g3 + x_3 * g0 + x_2 * g1$$

$$r_4 = x_1 * g3 + x_0 * g0 + x_3 * g1 + x_2 * g2$$

$$\begin{aligned} r_{obtained} &= x_0 * \boldsymbol{g^{l(1)}} + x_1 * \ \boldsymbol{g} + x_2 * \boldsymbol{g^{r(1)}} + x_3 * \boldsymbol{g^{r(2)}} \\ &= x_0 * \boldsymbol{f} + x_1 * \ \boldsymbol{f^{r(1)}} + x_2 * \boldsymbol{f^{r(2)}} + x_3 * \boldsymbol{f^{r(3)}} = \ p_{actual} \end{aligned}$$

In our analysis, we noticed that, to match the expressions for parity for higher values of Lm and circulant size, we found 2 feeding criteria:

1. The exact order of feeding systematic bits: first step is to bit-reverse the entire 1024 systematic bits. Then feed each Lm bits from MSB of the reversed systematic bits.
2. The exact order of feeding the first row of circulant: the first row of circulant ($\boldsymbol{f}$) is shifted right by (Lm-1) places

Computation Units

The computation units perform multiplication of bit vector and circulant matrix to generate parity bits. In Parallel RCE, it is the Parity generation unit,

In Parallel RCE encoder, the parity generation unit realizes the multiplication of systematic bits with the shifted versions of first row of dense circulant in generator matrix.

$$s * G_{paritypart} = p \ = p1 = \{p11, p12\}$$

The various shifted version are generated by appropriate wiring. In the Lm rows of AND and XOR gates, the multiplication of Lm bits of systematic part with Lm rows out of 32 rows of a dense circulant occurs in 1 cycle. Here we take Lm = 16, therefore the product is ready in 2 cycles.

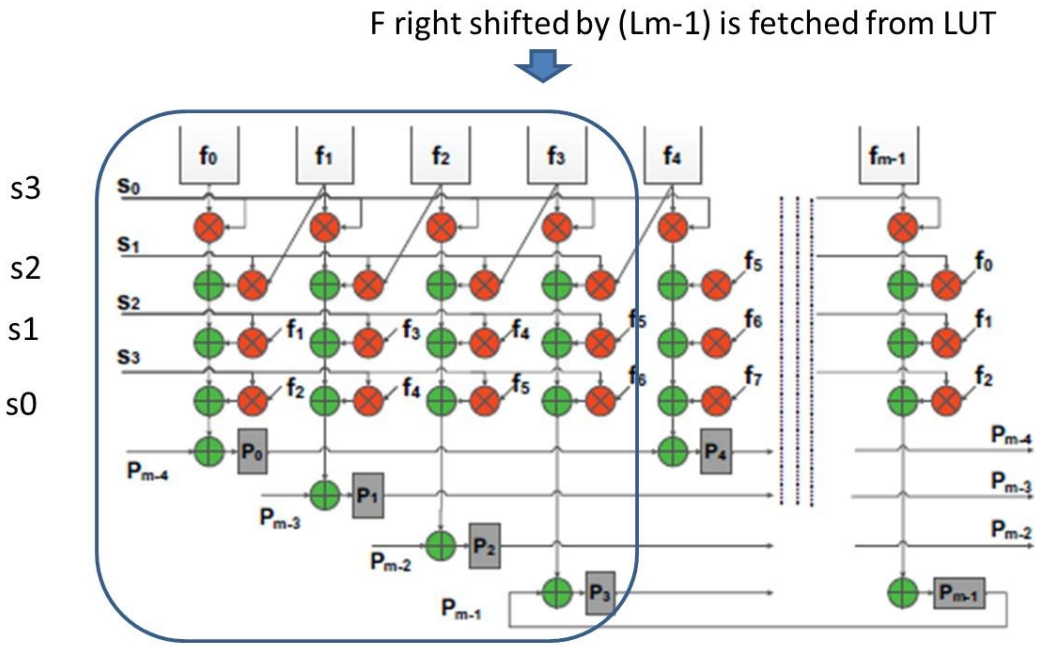


Fig. 4 Parallel RCE implementation (*Lm* parallelism)

The above shown is an example of parity generation unit core structure from [6], with Lm=4, and circulant size m=8, with the corrections obtained from pre-processing step. The circuit is partitioned based on Lm. The circuit elements in the box shown above is replicated M/Lm times, until the circulant size is reached.

The parity convolution buffer is formed by the registers { p0, to pm-1) along with the XORs in between. After multiplication of Lm bits and Lm rows, the Lm bits of product is accumulated with previous cycle's product obtained for preceding Lm bits. For example, in the above figure, to obtain the result for {p4, p5, …, p7}, the corresponding product from the

network of current cycle is XOR-ed with product obtained in previous cycle for preceding Lm bits residing in {p0,p1,…, p3}.

Also, there are multiple circulants in a block column j. In $G_{paritypart}$, we have 32 circulants in a block column.

So further parallelism is implemented to perform multiplication of La=8 circulants out of 32 circulants of a block column at a time, by replicating this structure La times. The 8 intermediate products in each cycle are accumulated (XOR-ed) before feeding to the parity convolution buffer. In 2 cycles, multiplication with 8 circulants completes to get the intermediate parity results. These are then Then next set of circulant data are loaded, and multiplication continues. After 4 passes, the parity results are ready. Therefore, the total encoding cycles taken is 8 cycles.

Controllers

For the Parallel RCE architecture, the controller consists of 2 FSMs that consists of various conunters:

1. Receiver and Encoder FSM: This consists of counters to count incoming systematic bits to provide control signals for input buffer and starting the encoder. It also consists of counters to count the number of encoding cycles. Same is used as address to the ROMs. Based on the counter values, appropriate control signals are generated for enabling shift operations and transmission of codeword.
2. Transmit FSM: This FSM functions as the output interface FSM to send out the systematic code word bits. It is initiated once a frame of systematic bits is received. In its first state, it counts the systematic bits as it enables the transmission of systematic bits from input buffer. During this time, the encoding is completed. In the state transition, control signals are given to unload parity bits from convolution buffer. Then, in the next state, transmission from parity shifting buffer is enabled, and parity bits are transmitted.

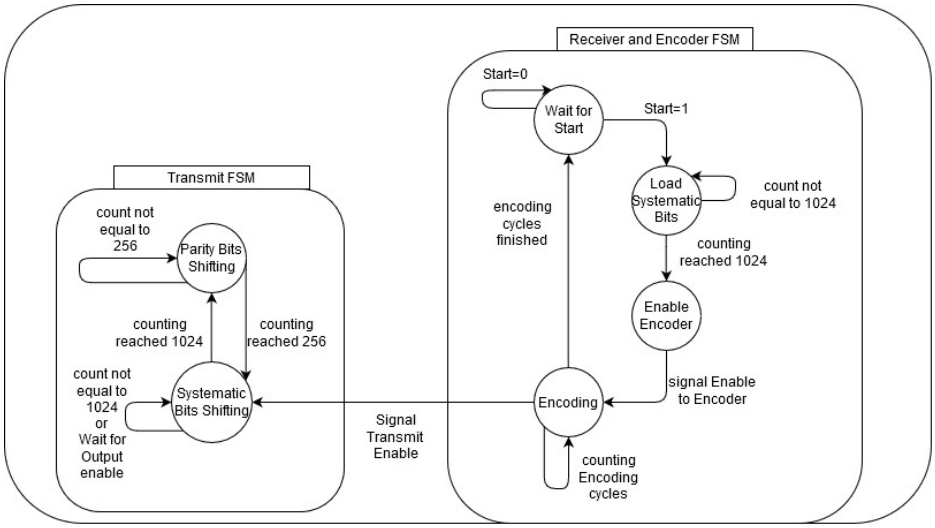


Fig. Parallel RCE Controller FSM state diagram

ROMs

First rows of dense circulant: In Parallel RCE encoder, the function generators are ROMs that store the first rows of dense circulants of G-matrix.

Near Earth Encoder

Pre-processing/Analysis

Expected equations:

Consider the case of 4x4 dense circulant being multiplied by a 4-bit vector to get a 4-bit parity result (r) :

$$[s0\ s1\ s2\ s3\ s4\ s5] * \begin{bmatrix} f0 & f1 & f2 & f3 & f4 & f5 \\ f1 & f2 & f3 & f4 & f0 & f5 \\ f2 & f3 & f4 & f0 & f1 & f5 \\ f3 & f4 & f0 & f1 & f2 & f5 \\ f4 & f0 & f1 & f2 & f3 & f5 \\ f5 & f5 & f5 & f5 & f5 & f5 \end{bmatrix} = [p_0 \quad p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5]$$

For the multiplication with 1 dense circulant, additional column and row added with f5:

$$Let\ \boldsymbol{f} = [f0, f1, f2, f3, f4, f5] = row\ vector\ of\ dense\ circulant\ of\ G - matrix$$

$$Let\ \boldsymbol{s} = [s0, s1, s2, s3, s4, s5] = systematic\ bits$$

With s5 as an additional bit added to make even length.

Assume s5=f5=0.

The required expression for parity bits by normal matrix multiplication ($m * G = p_{actual}$):

$$p_{actual} = [p_0 \quad p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5]$$

$$p_0 = s0 * f0 + s1 * f4 + s2 * f3 + s3 * f2 + s4 * f1 + s5 * f5$$

$$p_1 = s0 * f1 + s1 * f0 + s2 * f4 + s3 * f3 + s4 * f2 + s5 * f5$$

$$p_2 = s0 * f2 + s1 * f1 + s2 * f0 + s3 * f4 + s4 * f3 + s5 * f5$$

$$p_3 = s0 * f3 + s1 * f2 + s2 * f1 + s3 * f0 + s4 * f4 + s5 * f5$$

$$p_4 = s0 * f4 + s1 * f3 + s2 * f2 + s3 * f1 + s4 * f0 + s5 * f5$$

$$p_5 = s0 * f5 + s1 * f5 + s2 * f5 + s3 * f5 + s4 * f5 + s5 * f5$$

$$p_{actual} = s0 * \boldsymbol{f} + s1 * \boldsymbol{f^{r(1)}} + s2 * \boldsymbol{f^{r(2)}} + s3 * \boldsymbol{f^{r(3)}} + s4 * \boldsymbol{f}^{r(4)} + s5 * f5 \qquad Eq(2)$$

Observed Equations

Deriving expressions for parallel rce network of (Z=5, Lm=2):

The parallel RCE network in shorthand form:

$$cycles: \quad 2,\ 1,\ 0$$

$$s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow f_0,\ f_1,\ f_2,\ f_3,\ f_4,\ f_5$$

$$s_0 \rightarrow s_2 \rightarrow s_4 \rightarrow f_1,\ f_2,\ f_3,\ f_4,\ f_0,\ f_5$$

$$u_0,\ u_1,\ u_2,\ u_3,\ u_4,\ u_5$$

$$p_0 \quad p_2 \quad p_4$$
$$p_1 \quad p_3 \quad p_5$$

(Superscript represent cycle)

At cycle 0

$$u_0^0 = s5f0 + s4f1$$

$$u_1^0 = s5f1 + s4f2$$

$$u_2^0 = s5f2 + s4f3$$

$$u_3^0 = s5f3 + s4f4$$

$$u_4^0 = s5f4 + s4f0$$

$$u_5^0 = s5f5 + s4f5$$

At cycle 1

$$u_0^1 = s3f0 + s2f1$$

$$u_1^1 = s3f1 + s2f2$$

$$u_2^1 = s3f2 + s2f3$$

$$u_3^1 = s3f3 + s2f4$$

$$u_4^1 = s3f4 + s2f0$$

$$u_5^1 = s3f5 + s2f5$$

At cycle 2

$$u_0^2 = s1f0 + s0f1$$

$$u_1^2 = s1f1 + s0f2$$

$$u_2^2 = s1f2 + s0f3$$

$$u_3^2 = s1f3 + s0f4$$

$$u_4^2 = s1f4 + s0f0$$

$$u_5^2 = s1f5 + s0f5$$

Parity Register expressions:

At cycle 1

$$P_0 = u_0^0$$

$$P_1 = u_1^0$$

$$P_2 = u_2^0$$

$$P_3 = u_3^0$$

$$P_4 = u_4^0$$

$$P_5 = u_5^0$$

At cycle 2

$$P_0 = u_0^1 + u_4^0$$

$$P_1 = u_1^1 + u_5^0$$

$$P_2 = u_2^1 + u_0^0$$

$$P_3 = u_3^1 + u_1^0$$

$$P_4 = u_4^1 + u_2^0$$

$$P_5 = u_5^1 + u_3^0$$

At cycle 3

$$P_0 = u_0^2 + u_4^1 + u_2^0$$

$$P_1 = u_1^2 + u_5^1 + u_3^0$$

$$P_2 = u_2^2 + u_0^1 + u_4^0$$

$$P_3 = u_3^2 + u_1^1 + u_5^0$$

$$P_4 = u_4^2 + u_2^1 + u_0^0$$

$$P_5 = u_5^2 + u_3^1 + u_1^0$$

At cycle 4:

Assuming parity bits will align properly, here we are taking cycle 4 when we feed one more set of zero systematic messages:

$$P_0 = u_4^2 + u_2^1 + u_0^0$$

$$P_1 = u_5^2 + u_3^1 + u_1^0$$

$$P_2 = u_0^2 + u_4^1 + u_2^0$$

$$P_3 = u_1^2 + u_5^1 + u_3^0$$

$$P_4 = u_2^2 + u_0^1 + u_4^0$$

$$P_5 = u_3^2 + u_1^1 + u_5^0$$

Comparing with the normal matrix multiplication expressions, some corrections are made. The corrected expressions are shown:

$$P_0 = u_4^2 + u_2^1 + u_0^0$$

$$P_1 = u_5^2 + u_3^1 + u_1^0 \rightarrow \boldsymbol{u_0^2} + u_3^1 + u_1^0$$

$$P_2 = u_0^2 + u_4^1 + u_2^0 \rightarrow \boldsymbol{u_1^2} + u_4^1 + u_2^0$$

$$P_3 = u_1^2 + u_5^1 + u_3^0 \rightarrow \boldsymbol{u_2^2} + \boldsymbol{u_0^1} + u_3^0$$

$$P_4 = u_2^2 + u_0^1 + u_4^0 \rightarrow \boldsymbol{u_3^2} + \boldsymbol{u_1^1} + u_4^0$$

$$P_5 = u_3^2 + u_1^1 + u_5^0 \rightarrow \boldsymbol{u_5^2} + \boldsymbol{u_5^1} + \boldsymbol{u_5^0} \rightarrow \textit{can be avoided}$$

By grouping terms: to get cycle2 parity register expressions

$$P_{II} = u_2^2 + u_0^1$$

$$P_{III} = u_3^2 + u_1^1$$

$$P_{IV} = u_4^2 + u_2^1$$

$$P_O = u_0^2 + u_3^1$$

$$P_I = u_1^2 + u_4^1$$

Ignoring P5.

Further group terms as: to get cycle 3 parity register expressions

$$P_4 = u_4^3 + P_{II}$$

$$P_0 = u_0^3 + P_{III}$$

$$P_1 = u_1^3 + P_{IV}$$

$$P_2 = u_2^3 + P_O$$

$$P_3 = u_3^3 + P_I$$

Further add 0 and get cycle4 expressions:

$$P_0^4 = 0 + P_4$$

$$P_1^4 = 0 + P_0$$

$$P_2^4 = 0 + P_1$$

$$P_3^4 = 0 + P_2$$

$$P_4^4 = 0 + P_3$$

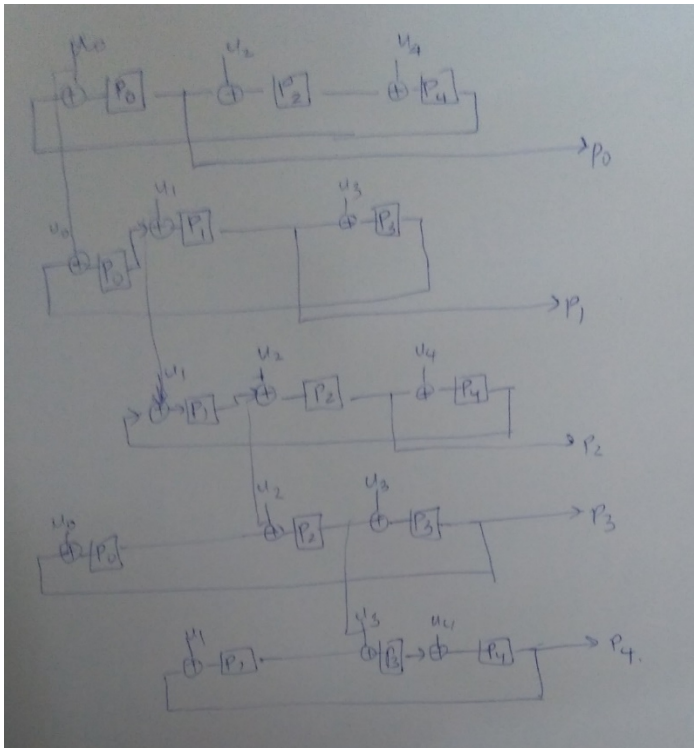


Fig.7. Parity shift networks based on above equations.

Observed Rule 1: Left shift Wiring of circulant bits

From the parallel RCE network, circulant bits are left shifted in each row of the Multiply-Accumulate (MAC) network.

Rule2: Add Rule, Parity Shift sequence:

Further Compressing above circuit (fig.7), we obtain a rule for generating parity shift sequence:

From Fig.6., if we analyze the subscripts alone of 'u', we get the following format

$$0 \rightarrow 2 \rightarrow 4 \rightarrow P_0$$

$$1 \rightarrow 3 \rightarrow 0 \rightarrow P_1$$

$$2 \rightarrow 4 \rightarrow 1 \rightarrow P_2$$

So we can combine these to form the following loop:

$$0 \rightarrow 2 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 0$$

So parity shift loop can be formed using Registers:

$$R_0 \rightarrow (+u_2) = R_2 \rightarrow (+u_4) = R_4 \rightarrow (+u_1) = R_1 \rightarrow (+u_3) = R_3 \rightarrow (+u_0) = R_0$$

i.e.

$$u_0\ u_1\ u_2\ u_3\ u_4\ \ +$$

$$R_3\ R_4\ R_0\ R_1\ R_2\ =$$

$$R_0\ R_1\ R_2\ R_3\ R_4$$

$$\boldsymbol{u}[0:4] + \boldsymbol{R}[0:4]^{r(2)} = \boldsymbol{R}[0:4]$$

Design Rule: For arbitrary Lm value which is not a factor of Z, this sequence can be obtained by a modulo-Z arithmetic progression with a difference of Lm, stopping when first index occurs again.

Addition Rule:

$$\boldsymbol{u}[0:Z-1] + \boldsymbol{R}[0:Z-1]^{r(Lm)} = \boldsymbol{R}[0:Z-1]$$

Note: For Lm which is a factor of Z, such a sequence starting from 0, will not include all the indices (0 to Z-1). So for indices 1 to Lm-1, separate arithmetic progressions are to be written until we get sequences covering all indices. This results in Lm sequences each of length (Z/Lm).

Rule3: for output parity bits or circulant bits:

Case 1: First row of circulant bits fed as input to network:

$$cycles:\ \ 2,\ 1,\ 0$$

$$s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow f_0,\ \ f_1,\ \ f_2,\ \ f_3,\ \ f_4$$

$$s_0 \rightarrow s_2 \rightarrow s_4 \rightarrow f_1,\ \ f_2,\ \ f_3,\ \ f_4,\ \ f_0$$

$$u_0,\ \ u_1,\ \ u_2,\ \ u_3,\ \ u_4$$

Parity bits can be taken, **in parallel,** as output from R registers at 3rd cycle itself instead of 4th cycle as follows:

$$P_0 = u_4^2 + u_2^1 + u_0^0 = R_4^3$$

$$P_1 = u_0^2 + u_3^1 + u_1^0 = R_0^3$$

$$P_2 = u_1^2 + u_4^1 + u_2^0 = R_1^3$$

$$P_3 = u_2^2 + u_0^1 + u_3^0 = R_2^3$$

$$P_4 = u_3^2 + u_1^1 + u_4^0 = R_3^3$$

Here output parity vector **p** can be obtained from vector **R** of 3rd cycle as:

$$\boldsymbol{p} = \{p_0\ p_1\ p_2\ p_3\ p_4\} = \{R_4^3\ R_0^3\ R_1^3\ R_2^3\ R_3^3\} = (\boldsymbol{R}^3)^{r(1)}$$

The right shift amount is same as the left shift amount of last line (i.e. $f^{l(1)}$).

Case2: If last row of circulant is fed as input:

$$s_1 \to s_3 \to s_5 \to f_1,\ f_2,\ f_3,\ f_4,\ f_0\ (same\ as\ f^{l(1)})$$

$$s_0 \to s_2 \to s_4 \to f_2,\ f_3,\ f_4,\ f_0,\ f_1\ (same\ as\ f^{l(2)})$$

$$u_0,\ u_1,\ u_2,\ u_3,\ u_4$$

$$Parity\ registers: R_0\ R_1\ R_2\ R_3\ R_4\ R_5\ (connected\ with\ xor\ in\ between\ in\ the\ 0 \to 2 \to 4 \to 1 \to 3 \to 0\ sequence)$$

Then parity bits can be taken out as:

$$P_0 = u_3^2 + u_1^1 + u_4^0 = R_3^3$$

$$P_1 = u_4^2 + u_2^1 + u_0^0 = R_4^3$$

$$P_2 = u_0^2 + u_3^1 + u_1^0 = R_0^3$$

$$P_3 = u_1^2 + u_4^1 + u_2^0 = R_1^3$$

$$P_4 = u_2^2 + u_0^1 + u_3^0 = R_2^3$$

We can conclude that vector **p** can be obtained from vector **R** of 3rd cycle as:

$$\boldsymbol{p} = \{p_0\ p_1\ p_2\ p_3\ p_4\} = \{R_3^3\ R_4^3\ R_0^3\ R_1^3\ R_2^3\} = (\boldsymbol{R}^3)^{r(2)}$$

The right shift amount is same as the left shift amount of last line (i.e. $f^{l(2)}$).

Design Rule: If the circulant bits fed into the network is $f^{l(a)}$, where a is the left shift amount, then the last line will have a left shift of $(a + Lm - 1)$, then parity result vector is R at the cycle=ceil(Z/Lm), and right shifted by $(a + Lm - 1)$ given as:

$$\boldsymbol{p} = (\boldsymbol{R}^{ceil(\frac{Z}{Lm})})^{r(a+Lm-1)}$$

Instead of taking shifted version of R as parity, the circulant bits fed to the network can be shifted so that P=R.

Case 2a: To make Parity P= R, If second row of circulant is fed as input:

$$s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow f_4,\ f_0,\ f_1,\ f_2,\ f_3\ \left(same\ as\ f^{l(4)}\right)$$

$$s_0 \rightarrow s_2 \rightarrow s_4 \rightarrow f_0,\ f_1,\ f_2,\ f_3,\ f_4\ (same\ as\ f^{l(5)} = f^{l(0)})$$

$$u_0,\ u_1,\ u_2,\ u_3,\ u_4$$

$$Parity\ registers: R_0\ R_1\ R_2\ R_3\ R_4\ R_5\ (\ connected\ with\ xor\ in\ between\ in\ the\ 0 \rightarrow 2 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 0\ sequence)$$

Then parity bits can be taken out as:

$$P_0 = u_0^2 + u_3^1 + u_1^0 = R_0^3$$

$$P_1 = u_1^2 + u_4^1 + u_2^0 = R_1^3$$

$$P_2 = u_2^2 + u_0^1 + u_3^0 = R_2^3$$

$$P_3 = u_3^2 + u_1^1 + u_4^0 = R_3^3$$

$$P_4 = u_4^2 + u_2^1 + u_0^0 = R_4^3$$

We can conclude that vector **p** can be obtained from vector **R** of 3rd cycle as:

$$\boldsymbol{p} = \{p_0\ p_1\ p_2\ p_3\ p_4\ \} = (\boldsymbol{R}^3)$$

Note: This is same as the rule obtained for deep space case, where we are feeding circulant bits after one right shift (i.e. second row of circulant).

Case 3: If some information bits are zero, say s0=s1=s2=0. This makes, u at cycle 3 to be zero.

Then we can avoid cycle 3 entirely, as the terms are zero, and the equations:

For Case1,

$$P_0 = u_2^1 + u_0^0 = R_2^2$$

$$P_1 = u_3^1 + u_1^0 = R_3^2$$

$$P_2 = u_4^1 + u_2^0 = R_4^2$$

$$P_3 = u_0^1 + u_3^0 = R_0^2$$

$$P_4 = u_1^1 + u_4^0 = R_1^2$$

$$\boldsymbol{p} = \{p_0\ p_1\ p_2\ p_3\ p_4\ \} = (\boldsymbol{R}^2)^{r(3)}$$

For Case 2,

$$P_0 = u_1^1 + u_4^0 = R_1^2$$

$$P_1 = u_2^1 + u_0^0 = R_2^2$$

$$P_2 = u_3^1 + u_1^0 = R_3^2$$

$$P_3 = u_4^1 + u_2^0 = R_4^2$$

$$P_4 = u_0^1 + u_3^0 = R_0^2$$

$$\boldsymbol{p} = \{p_0\ p_1\ p_2\ p_3\ p_4\} = (\boldsymbol{R}^2)^{r(4)}$$

For Case2a,

$$P_0 = u_3^1 + u_1^0 = R_3^2$$

$$P_1 = u_4^1 + u_2^0 = R_4^2$$

$$P_2 = u_0^1 + u_3^0 = R_0^2$$

$$P_3 = u_1^1 + u_4^0 = R_1^2$$

$$P_4 = u_2^1 + u_0^0 = R_2^2$$

$$\boldsymbol{p} = \{p_0\ p_1\ p_2\ p_3\ p_4\} = (\boldsymbol{R}^2)^{r(2)}$$

As per paper:

For Lm=16, Z=511 first circulant case:

If the circulant row fed to the network is last row (i.e. $f^{l(1)}$), and for all other circulants the circulant row fed to the network is (i.e. f), then the parity P expression equations from registers R should be constant.?

But as per the observation from Lm=2, Z=5, it is not observed so.

Case 3a: To correct the order and make P=R in second cycle of Case2a,

Case 2a circuit:

$$s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow f_4,\ f_0,\ f_1,\ \boldsymbol{f_2},\ f_3\ \left(same\ as\ f^{l(4)}\right)$$

$$s_0 \rightarrow s_2 \rightarrow s_4 \rightarrow f_0,\ f_1,\ f_2,\ \boldsymbol{f_3},\ f_4\ (same\ as\ f^{l(5)} = f^{l(0)})$$

$$u_0,\ u_1,\ u_2,\ \boldsymbol{u_3},\ u_4$$

F values for u3 should arrive at u0, f values for u4 should arrive at u1 and so on.

This change in arrangement of f values results in following, where the second last row of circulant (i.e. $f^{l(2)}$), is fed as input:

$$s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow \boldsymbol{f_2},\ f_3,\ f_4,\ f_0,\ f_1\left(same\ as\ f^{l(2)}\right)$$

$$s_0 \rightarrow s_2 \rightarrow s_4 \rightarrow \boldsymbol{f_3},\ f_4,\ f_0,\ f_1,\ f_2(same\ as\ f^{l(3)})$$

$$\boldsymbol{u_0},\ u_1,\ u_2,\ u_3,\ u_4$$

$$P_0 = u_0^1 + u_3^0 = R_0^2$$

$$P_1 = u_1^1 + u_4^0 = R_1^2$$

$$P_2 = u_2^1 + u_0^0 = R_2^2$$

$$P_3 = u_3^1 + u_1^0 = R_3^2$$

$$P_4 = u_4^1 + u_2^0 = R_4^2$$

$$\boldsymbol{p} = \{p_0\ p_1\ p_2\ p_3\ p_4\ \} = (\boldsymbol{R}^2)$$

Rule: From paper, rule followed is for first circulant, feed the last row of circulant (i.e. $f^{l(1)}$), while for remaining circulants feed the first row of circulant (i.e. f).

From above analysis, rule to be followed for first circulant case is feed the second last row of circulant (i.e. $f^{l(2)}$), while for remaining circulants feed the last row of circulant (i.e. $f^{l(4)}$).

For an arbitrary case, we have cycles 0, 1, 2, 3, …, (Z/Lm)-1, (Z/Lm). Where we take output parity bits at cycle=ceil(Z/Lm), since at cycle0, parity bit values are assumed zero.

For Cycle ceil(Z/Lm), i.e 3rd cycle, to make P=R, feed circulant bits = $f^{l(4)} = f^{r(1+0)}$

For cycle ceil(Z/Lm)-1, i.e 2nd cycle, to make P=R, feed circulant bits = $f^{l(2)} = f^{l(4-2)} = f^{r(3)} = f^{r(1+2)}$

For cycle ceil(Z/Lm-2), i.e 1st cycle, to make P=R, feed circulant bits = $f = f^{l(5)} = f^{l(4-4)} = f^{r(5)} = f^{r(1+4)}$

For arbitrary cycle b from last cycle i.e. at cycle (Z/Lm)-b, to make P=R, feed circulant bits = $f^{r(1+(b*Lm))}$

RCE parallelizing Method

Bit serial RCE encoder:

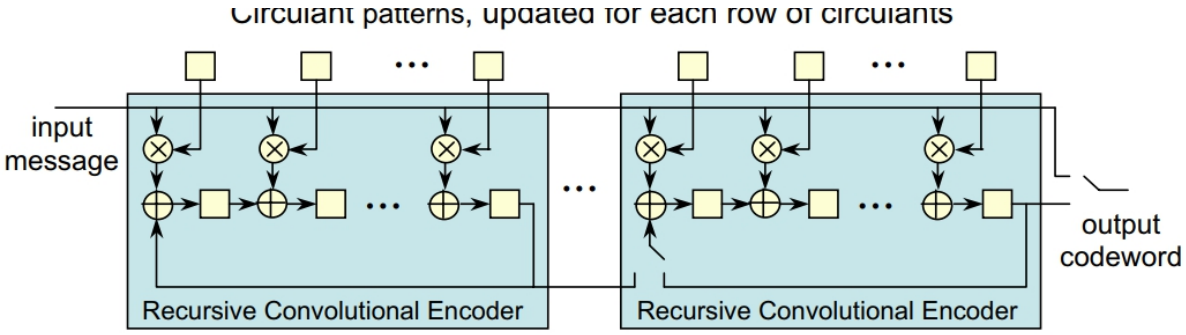


Fig. 6. A Quasicyclic Encoder Using Feedback Shift Registers

Bit serial RCE encoder in shorthand form:

$$s_0 \to s_1 \to s_2 \to s_3 \to s_4 \to s[n] \to \quad f_0, \quad f_1, \quad f_2, \quad f_3, \quad f_4$$
$$u_0, \quad u_1, \quad u_2, \quad u_3, \quad u_4$$
$$p4+, \; p0+, \; p1+, \; p2+, \; p3+$$
$$p_0, \quad p_1, \quad p_2, \quad p_3, \quad p_4 \to p[n]$$

Serially feed the information bits (s0,s1,s2,s3,s4) such that

$$s[0] = s4, \; s[1] = s3, \; s[2] = s2, \; s[3] = s1, \; s[4] = s0, \; s[5] = 0$$

Take parity bits serially out after first 5 cycles (cycles 0 to 4), i.e. at cycle 5 onwards

$$p[5] = p4, \; p[6] = p3, \; p[7] = p2, \; p[8] = p1, \; p[9] = p0, \; p[10] = 0$$

Unfolding:

Use unfolding DSP algorithm, we can convert bit serial structures to parallel structure. If the parity registers are represented as {R0,R1,R2,R3,R4}

$$s_0 \to s_1 \to s_2 \to s_3 \to s_4 \to s[n] \to \quad f_0, \quad f_1, \quad f_2, \quad f_3, \quad f_4$$
$$u_0, \quad u_1, \quad u_2, \quad u_3, \quad u_4$$

$$p4+,\ p0+,\ p1+,\ p2+,\ p3+$$

$$a,\quad b,\quad c,\quad d,\quad e \rightarrow p[n]$$

$$e \rightarrow R4 \rightarrow a,\quad a \rightarrow R0 \rightarrow b,\quad b \rightarrow R1 \rightarrow c,\quad c \rightarrow R2 \rightarrow d,\quad d \rightarrow R3 \rightarrow e$$

Representing parity registers (R0,to R4) as Delay elements D:

$$e \rightarrow D \rightarrow a,\quad a \rightarrow D \rightarrow b,\quad b \rightarrow D \rightarrow c,\quad c \rightarrow D \rightarrow d,\quad d \rightarrow D \rightarrow e$$

Let (a,b,c,d,e) represent the nodes at the (+) operations, which represent the wole operation of (multiply and accumulate) at that point. Also there is one delay element (D) between each node connections, parameter w=1.

To make Lm=j=2 parallelism, we replicate the structure Lm times, so that we get Lm versions of the input node s[n] and output node p[n] and internal nodes (a,b,c,d,e).

Input and output node becomes 2:

s1[k]= s[2k+1] ,s0[k]=s[2k]: here we are assuming to feed s1[0]=s5, assuming s5 as 0.

p1[k]= p[2k+1], p0[k]= p[2k]

Where k=0,1,…, ceil(5/Lm)-1

To generalize: for arbitrary Lm value we have inputs: s[Lm*k], s[Lm*k +1], s[Lm*k + 2], …, s[Lm*k + Lm-1

Replicated unconnected structure :

$$s0[k] \rightarrow\quad f_0,\quad f_1,\quad f_2,\quad f_3,\quad f_4$$

$$u_0,\quad u_1,\quad u_2,\quad u_3,\quad u_4$$

$$p4+,\ p0+,\ p1+,\ p2+,\ p3+$$

$$a0,\quad b0,\quad c0,\quad d0,\quad e0 \rightarrow p0[k]$$

$$s1[k] \rightarrow\quad f_0,\quad f_1,\quad f_2,\quad f_3,\quad f_4$$

$$u_0,\quad u_1,\quad u_2,\quad u_3,\quad u_4$$

$$p4+,\ p0+,\ p1+,\ p2+,\ p3+$$

$$a1,\quad b1,\quad c1,\quad d1,\quad e1 \rightarrow p1[k]$$

The connections have to preserved in the following order (as it was in serial circuit):

$$e \to D \to a, \qquad a \to D \to b, \qquad b \to D \to c, \qquad c \to D \to d, \qquad d \to D \to e$$

Starting node (index=u)	Connect to (u+w)%j	Delay elements=Floor((u+w)/j)	Connection precedence
a0	(0+1)%2=1 => b1	(0+1)/2=0=>0 delay	a->b
b0	c1	0	b->c
c0	d1	0	c->d
d0	e1	0	d->e
e0	a1	0	e->a
a1	(1+1)%2=0 => b0	(1+1)/2=1=>1 delay	a->b
b1	c0	1 delay	b->c
c1	d0	1 delay	c->d
d1	e0	1 delay	d->e
e1	a0	1 delay	e->a
e0	(0+0)%2=0=>p0	(0+0)/2=0=>0 delay	e->p[n]
e1	(1+0)%2=1=>p1	(1+0)/2=0=>0 delay	e->p[n]

Applying above results on the Replicated unconnected structure, we get:

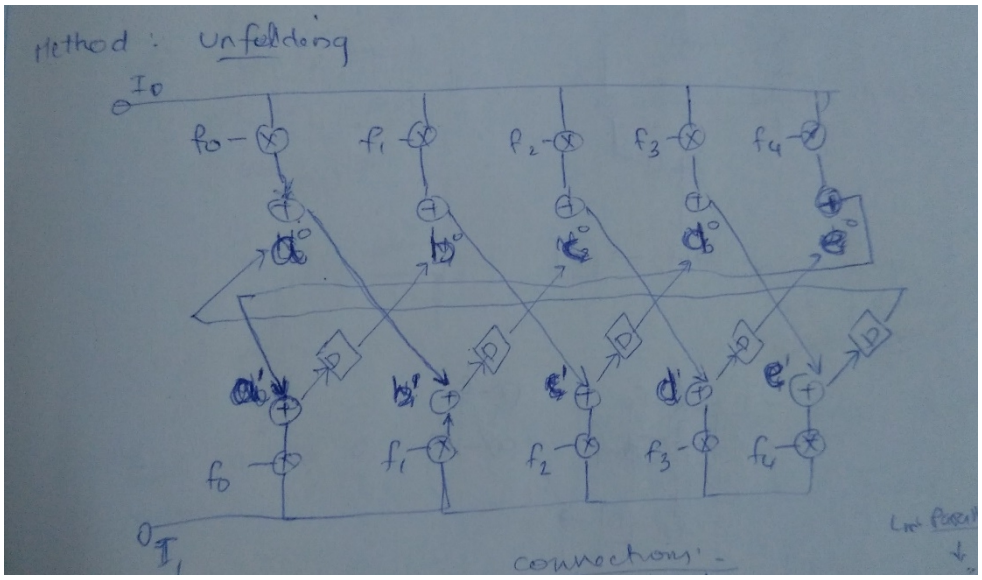


Fig. unfolded structure.

$$s0[k] \to \quad f_0, \quad f_1, \quad f_2, \quad f_3, \quad f_4$$

$$u_0, \quad u_1, \quad u_2, \quad u_3, \quad u_4$$

$$p4+, \quad p0+, \quad p1+, \quad p2+, \quad p3+$$

$$a0, \quad b0, \quad c0, \quad d0, \quad e0 \to p0[k]$$

$$s1[k] \rightarrow \quad f_0, \quad f_1, \quad f_2, \quad f_3, \quad f_4$$

$$u_0, \quad u_1, \quad u_2, \quad u_3, \quad u_4$$

$$p4+, \; p0+, \; p1+, \; p2+, \; p3+$$

$$a1, \quad b1, \quad c1, \quad d1, \quad e1 \rightarrow p1[k]$$

$$e0 \rightarrow a1, \quad a0 \rightarrow b1, \quad b0 \rightarrow c1, \quad c0 \rightarrow d1, \quad d0 \rightarrow e1$$

$$e1 \rightarrow D \rightarrow a0, \quad a1 \rightarrow D \rightarrow b0, \quad b1 \rightarrow D \rightarrow c0, \quad c1 \rightarrow D \rightarrow d0, \quad d1 \rightarrow D \rightarrow e0$$

Observed Rule1: Inherent left shift of circulant bits:

When we connect based on the table, we can see that the 0-delay connections will align f1 under f0, f2 under f1, and so on. For arbitrary Lm, each line is fed with circulant bits $f^{l(j)}$ where j = 0 to Lm-1.

Further compressing, we get the structure (as initially assumed parallel rce structure):

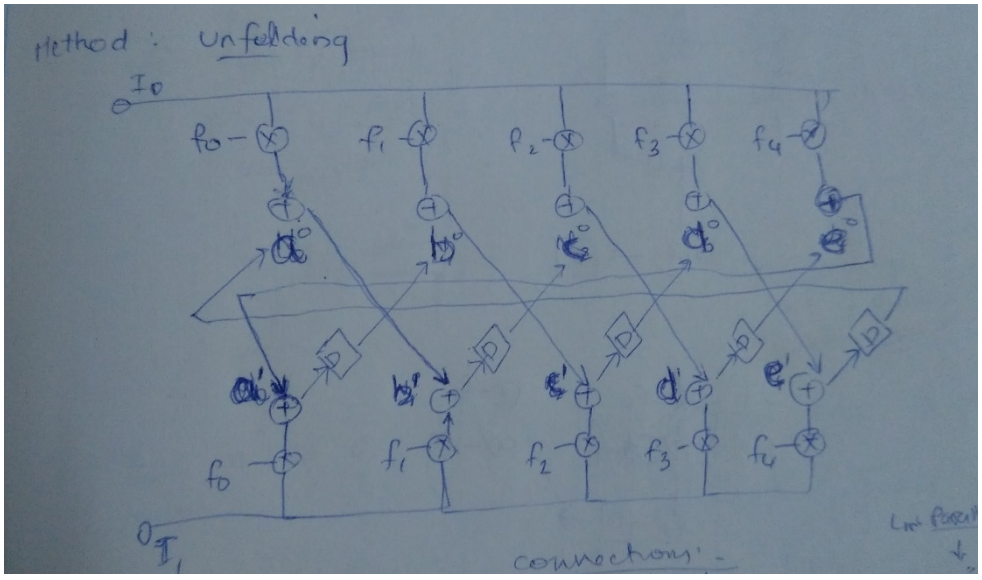
$$cycles, k: \quad 2, \; 1, \; 0$$

$$s_1 \rightarrow s_3 \rightarrow s_5 \rightarrow s[2k+1] \rightarrow f_0, \; f_1, \; f_2, \; f_3, \; f_4$$

$$s_0 \rightarrow s_2 \rightarrow s_4 \rightarrow \quad s[2k] \rightarrow f_1, \; f_2, \; f_3, \; f_4, \; f_0$$

$$u_0, \; u_1, \; u_2, \; u_3, \; u_4$$

Observed Rule 2: Add Rule, Register indices and Parity Shift Sequence



Initial in serial architecture, register indices and nodes were related as:

$$e \rightarrow R4 \rightarrow a, \quad a \rightarrow R0 \rightarrow b, \quad b \rightarrow R1 \rightarrow c, \quad c \rightarrow R2 \rightarrow d, \quad d \rightarrow R3 \rightarrow e$$

Let the nodes {a,b,c,d,e} be represented as {a0,b0,c0,d0,e0}. Then we get the following naming convention, where register that comes after a0 is R0, register that comes after b0 is R1 and so on.

$$e0 \to R4 \to a0, \quad a0 \to R0 \to b0, \quad b0 \to R1 \to c0, \quad c0 \to R2 \to d0, \quad d0 \to R3 \to e0 \; -- (naming1)$$

For parallel structure, we have the connections obtained based on the table as:

$$e0 \to a1, \quad a0 \to b1, \quad b0 \to c1, \quad c0 \to d1, \quad d0 \to e1$$

$$e1 \to D \to a0, \quad a1 \to D \to b0, \quad b1 \to D \to c0, \quad c1 \to D \to d0, \quad d1 \to D \to e0$$

Combining above rules and substituting registers in place of delay elements by following the naming convention as in naming1:

$$\boldsymbol{e0} \to a1 \to \boldsymbol{R_4} \to b0, \quad \boldsymbol{a0} \to b1 \to \boldsymbol{R_0} \to c0, \quad \boldsymbol{b0} \to c1 \to \boldsymbol{R_1} \to d0, \quad \boldsymbol{c0} \to d1 \to \boldsymbol{R_2} \to e0, \quad \boldsymbol{d0} \to e1 \to \boldsymbol{R_3} \to a0$$

To generalize, to find the register indices from unfolded structure,

If nodes are $\{ n_0, n_1, n_2, ..., n_a, n_b ..., n_{Z-1} \}$, then after replication and node connections,

For arbitrary node connection pair 'n_a' to 'n_b' in the table, having a delay element in the connection sequence, the register index for the delay element is:

$$R_i, where\ i = \big(a - (Lm - 1)\big)\%Z$$

The Shift sequence of registers R:

$$0 \to 2 \to 4 \to 1 \to 3 \to 0$$

Based on above rule:

$$R_2 \to e0 \to a1(+u4) \to R_4 \to b0, \quad R_3 \to a0 \to b1(+u0) \to R_0 \to c0, \quad R_4 \to b0 \to c1(+u1) \to R_1 \to d0, \quad R_0 \to c0 \to d1(+u2) \to R_2 \to e0, \quad R_1 \to d0 \to e1(+u3) \to R_3 \to a0$$

$$R_2 \to (+u4) \to R_4, \quad R_3 \to (+u0) \to R_0, \quad R_4 \to (+u1) \to R_1, \quad R_0 \to (+u2) \to R_2, \quad R_1 \to (+u3) \to R_3$$

Addition Rule:

$$u_0\ u_1\ u_2\ u_3\ u_4 \quad +$$

$$R_3\ R_4\ R_0\ R_1\ R_2 \quad =$$

$$R_0\ R_1\ R_2\ R_3\ R_4$$

$$\boldsymbol{u}[0:4] + \boldsymbol{R}[0:4]^{r(2)} = \boldsymbol{R}[0:4]$$

Observed Rule 3: Output Parity bits:

To take parity bits output correctly **to satisfy equations**, the mapping of actual parity bits to register values is as shown:

$$\boldsymbol{p} = \{p_0\, p_1\, p_2\, p_3\, p_4\,\} = \{\, R_4^3\, R_0^3\, R_1^3\, R_2^3\, R_3^3\,\} = (\boldsymbol{R}^3)^{r(1)}$$

Take parity bits serially out after first 3 cycles, i.e. at cycle 3 onwards

$$R3 = p4,\ R2 = p3,\ R1 = p2,\ R0 = p1,\ R4 = p0$$

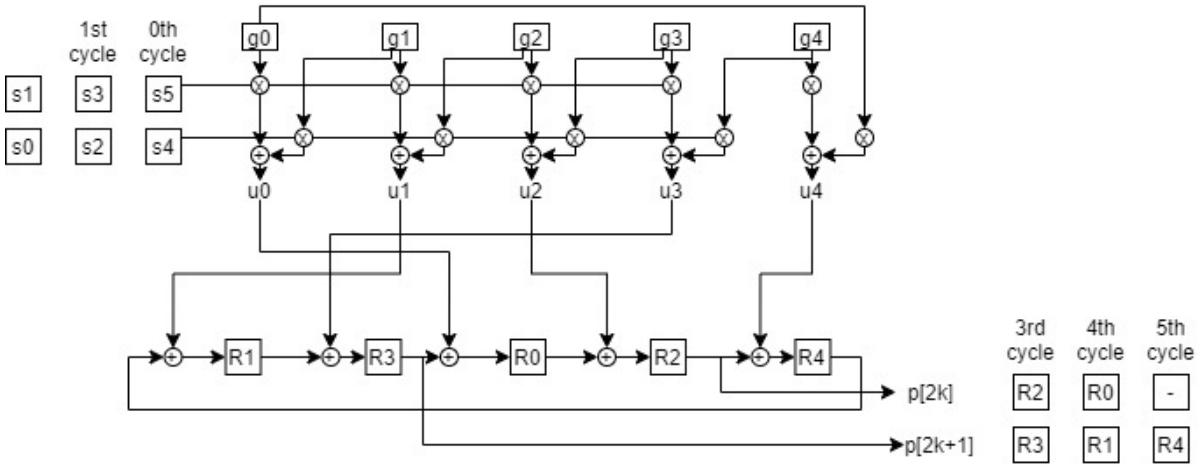
We should expect at k=3rd cycle, register values of R3 and R2, and at 4th cycle , register values of R1 and R0 and at cycle 5th cycle, register values of R4 as shown:

$cycles(k)$: 5 4 3 5 4 3

$$R_4 \rightarrow R_1 \rightarrow R_3 \rightarrow p1[k] = p_0 \rightarrow p_2 \rightarrow p_4$$

$$0 \rightarrow R_0 \rightarrow R_2 \rightarrow p0[k] = 0 \rightarrow p_1 \rightarrow p_3$$

Take outputs p[2k+1], p[2k] in parallel from register output R3 and R2 and the output parity bits will be shifted out in order from these points (nodes e and d).



To take parity bits in parallel, the relation $\boldsymbol{p} = (\boldsymbol{R}^3)^{r(1)}$ is to be considered. Appropriate alignment in circulant bits can make P=R, as discussed in previous section (Case2a under the Rule for output parity bits or circulant Bits).

For arbitrary Lm, and nodes $N = \{\, n_0, n_1, n_2, \ldots, n_a, \ldots, n_{Z-1}\,\}$

Precedence rule with w=1 delay elements ($\{R_0, R_1, R_2, ..., R_a, ..., R_{Z-1}\}$:

$$n_0 \rightarrow R_0 \rightarrow n_1 \rightarrow R_1 \rightarrow ... \rightarrow n_{Z-1} \rightarrow R_{Z-1} \rightarrow n_0$$

Replication by j=Lm gives: $N^0 = \{n_0^0, n_1^0, ..., n_a^0, ..., n_{Z-1}^0\}, N^1, N^2, .., N^{Lm-1}$

Starting node index=u	Connect to node index (u+w)%j	Delay elements=Floor((u+w)/j)	Connection precedence
n_0^0	(0+1)%Lm=1 => n_1^1	(0+1)/Lm=0=>0 delay	$n_0 \rightarrow n_1$
n_1^0	n_2^1	0	$n_1 \rightarrow n_2$
n_2^0	n_3^1	0	…
…	…	0	…
n_{Z-1}^0	n_0^1	0	$n_{Z-1} \rightarrow n_0$
n_0^1	$n_1^{(1+1)\%Lm}$	(1+1)/Lm delay : R_0	$n_0 \rightarrow n_1$
n_1^1	$n_2^{(1+1)\%Lm}$	(1+1)/Lm delay	$n_1 \rightarrow n_2$
n_2^1	d0	1 delay	…
…	e0	1 delay	…
n_{Z-1}^1	a0	1 delay	$n_{Z-1} \rightarrow n_0$
e0	(0+0)%2=0=>p0	(0+0)/2=0=>0 delay	e->p[n]
e1	(1+0)%2=1=>p1	(1+0)/2=0=>0 delay	e->p[n]

NE Encoder Verilog Design:

Design Goals:

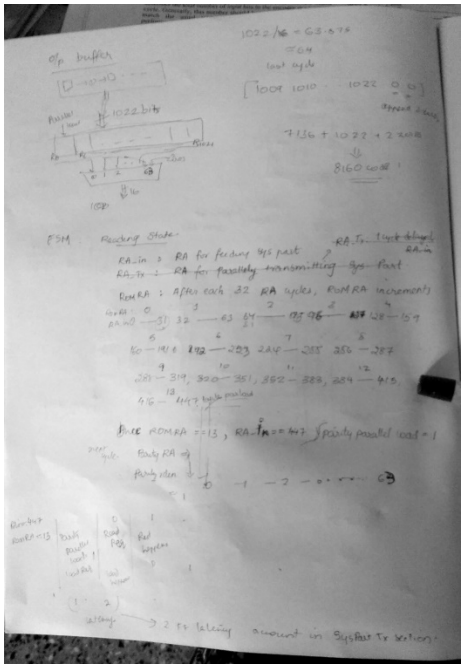
Low power is preferred for Encoder, since encoder is located at space or satellite.

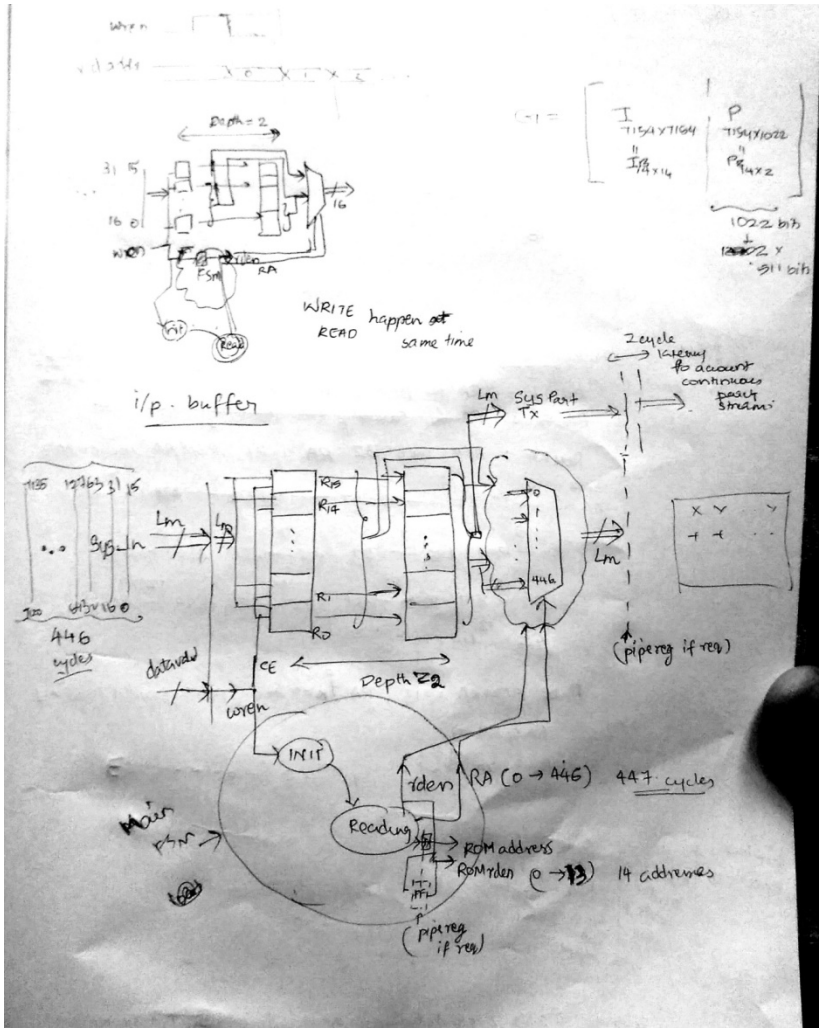
But Gbps throughput has to be achieved atleast 1.6Gbps.

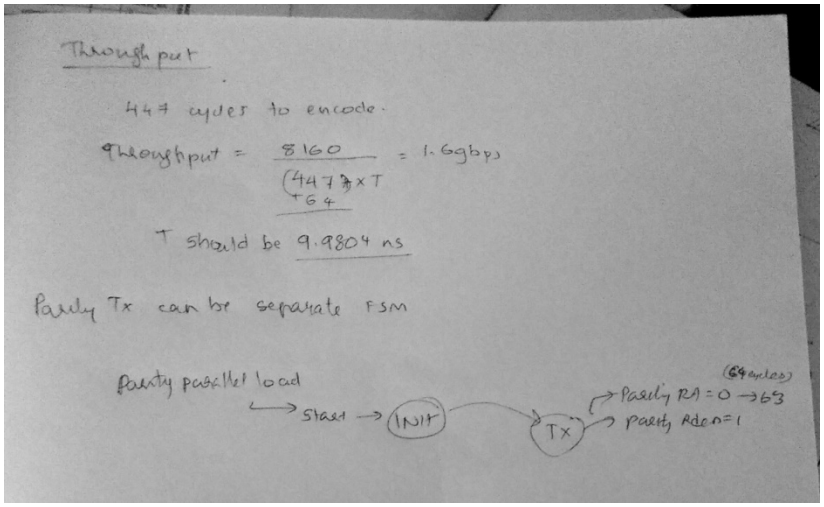
Design of interfaces and Memory

Initial design: 1 bit to 7136 bits output (SIPO) followed by a local memory for all bits : Following initial design.

New design: (not followed, draft idea, reference model not validated.)




Depth = 2
WRITE happen
READ same time
i/p buffer
Lm
Sys Part Tx
2 cycle latency to account continuous pass stream
446 cycles
Depth = 2
(pipe reg if req)
INIT
RA (0 → 446)
447 cycles
Reading
ROM address
ROM rden
14 addresses
(pipe reg if req)



Local memory:

~~Shift register type hardwired (scripted) memory circuit. (Not followed).~~

~~Input 16 bits, output 16 bits.~~

Input 7136 bits parallel to 16 bits output

For each address, provide those bits in the order:

Address0 : Take first circulant's 511 bits (assuming it includes 18 zeros) ={0,1,2,.., 17, 18,…,511}, in verilog it is stored in Big-endian= msg511_1 = {511, …, 18, 17, …, 2, 1, 0}.

take MSB Lm bits = {511, .., 511-16+1} = msg16_1 = {Lm-1, …, 0}. Then bit rev it : msg16rev_1 = {511-16+1, …, 511} = {Lm-1, …, 0}. Feed msg16rev_1 to encoding network.

Current Design of Encoder

Reference Model

encodervalidation.m

Verilog Design:

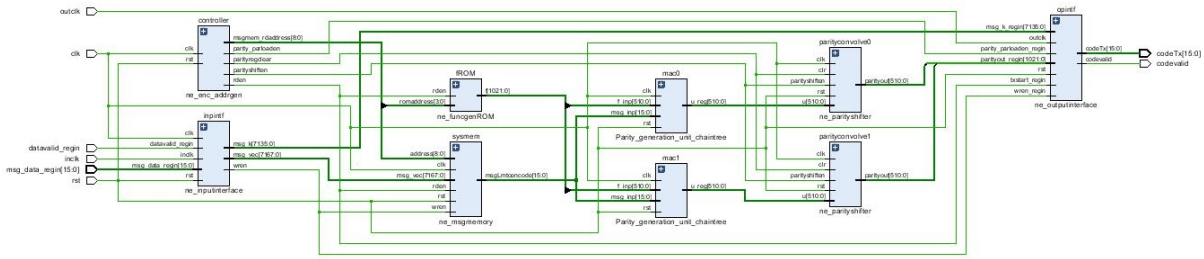


Fig. ne_enc_top_v0

Features:

1. Based on reference model of script: encodervalidation.m
2. ROM script: ne_funcgenROM.m
3. Message bits fed to MAC network without removing the 18 zeros.
4. Clock sync logic provided for Input interface and output interface
5. Fully parallel loading (synthesis check to do)
6. Memory completely made of FF and shift reg circuits (No RAMs)
7. Expected cycles:
    a. input: 7136/16 = 446 inclk cycles
    b. encoding: 7168/16 = 448 clk cycles + 1 (parload of result) + other misc
    c. output: 8160/16 = 510 outclk cycles
8. Rough Throughput based Timeperiod/Critical path delay requirement :
    a. 8160/(446*1.6gbps) -> Tinclk = 11.435ns
    b. 8160/(510*1.6gbps) -> Toutclk = 10ns
    c. 8160/(460*1.6gbps) -> Tclk approx.. = 11.087ns

Design of Randomizer and ASM sync Marker attacher

(not yet designed)

Randomizer and ASM sync Marker attachment is to be added in Output interface.

It may require possible modifications in FSM of Output interface also.

Verification of Encoder:

Without interface or memory: (not needed now)
Testcases:

1. Feed circulant bits for Z=5
2. Feed the 5 message bits in parts of Lm=2
3. Ex; f = {5'b11001}, msg={5'b10001} = 6'b010001, msg_0 = {2'b01}, msg_1={2'b00}, msg_2={2'b01}.
4. See if matlab reference for product of f and msg is obtained in TB simulation.

without interface: (not needed now)
reference model in MATLAB -> reference codeword bits(8176 bits) and sample input message bits(7154), and TB in vivado.

Test Plan:

1. Feed only first 511 message bits and first circulant bits. Check the output matching with the MAtlab output
2. Feed (7154 + 14zeros - first 16zeros) message bits

with interface:
reference model in MATLAB-> reference codeword bits(8160 bits) and sample input message bits (7136) and TB in vivado

Stimulus pattern:

Feed the 7136 message bits in groups of 16 bits from LSB onwards:
D16_# = { D15, D14, …, D1, D0}
cycles = 7136/16 = 446

cycle0: D16_0
cycle1: D16_1
…
cycle445: D16_445

Sample Clock periods:

create_clock -period 11.435 -name inclk [get_ports inclk]
create_clock -period 5.000 -name clk [get_ports clk]
create_clock -period 10.000 -name outclk [get_ports outclk]

Test Cases:

1. Input Interface outputs and Message memory bit pattern should match that with reference model
    a. msg_vec : should be arranged in following bit pattern 7154
    b. K+18+14 in length: Each 512 bits are in reversed form with MSB being zero.
2. Message memory read operation: Lm bit pattern should be matching with reference model message pattern.
3. ROM address output should be circulant row shifted right by (2 + circulant index-1), where circulant index varies from 0 to 13, corresponding to circulant block fetched.
4. u values : Output of Parity Generation network (MAC network) should match the reference model values.
5. parity register values: update at the next cycle. After 448 +1 cycle, the parity register values read in the normal order : parityout_regin [2*Z-1:0] should have the same value as observed in reference model i.e. parityout_regin [i] == prconcatenate [i].
6. Controller features to be tested:
    a. The rden signal issued form controller is used as transmit start for output interface. This signal stays 1 for long time, to be captured by the outclk.
    b. The rden signal should initiate the transmission of systematic bits while encoding is going on.

    c. After encoding count is finished, shifting is disabled for parity shift reg by asserting parityshiften, at the same time parity_parloaden is asserted.
    d. To make the parity values stay constant for more cycles of clk (faster) to be captured by outclk (slower)
7. Output interface:
    a. There should be no gap between transmission of systematic bits and parity bits.
    b. There should be 2 trailing zeros at the end of 1022 parity bits.
    c. Total cycles = 8160/16
8. Measure Number of cycles of inclk, clk, outclk

Synthesis Check and Design optimisations

See the synthesis results, if it is suitable for:

1. Kintex7: xc7k160t
2. Kintex ultrascale plus: xcku5p
3. Virtex ultrascale plus:VCU118 / xcvu9p

FPGA device	Resources	Power	Tinclk Tc	Tclk Tc	Toutclk Tc