101.3.2.4 FEC encoding process

The {EPoC_PMD_Name} encodes the transmitted data using a systematic Low-Density Parity-Check (LDPC) (F_C, F_P) code. A LDPC encoder encodes F_P information bits $i_{0 \ldots i_{F_P-1}}$ into a codeword

$$c = (i_0,\ldots, i_{F_P-1}, p_{F_P}, \ldots, p_{F_C-1})$$

by adding F_R parity bits $p_{F_p \ldots p_{F_C-1}}$ obtained so that

$$Hc^T = 0$$

where H is an $F_R \times F_C$ binary matrix containing mostly '0' and relatively few '1', called low-density parity-check matrix. (see [1] and [2]). The detailed description of such parity check matrices is given in 101.3.2.4.1.

{to be included in informative references: [1] R. G. Gallager, "Low density parity check codes," IRE Trans. Inform. Theory, vol. IT-8, pp. 21–28, Jan. 1962.; [2] T. Richardson and R. Urbanke, "Modern Coding Theory," Cambridge University Press, 2008}

The CLT {EPoC_PMD_Name} PCS operating on active CCDN shall encode the transmitted data using one of the LDPC (FC, FP) codes per Table 101–1, as selected using register TBD. The CNU {EPoC_PMD_Name} PCS operating on active CCDN shall encode the transmitted data using one of the LDPC (FC, FP) codes per Table 101–2, as selected using register TBD.

Table 101–1—LDCP codes used by the CLT {EPoC_PMD_Name} PCS for active CCDN

Codeword F_C [bits]	Payload F_P [bits]	Parity F_R [bits]	Payload		Parity		
			65-bit blocks B_Q	Padding bits B_P	64-bit blocks C_Q	Parity bits in last block C_{PL}	Padding bits C_P
16200	14400	1800	220	59	28	31	34

Annex 101A gives an example of LDPC (F_C, F_P) FEC encoding. {we will need to select one of the codes from the family of codes we use in either downstream or upstream and then generate examples}

101.3.2.4.1 LDPC matrix definition

The low-density parity check matrix H for LDPC (F_C, F_P) encoder can be divided into blocks of L^2 sub-matrices. Its compact circulant form is represented by an m × n block matrix:

where the submatrix $H_{i,j}$ is an L × L all-zero submatrix or a cyclic right-shifted identity submatrix. The last n–m sub-matrix columns represent the parity portion of the matrix. Moreover, $nL = F_C$, $mL = F_P$ and the code rate is $(n–m)/n = (F_C–F_P)/F_C$. In this specification, the sub-matrix size L is called the lifting factor.

$$H = \begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} & \dots & H_{1,n} \\ H_{2,1} & H_{2,2} & H_{2,3} & \dots & H_{2,n} \\ H_{3,1} & H_{3,2} & H_{3,3} & \dots & H_{3,n} \\ \dots & \dots & \dots & & \dots \\ H_{m,1} & H_{m,2} & H_{m,3} & \dots & H_{m,n} \end{bmatrix}$$

Table 101–2—LDCP codes used by the CLT {EPoC_PMD_Name} PCS for active CCDN

Codeword F_C [bits]	Payload F_P [bits]	Parity F_R [bits]	Payload		Parity		
			65-bit blocks B_Q	Padding bits B_P	64-bit blocks C_Q	Parity bits in last block C_{PL}	Padding bits C_P
16200	14400	1800	220	59	28	31	34
5940	5040	900	76	59	14	27	38
1120	840	280	12	19	4	47	18

In this specification, the sub-matrix $H_{i,j}$ is represented by a value in {-1, 0,…, L-1}, where a '-1' value represents an all-zero submatrix, and the remaining values represent an L by L identity submatrix cyclically right-shifted by the specified value. Such representation of the parity-check matrix is called a base matrix.

{The following matrices were extracted from http://www.ieee802.org/3/bn/public/jul13/prodan_3bn_01b_0713.pdf, as updated. This material with technical changes has not been yet adopted as baseline proposal.}

Table 101–3a through Table 101–3c present a 5 × 45 base matrix of the low-density parity-check matrix H for LDPC (16200, 14400) code listed in Table 101–1 for downstream and Table 101–2 for upstream, respectively. The lifting factor of the matrix is L=360.

Table 101–3a—LDPC (16200, 14400) code matrix, columns 1-15

Row	Column														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	93	271	-1	83	26	208	245	200	-1	175	331	17	86	-1	337
2	274	115	329	338	124	-1	293	-1	69	64	342	-1	88	139	-1
3	134	355	175	24	253	242	-1	187	94	26	87	302	-1	191	323
4	-1	-1	184	70	247	14	22	7	285	54	-1	352	26	108	10
5	253	273	90	-1	-1	151	311	320	339	-1	295	148	48	91	62

Table 101–3b—LDPC (16200, 14400) code matrix, columns 16-30

Row	Column														
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
1	-1	238	81	-1	307	-1	165	-1	47	76	73	150	349	139	331
2	137	212	-1	157	195	357	81	194	1	159	56	72	126	277	156
3	22	-1	245	294	240	84	76	342	345	174	269	329	-1	214	-1
4	298	123	139	117	-1	336	49	202	359	342	-1	224	106	-1	273
5	100	232	146	200	135	12	-1	179	-1	-1	232	-1	21	331	313

Table 101–3c—LDPC (16200, 14400) code matrix, columns 31-45

Row	Column														
	31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**
1	118	345	27	294	-1	145	279	97	106	160	143	-1	-1	-1	-1
2	32	111	175	-1	306	224	-1	206	-1	29	106	334	-1	-1	-1
3	-1	-1	-1	218	104	40	197	73	229	63	-1	270	72	-1	-1
4	177	245	98	355	178	176	147	-1	280	-1	-1	-1	221	208	-1
5	349	34	97	187	38	-1	235	52	170	58	-1	-1	-1	257	0

Table 101–4a through Table 101–4c present a 5 × 33 base matrix of the low-density parity-check matrix H for LDPC (5940, 5040) code listed in Table 101–2 for upstream. The lifting factor of the matrix is L=180.

Table 101–4a—LDPC (5940, 5040) code matrix, columns 1-11

Row	Column										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	142	158	113	124	92	44	93	70	172	3	25
2	54	172	145	28	55	19	159	22	96	12	85
3	63	11	112	114	61	123	72	55	114	20	53
4	28	160	102	44	8	84	126	9	169	174	147
5	52	159	75	74	46	71	42	11	108	153	-1

Table 101–5a and Table 101–4c present a 5 × 20 base matrix of the low-density parity-check matrix H for LDPC (1120, 840) code listed in Table 101–2 for upstream. The lifting factor of the matrix is L=56.

Table 101–4b—LDPC (5940, 5040) code matrix, columns 12-22

Row	Column										
	12	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
1	44	141	160	50	45	118	84	-1	64	66	97
2	-1	128	5	158	120	51	171	65	141	-1	42
3	114	42	33	4	66	163	50	46	17	175	-1
4	24	145	-1	26	-1	-1	-1	67	82	4	177
5	72	-1	163	-1	9	2	168	158	-1	1	49

Table 101–4c—LDPC (5940, 5040) code matrix, columns 23-33

Row	Column										
	23	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**
1	1	115	8	108	-1	-1	22	-1	-1	-1	-1
2	83	7	-1	39	121	84	101	171	-1	-1	-1
3	-1	-1	92	-1	41	138	-1	34	74	-1	-1
4	151	131	139	117	36	18	-1	-1	23	8	-1
5	89	63	179	10	75	161	-1	-1	-1	177	19

Table 101–5a—LDPC (1120, 840) code matrix, columns 1-10

Row	Column									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	5	14	12	1	2	37	45	26	24	0
2	0	35	1	26	0	10	16	16	34	4
3	12	28	22	46	3	16	51	2	25	29
4	0	51	16	31	13	39	27	33	8	27
5	36	6	3	51	4	19	4	45	48	9

101.3.2.4.2 LDPC encoding process within CLT (downstream)

The process of padding FEC codewords and appending FEC parity octets in the {EPoC_PMD_Name} CLT transmitter is illustrated in Figure 101–1. The 64B/66B encoder produces a stream of 66-bit blocks, which are then delivered to the FEC encoder. The FEC encoder accumulates B_Q (see Table 101-1) of these 66-bit blocks to form the payload of a FEC codeword, removing the redundant first bit (i.e., sync header bit <0>) in each 66-bit block received from the 64B/66B encoder. The first bit <0> of the sync header in the 66-bit block in the transmit direction is guaranteed to be the complement of the second bit <1> of the sync header – see 49.2.4.3 for more details.

Table 101–5b—LDPC (1120, 840) code matrix, columns 11-20

Row	Column									
	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	3	-1	34	7	46	10	-1	-1	-1	-1
2	2	23	0	51	-1	49	20	-1	-1	-1
3	19	18	52	-1	37	-1	34	39	-1	-1
4	53	13	-1	52	33	-1	-1	38	7	-1
5	-1	11	22	23	43	-1	-1	-1	14	-1

Next, the FEC encoder calculates CRC40 over the aggregated B_Q 65-bit blocks, placing the resulting 32 bits of CRC32 code prepended with one bit truncated sync header (with the binary value of "1") immediately after the B_Q 65-bit blocks, forming the payload of the FEC codeword. Finally, the FEC encoder prepends B_P (see Table 101–1) padding bits (with the binary value of "0") to the payload of the FEC codeword as shown in Figure 101–1. This data is then LDPC-encoded, resulting in the F_R bits of parity data. The first 32 bits of parity data are inserted into the 65-bit block carrying CRC40 code, complementing it. The remaining F_{R-32} bits of parity data is then divided into C_Q 64-bit blocks, each of which is then prepended with one bit sync header <1> with the value of binary "1". The last 64-bit block of the parity data contains C_{PL} bits of parity data, and the remaining C_P bits are filled with padding (binary "0").

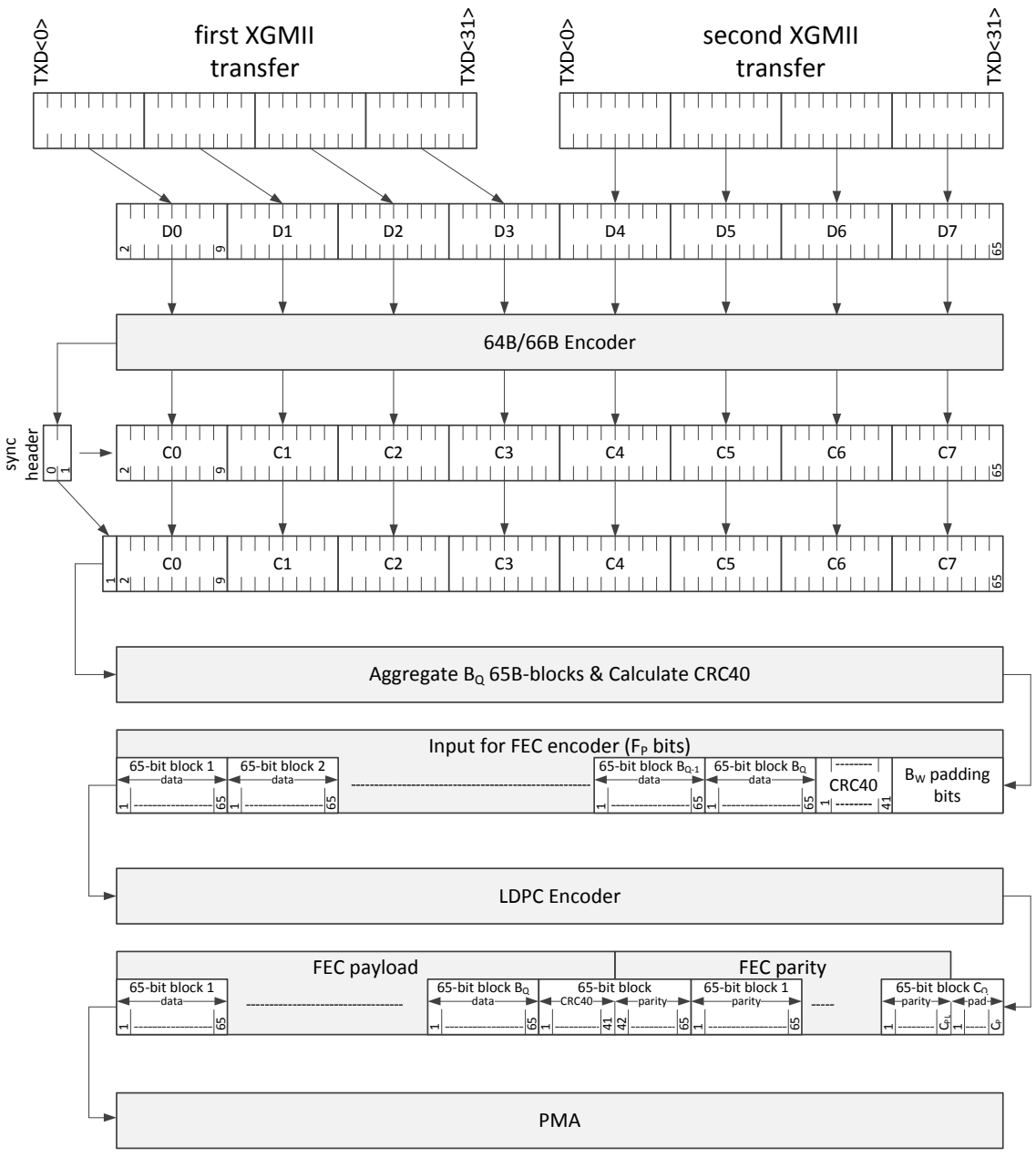

Figure 101–1—PCS Transmit bit ordering within CLT (downstream)

101.3.2.4.3 LDPC codeword transmission order within CLT (downstream)

Once the process of calculating FEC parity is complete, the payload portion of the FEC codeword and the parity portion of the FEC codeword are then transferred towards the PMA across the PMA service interface, one 65-bit block at a time. Note that the B_P padding bits used to generate the FEC codeword are not transmitted across the PMA service interface. The C_P padding bits in the last parity codeword (block number C_Q) are transmitted to PMA, where they are the discarded prior to encoding into OFDM medium.

101.3.2.4.4 LDPC encoding process within CNU (upstream)

{the upstream FEC encoding for CNU will be described when we have a consistent proposal on how to mix three different FEC codes into a single transmission slot}

101.3.2.4.5 LDPC codeword transmission order within CNU (upstream)

{the content of this subclause ought to be quite similar with the content of 101.3.2.4.3}

101.3.2.4.6 State diagrams

101.3.2.4.6.1 Constants

B_Q

VALUE: see Table 101–1 for downstream FEC, Table 101–2 for upstream FEC
This constant represents the number of 65-bit blocks within the payload portion of the FEC codeword.

C_P

VALUE: see Table 101–1 for downstream FEC, Table 101–2 for upstream FEC
This constant represents the number of padding bits within the last 65-bit block of the parity portion of the FEC codeword.

C_Q

VALUE: see Table 101–1 for downstream FEC, Table 101–2 for upstream FEC
This constant represents the number of 65-bit blocks within the parity portion of the FEC codeword.

F_P

VALUE: see Table 101–1 for downstream FEC, Table 101–2 for upstream FEC
This constant represents the number of bits within the payload portion of the FEC codeword.

F_R

VALUE: see Table 101–1 for downstream FEC, Table 101–2 for upstream FEC
This constant represents the number of bits within the parity portion of the FEC codeword.

101.3.2.4.6.2 Variables

blockCount

TYPE: 16-bit unsigned integer
This variable represents the number of data blocks (either 65-bit or 66-bit blocks, depending on the state diagram).

CLK

TYPE: Boolean
This Boolean is true on every negative edge of TX_CLK (see 46.3.1) and represents instances of time at which a 66-bit block is passed from the output of the 64B/66B encoder into the FEC encoder. This variable is reset to false upon read.

dataPayload<F_P-1:0>

TYPE: Bit array
This array represents the payload portion of the FEC codeword, accounting for the necessary padding. It is initialized to the size of F_P bits and filled with the binary value of "0".

dataParity<F_R-1+C_P:0>

TYPE: Bit array

This array represents the parity portion of the FEC codeword, accounting for the necessary padding. It is initialized to the size of $F_R + C_P$ bits and filled with the binary value of "0".

FIFO_FEC

TYPE: Array of 65-bit blocks
A FIFO array used to store tx_coded<65:1> blocks, inserted by the input process in the FEC encoder, while FEC parity is sent out towards PMA.

loc

TYPE: 16-bit unsigned integer
This variable represents the position within the bit array, indicating how much data is stored within the given bit array.

rx_coded_in<64:0>

TYPE: 65-bit block
This 65-bit block contains the input of the FEC decoder being received from PMA. The left-most bit is rx_coded_out<0> and the right-most bit is rx_coded_out<64>.

sizeFifo

TYPE: 16-bit unsigned integer
This variable represents the number of 65-bit blocks stored in FIFO_FEC.

tx_coded<65:0>

TYPE: 66-bit block
This 66-bit block contains 64B/66B encoded data. The format for this data block is shown in Figure 49–7. The left-most bit in the figure is tx_coded<0> and the right-most bit is tx_coded<65>.

tx_coded_out<64:0>

TYPE: 65-bit block
This 65-bit block contains the output of the FEC encoder being passed into PMA. The left-most bit is tx_coded_out<0> and the right-most bit is tx_coded_out<64>.

101.3.2.4.6.3 Functions

calculateCrc (ARRAY_IN)

This function calculates CRC40 for data included in ARRAY_IN.

calculateParity(ARRAY_IN)

This function calculates LDPC parity (for the code per Figure 101–1 or Figure 101–2) for data included in ARRAY_IN.

resetArray(ARRAY_IN)

This function resets the content of ARRAY_IN, setting all the elements in this array to the binary value of "0".

removeFifoHead(FIFO_IN)

This function removes the first block in FIFO_IN and decrements the variable sizeFifo by 1.

```
removeFifoHead( FIFO_IN )
{
FIFO_IN[0] = FIFO_IN[1]
FIFO_IN[1] = FIFO_IN[2]
...
FIFO_IN[sizeFifo-2] = FIFO_IN[sizeFifo-1]
sizeFifo --
}
```

101.3.2.4.6.4 Messages

TBD

101.3.2.4.6.5 State diagrams

The CLT PCS shall implement the LDPC encoding process, comprising the input process as shown in Figure 101–6 and the output process as shown in Figure 101–7. The CNU PCS shall implement the LDPC encoding process, comprising the input process as shown in Figure 101–6 and the output process as shown in Figure 101–7.

In case of any discrepancy between state diagrams and the descriptive text, the state diagrams prevail.

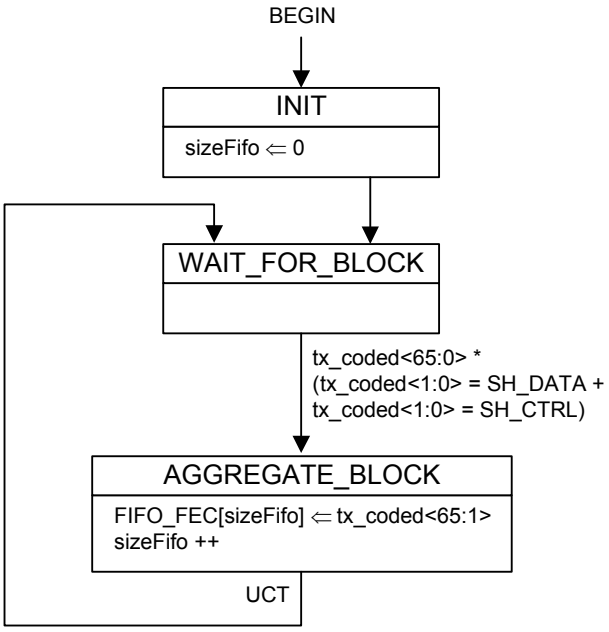

Figure 101–6—FEC encoder, input process state diagram

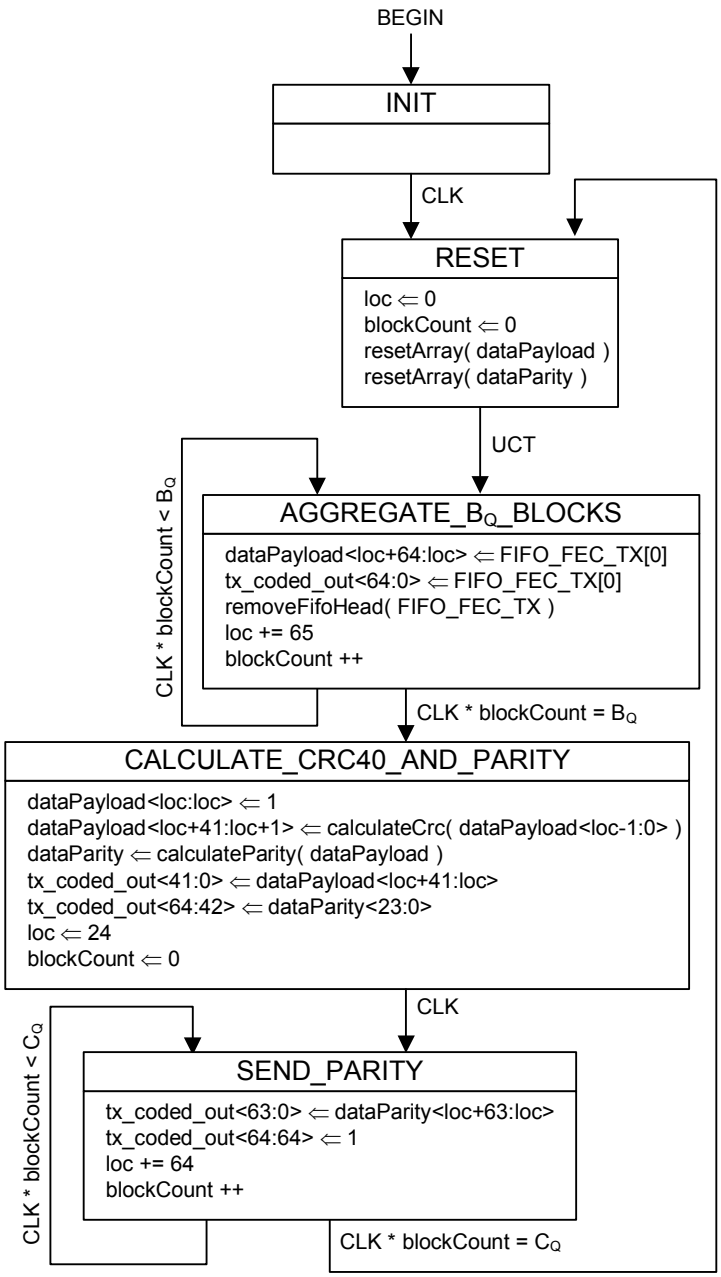

Figure 101–7—FEC encoder, output process state diagram (CLT)

101.3.3.3 FEC decoding process

The {EPoC_PMD_Name} decodes the received data using LDPC (F_C, F_P) code. The CLT {EPoC_PMD_Name} PCS operating on active CCDN shall decode the received data using one of the LDPC (F_C, F_P) codes per Table 101–2, as selected using register TBD. The CNU {EPoC_PMD_Name} PCS oper-

ating on active CCDN shall decode the received data using one of the LDPC (F_C, F_P) codes per Table 101–1, as selected using register TBD.

Annex 101B gives an example of LDPC (F_C, F_P) FEC decoding. {we will need to select one of the codes from the family of codes we use in either downstream or upstream and then generate examples}

101.3.3.3.1 LDPC decoding process within CLT (upstream)

{the upstream FEC decoding for CLT will be described when we have a consistent proposal on how to mix three different FEC codes into a single transmission slot}

101.3.3.3.2 LDPC decoding process within CNU (downstream)

The process of decoding FEC codewords in the {EPoC_PMD_Name} CNU receiver is illustrated in Figure 101-2.

{FEC codeword alignment needs to be tackled somewhere between the PMA and the bottom of the PCS – we had some proposals on how to find FEC codeword lock in the downstream, but I am not sure we baselined anything with sufficient level of detail to actually put it into the draft}

Once the alignment to FEC codeword is found, the {EPoC_PMD_Name} CNU receiver aggregates the total of $B_Q + 1 + C_Q$ 65-bit blocks received from the PMA, forming the FEC payload (blocks number 1 to B_Q, and bits <0> through <41> from the following 65-bit block) and the FEC parity (bits <42> through <64> from the 65-bit block following payload portion of the FEC codeword and followed by blocks number 1 to C_Q) portions of the codeword. Note that the CP padding bits in the last parity codeword (block number C_Q) are locally generated within the PMA and transmitted to the PCS.

Next, each 65-bit block in the FEC parity portion of the codeword (blocks 1 through C_Q) is stripped from the sync header by removing bit <1>. Furthermore, the last 64-bit block of the FEC parity (block number C_Q) is truncated, removing bits <C_{PL}> … <63>, forming a single FEC parity portion of the codeword with size F_R (in bits).

Then the FEC payload portion of the codeword is prepended with B_P padding bits (with the binary value of "0") and subsequently fed into the FEC decoder for processing together with the stripped FEC parity portion of the codeword.

The FEC decoder produces the FEC payload portion of the codeword with the size of F_P (in bits), where bits <0> … <B_P-1> contain padding (with the binary value of "0"). Next, the CRC40 is calculated over the remaining blocks 1 through B_Q and then compared with the value of CRC40 retrieved from the received FEC codeword. If both CRC40 codes match, the decoded frame does not contain any detectable errors and it is treated as error-free. Otherwise, the decoded frame contains detected errors. The behavior of the FEC decoder in the presence of CRC40 code failure depends on status of the user-configurable option to indicate an uncorrectable FEC codeword.

Finally, the FEC decoder prepends each of the B_Q 65-bit blocks with bit <0> of the sync header containing the binary inverse of the value carried in bit <1> of the sync header, producing 66-bit blocks. This also guarantees that properly decoded blocks meet the requirements of 49.2.4.3.

The FEC decoder in the CNU shall provide a user-configurable option to indicate an uncorrectable FEC codeword (due to an excess of symbols containing errors) to higher layers. If this user-configurable option is enabled and the calculated value of CRC40 does not match the value of CRC40 retrieved from the received FEC codeword, the FEC decoder replaces bit <0> and <1> in the sync headers in all B_Q blocks with the binary value of "11". If this user-configurable option is disabled, the FEC decoder does not make any further changes to the sync headers in all B_Q blocks.

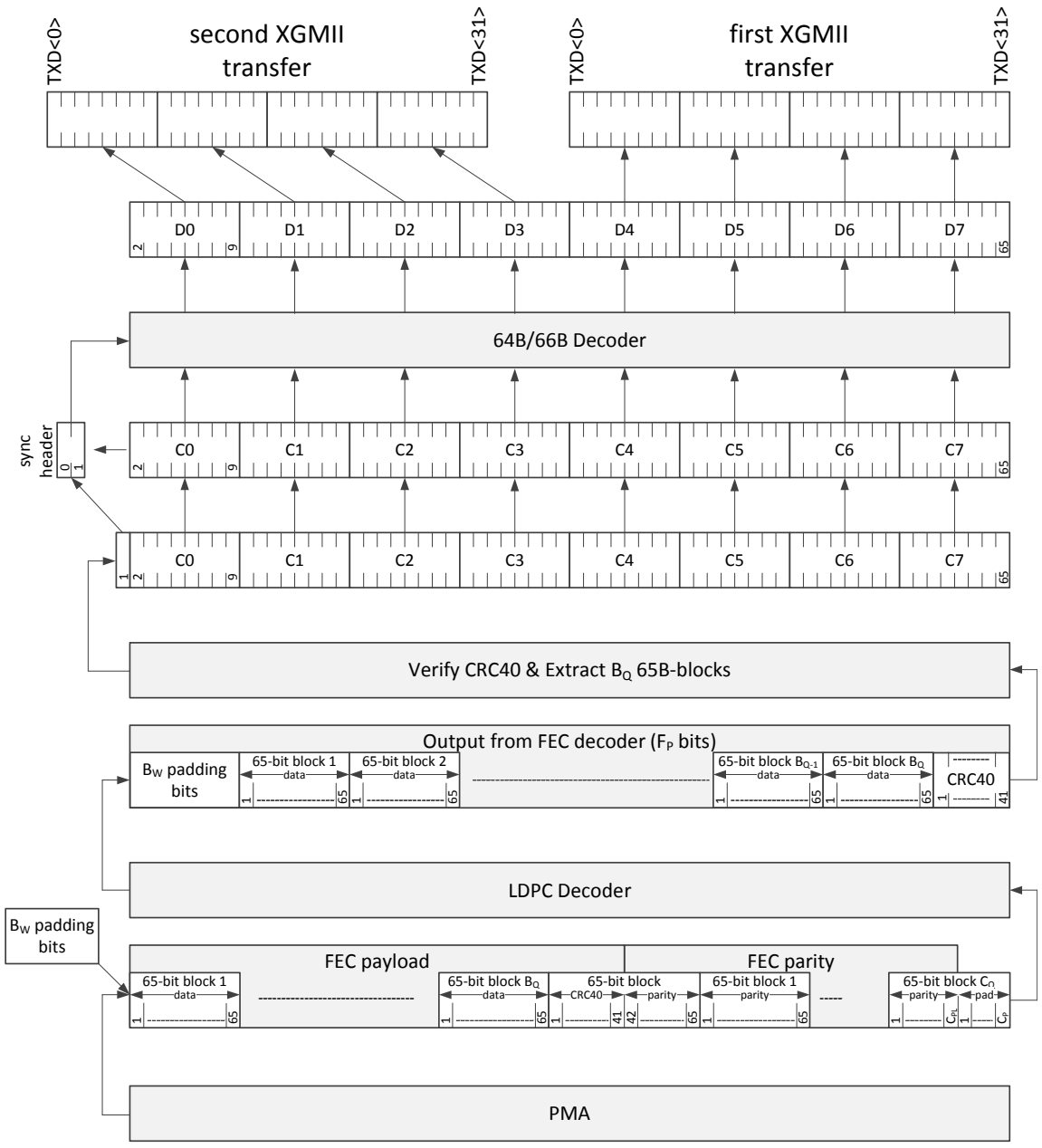

Figure 101–8—PCS Receive bit ordering within CNU (downstream)

Each resulting 66-bit block is then fed into the 64B/66B decoder, removing the sync header information (bit <0> and bit <1>), which is used to generate control signaling for the XGMII. Finally, the resulting 64-bit block is then separated into two 32-bit portions, which are transmitted across the XGMII on two consecutive transfers, with the proper control signaling retrieved from the sync header information retrieved in the 64B/66B decoder.

101.3.3.3.3 State diagrams

101.3.3.3.3.1 Constants

B_Q
see 101.3.2.4.6

B_W
VALUE: see Table 101–1 for downstream FEC, Table 101–2 for upstream FEC
This constant represents the number of padding bits within the payload portion of the FEC codeword.

C_Q
see 101.3.2.4.6

IDLE
TYPE: 66-bit vector
This constant represents /I/ character with 64B/66B encoding, as defined in 49.2.4.7.

101.3.3.3.3.2 Variables

blockCount
see 101.3.2.4.6

CLK
see 101.3.2.4.6

dataCrc<31:0>
TYPE: Bit array
This array represents the CRC40. It is initialized to the size of 40 bits and filled with the binary value of "0".

dataIn<F_C+C_P-1:0>
TYPE: Bit array
This array represents the combination of the payload portion of the FEC codeword, the parity portion of the FEC codeword, CRC40, and all the necessary padding. It is initialized to the size of F_C+C_P bits and filled with the binary value of "0".

dataOut<F_P-1:0>
TYPE: Bit array
This array represents the combination of the payload portion of the FEC codeword, CRC40, and all the necessary padding. It is initialized to the size of F_P bits and filled with the binary value of "0".

FIFO_FEC_RX
TYPE: Array of 66-bit blocks
A FIFO array used to store tx_coded<65:0> blocks, inserted by the input process in the FEC decoder, while encoded data is then sent to 64B/66B decoder for processing and transmission towards the XGMII.

loc
see 101.3.2.4.6

rx_coded_in<64:0>
TYPE: 65-bit block
This 65-bit block contains the input into the FEC decoder being passed from PMA. The left-most bit is rx_coded_in<0> and the right-most bit is rx_coded_in<64>.

sizeFifo

see 101.3.2.4.6

syncFec

TYPE: Boolean
This variable indicates whether the FEC codeword alignment was found (value equal to *true*) or not (value equal to *false*).

tx_coded<65:0>

see 101.3.2.4.6

101.3.3.3.3.3 Functions

calculateCrc (ARRAY_IN)

see 101.3.2.4.6

decodeFec(ARRAY_IN)

This function performs FEC decoding (for the code per Figure 101–1 or Figure 101–2) for data included in ARRAY_IN.

resetArray(ARRAY_IN)

see 101.3.2.4.6

101.3.3.3.3.4 Messages

TBD

101.3.3.3.3.5 State diagrams

The CNU PCS shall implement the LDPC decoding process, comprising the input process as shown in Figure 101–6 and the output process as shown in Figure 101–7.

In case of any discrepancy between state diagrams and the descriptive text, the state diagrams prevail.

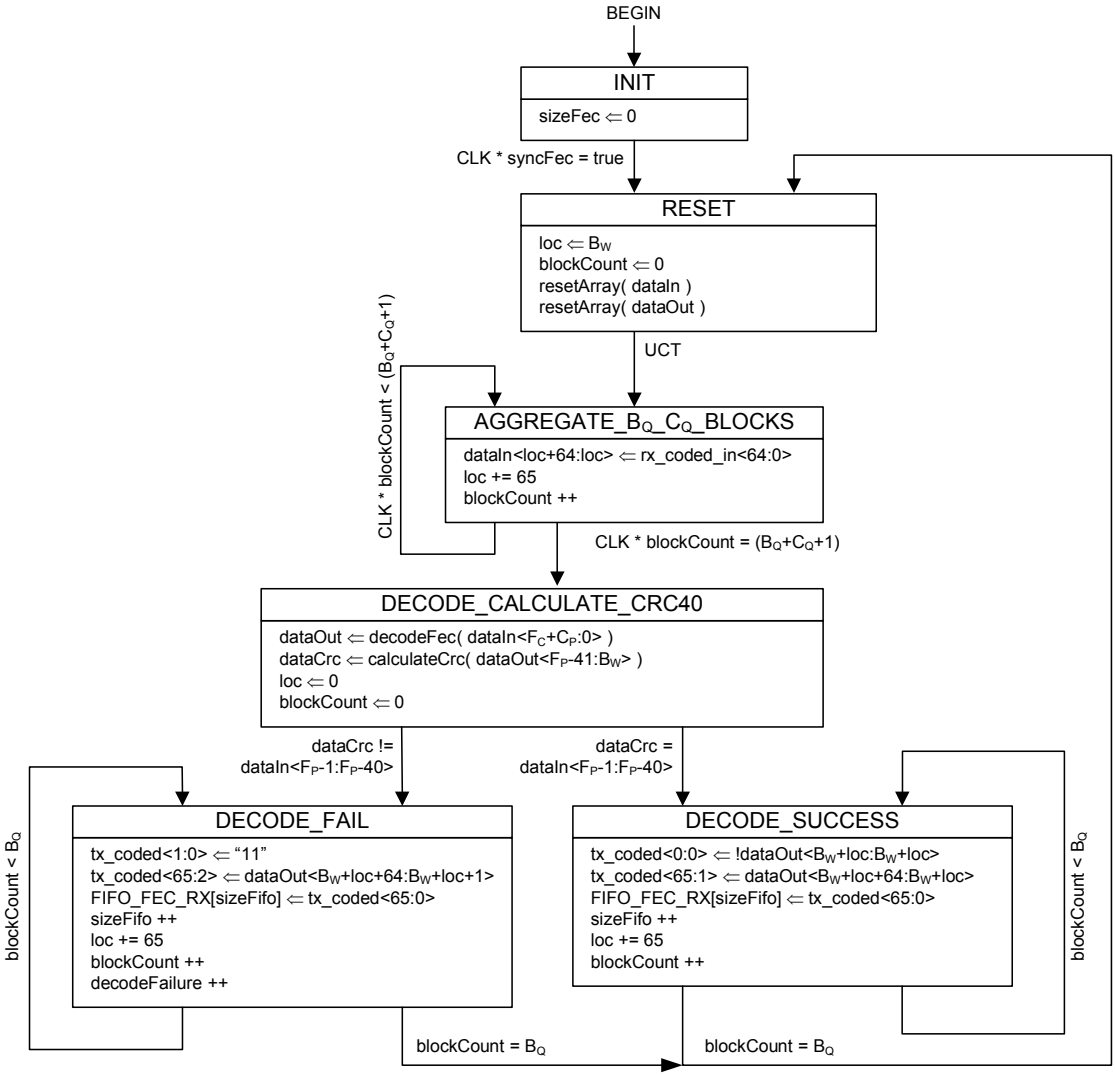

Figure 101–6—FEC decoder, input process state diagram (CNU)

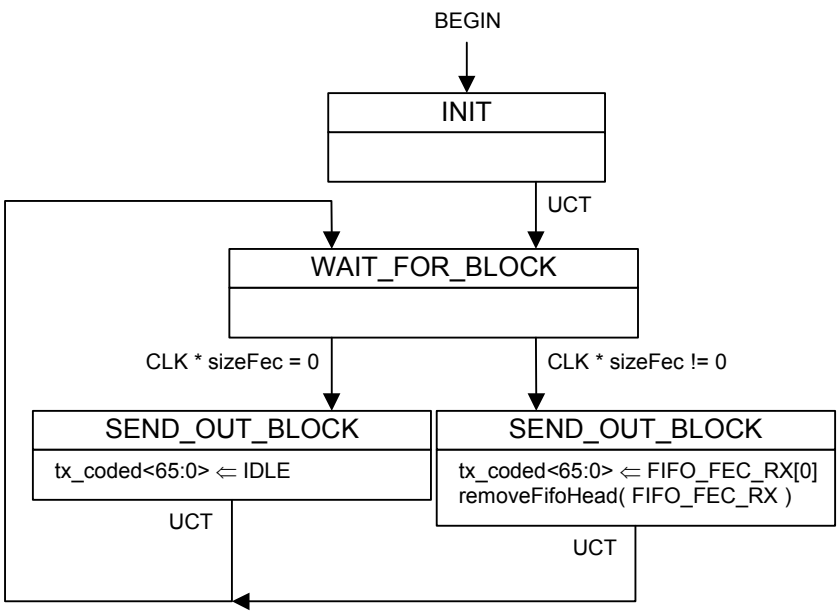

Figure 101–7—FEC decoder, output process state diagram (CNU)