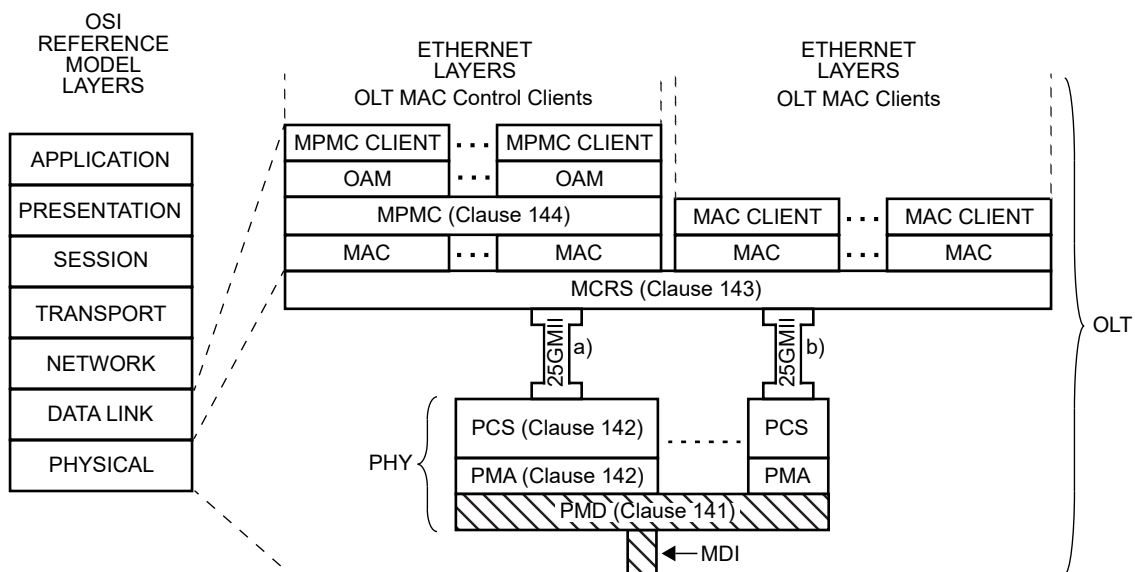


25GMII=25 GIGABIT MEDIA INDEPENDENT INTERFACE
 MDI = MEDIUM DEPENDENT INTERFACE
 OAM = OPERATIONS, ADMINISTRATION & MAINTENANCE
 OLT = OPTICAL LINE TERMINAL
 MCRS= MULTI-CHANNEL RECONCILIATION SUBLAYER
 MPMC= MULTI-POINT MAC CONTROL

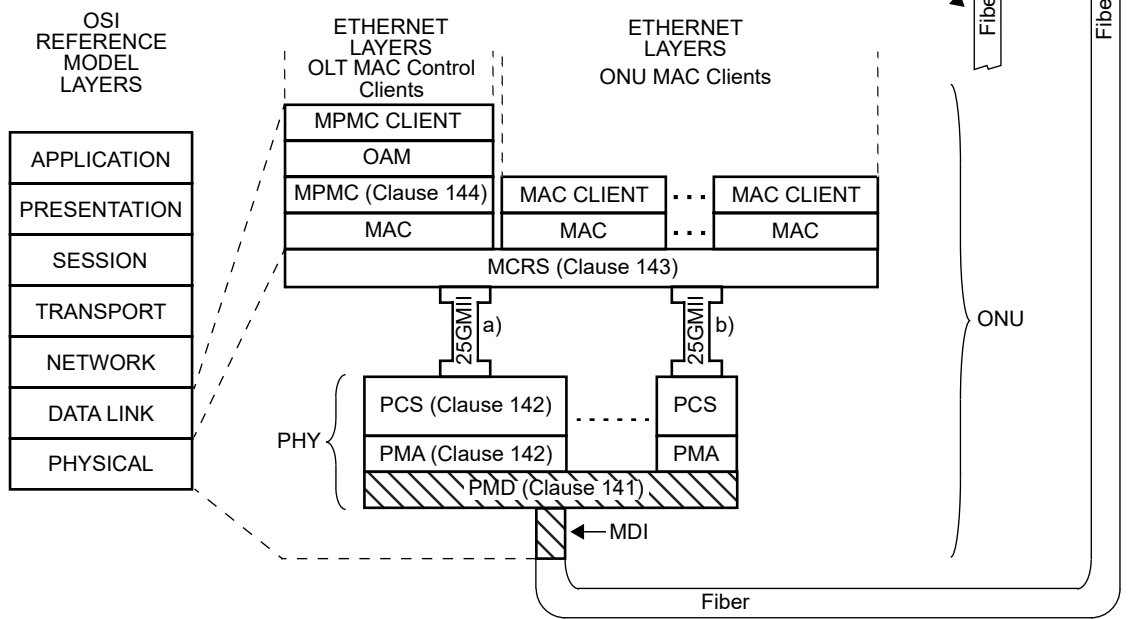
ONU = OPTICAL NETWORK UNIT
 PCS = PHYSICAL CODING SUBLAYER
 PHY = PHYSICAL LAYER DEVICE
 PMA = PHYSICAL MEDIUM ATTACHMENT
 PMD = PHYSICAL MEDIUM DEPENDENT

**Figure 56-5a—Architectural positioning of EFM:
 P2MP Nx25G-EPON architecture**

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54



- a) ~~In some instances of Nx25-EPON one half of an XGMII (transmit or receive) may be paired with its complementary peer (receive or transmit) of a 25GMII to provide a 25 Gb/s downstream and 10 Gb/s upstream interface.~~
- b) This interface may be absent in devices that do not support 50G-EPON PMDs.



PMD and MDI described in this clause

- 25GMII=25 GIGABIT MEDIA INDEPENDENT INTERFACE
- ONU = OPTICAL NETWORK UNIT
- MDI = MEDIUM DEPENDENT INTERFACE
- PCS = PHYSICAL CODING SUBLAYER
- OAM = OPERATIONS, ADMINISTRATION & MAINTENANCE
- PHY = PHYSICAL LAYER DEVICE
- OLT = OPTICAL LINE TERMINAL
- PMA = PHYSICAL MEDIUM ATTACHMENT
- MCRS= MULTI-CHANNEL RECONCILIATION SUBLAYER
- PMD = PHYSICAL MEDIUM DEPENDENT
- MPMC= MULTI-POINT MAC CONTROL

Figure 141-1—Relationship of Nx25G-EPON P2MP PMD to the ISO/IEC OSI reference model and the IEEE 802.3 Ethernet model

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

142. Physical Coding Sublayer and Physical Media Attachment for Nx25G-EPON

142.1 Overview

This clause describes the Physical Coding Sublayer (PCS) with forward error correction (FEC) and Physical Medium Attachment (PMA) used with Nx25G-EPON point-to-multipoint (P2MP) networks. P2MP networks are passive optical networks (PONs) that connect multiple DTEs using a single shared fiber. The architecture is asymmetric, based on a tree and branch topology utilizing passive optical splitters. This type of network requires that the Multipoint MAC Control sublayer exists above the MACs, as described in Clause 144 (see Figure 142–1).

~~In this clause the term xMII is used to refer to both the 25GMII and the XGMII interfaces.~~

Figure 142–2 illustrates the functional block diagram of the Nx25G-EPON PHY with emphasis placed on the PCS. The Nx25G-EPON PCS is specified to support Nx25G-EPON PMDs, where:

- both the receive and transmit paths operate at 25.78125 GBd rate (25/25G-EPON, 50/25G-EPON, and 50/50G-EPON), or
- the receive path operates at 25.78125 GBd rate and the transmit path operates at 10.3125 GBd (25/10G-EPON and 50/10G-EPON ONU), or
- the transmit path operates at 25.78125 GBd rate and the receive path operates at 10.3125 GBd (25/10G-EPON and 50/10G-EPON OLT).

See 143.3.1.1 for definition of TXD, TXC, TX_CLK, RXD, RXD, and RX_CLK.

142.1.1 Conventions

142.1.1.1 State diagrams

The body of this standard comprises state diagrams, including the associated definitions of variables, constants, and functions. The notation used in the state diagrams follows the conventions in 21.5, with extensions listed in the following subclauses. In case of any discrepancies between a state diagram and descriptive text, the state diagram prevails.

142.1.1.2 Hexadecimal Notation

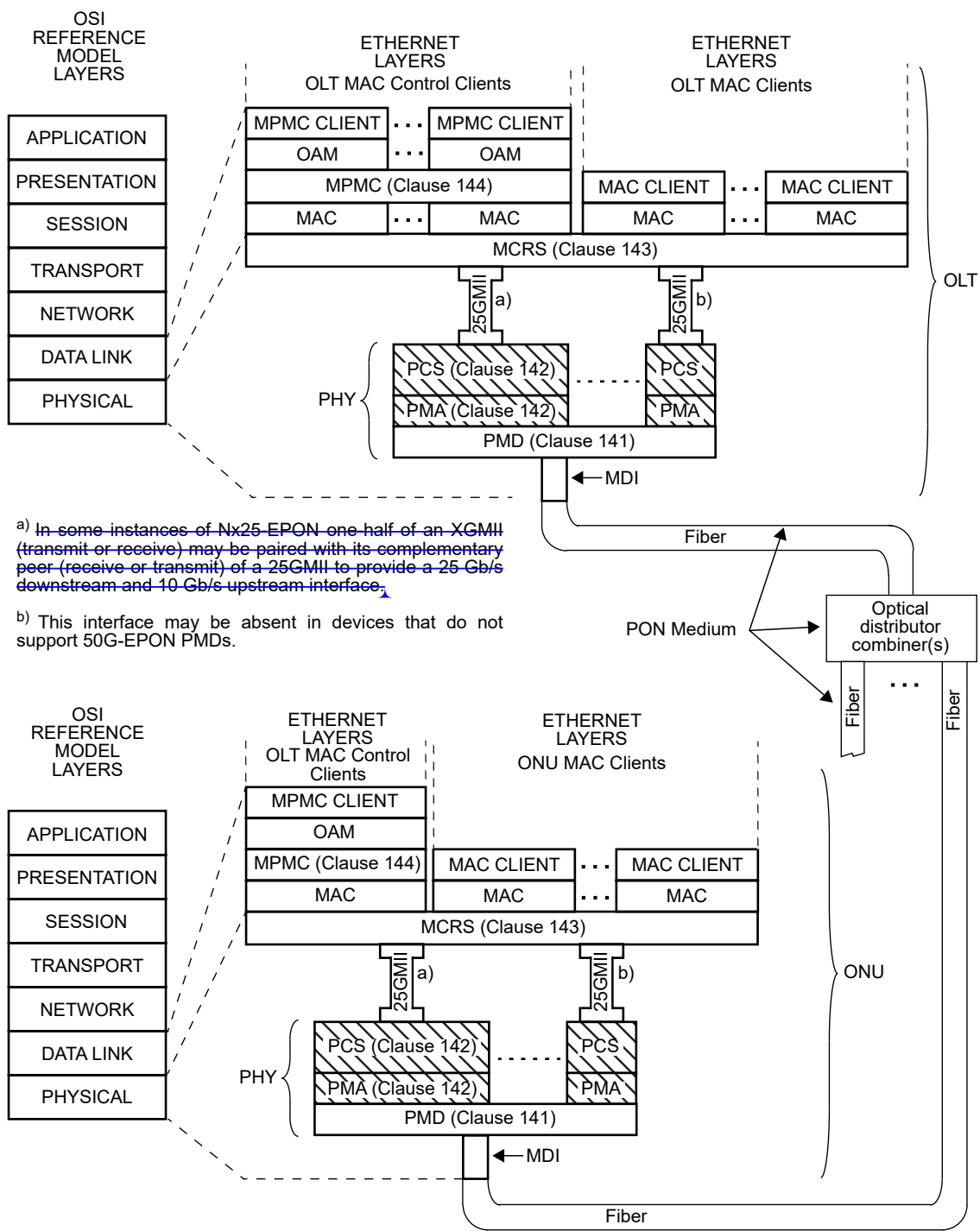
In addition to the rules for hexadecimal notation described in 1.2.5, the following conventions are used in this clause:

- Individual octets of a hexadecimal number are separated by hyphen, e.g., 0x1E-EE-80-23-CA.
- A part of hexadecimal number enclosed in parenthesis followed by a subscripted decimal number n indicates that the parenthetical portion is to be repeated n times. For example, 0x12-34-56-(AB-CD)₆-EF is equivalent to the following expanded representation of a 128-bit number: 0x12-34-56-AB-CD-AB-CD-AB-CD-AB-CD-AB-CD-AB-CD-AB-CD-EF.

142.1.1.3 Timers


Some state diagrams may utilize timers. Timers follow the conventions of 14.2.3.2 augmented as follows:

- a) [start x_timer , y] sets expiration of y to timer x_timer .



a) ~~In some instances of Nx25-EPON one half of an XGMII (transmit or receive) may be paired with its complementary peer (receive or transmit) of a 25GMII to provide a 25 Gb/s downstream and 10 Gb/s upstream interface.~~

b) This interface may be absent in devices that do not support 50G-EPON PMDs.

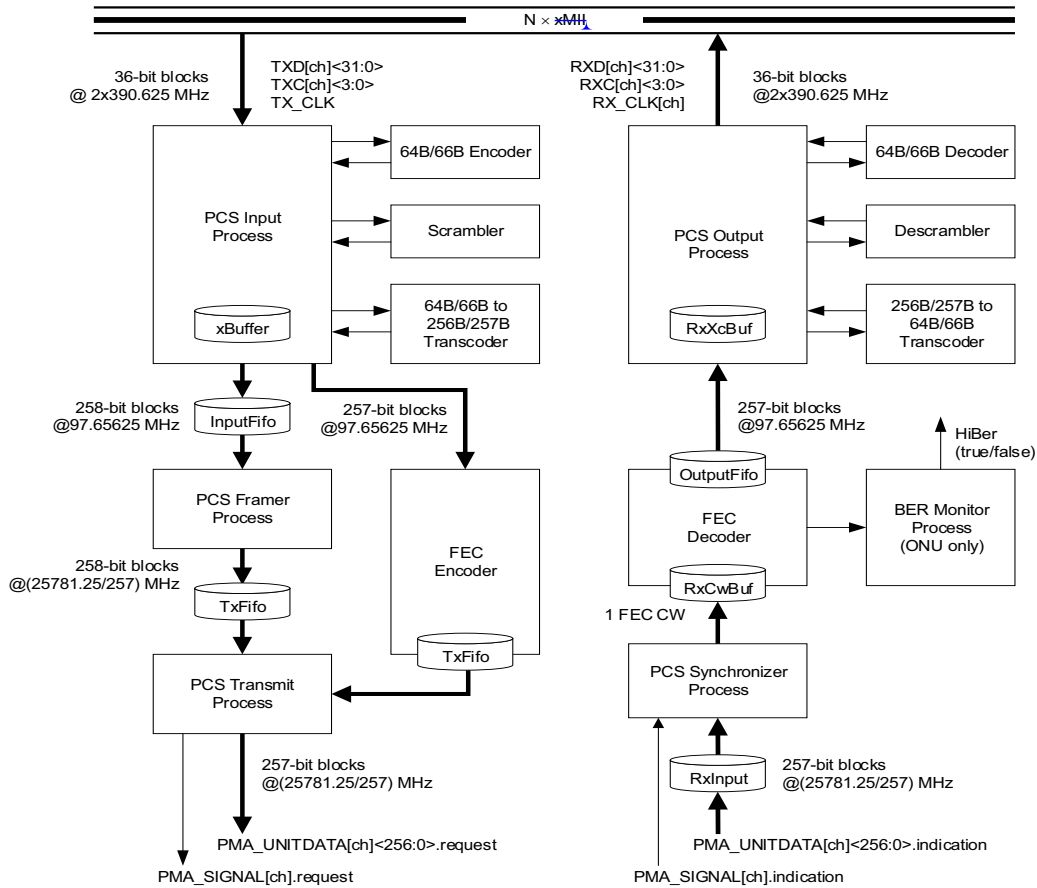
 PCS and PMA described in this clause

25GMII=25 GIGABIT MEDIA INDEPENDENT INTERFACE
 MDI = MEDIUM DEPENDENT INTERFACE
 OAM = OPERATIONS, ADMINISTRATION & MAINTENANCE
 OLT = OPTICAL LINE TERMINAL
 MCRS= MULTI-CHANNEL RECONCILIATION SUBLAYER
 MPMC= MULTI-POINT MAC CONTROL

ONU = OPTICAL NETWORK UNIT
 PCS = PHYSICAL CODING SUBLAYER
 PHY = PHYSICAL LAYER DEVICE
 PMA = PHYSICAL MEDIUM ATTACHMENT
 PMD = PHYSICAL MEDIUM DEPENDENT

Figure 142-1—Relationship of EPON P2MP PMD to the ISO/IEC OSI reference model and the IEEE 802.3 Ethernet model

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54



a) PCS transmit path

b) PCS receive path

NOTE-All clock frequencies in this figure are shown for the nominal MAC data rate of 25 Gb/s. For PCS devices supporting the nominal MAC data rate of 10 Gb/s, all clock frequencies are scaled down by a multiplicative coefficient 0.4.

Figure 142-2—PCS Functional Block Diagram

- b) Upon expiration of timer x_timer , a Boolean variable x_timer_done gets asserted automatically. Restarting the timer x_timer deasserts the value of x_timer_done .
- c) [stop x_timer] aborts the timer operation for x_timer deasserting x_timer_done indefinitely.

142.1.1.4 Operations on variables

The operators used in state diagrams and in associated definitions of variables, constants, and functions are defined in Table 142-1. The operators are listed in decreasing order of precedence.

Table 142-1—State diagram operators

Operator	Meaning
(...)	Indicates precedence or a set of function arguments
[...]	Array subscript

This subclause defines the transmit direction of the Nx25G-EPON PCS. In the OLT, the PCS transmit function operates in a continuous mode at 25.78125 GBd rate.

In the ONU, the PCS transmit function operates in burst mode at 25.78125 GBd rate (25/25G-EPON, 50/25G-EPON, and 50/50G-EPON) or at 10.3125 GBd rate (25/10G-EPON and 50/10G-EPON).

The PCS transmit function includes a mandatory QC-LDPC FEC encoder. The functional block diagram for the PCS transmit function is shown in Figure 142–2. The PCS transmit function consists of the following functional blocks:

- PCS Input (see 142.2.5.4.1),
- PCS Framer Process (see 142.2.5.4.2), and
- PCS Transmit (see 142.2.5.4.3).

As shown in Figure 142–5, the PCS transmitter first inputs two transfers from the ~~xMH~~ and consolidates these into a single 72-bit block which is then encoded into a 66-bit block. Four 66-bit blocks are accumulated, scrambled, and transcoded into a 257-bit block which is transferred to the *InputFifo* and also copied to the FEC encoder. Data is transferred to the *TxFifo*, along with framing information (see 142.2.5.4.2) by the PCS Framer Process. The PCS Transmit Process transfers 257-bit blocks containing framing, information, and parity bits to the PMA. The PCS shall transmit bits in the order shown in Figure 142–5.

142.2.1 64B/66B line encoder

The Nx25G PCS encodes a 72-bit block into a 64B/66B block structure as defined in 49.2.4, using all the block type fields in Figure 49-7 except block type field values of: 0x2D, 0x33, 0x66, 0x55, and 0x4B.

The control characters and their mappings to Nx25G-EPON control codes are specified in Table 142–2. The representations of the control characters are the control codes. Control characters are transferred over the ~~xMH~~ as 7-bit values. The Nx25G-EPON PCS encodes the start and terminate control characters implicitly using the block type field. The Nx25G-EPON PCS does not support ordered set control codes. All control code values that do not appear in Table 142–2 shall not be transmitted and are treated as an error if received.

Table 142–2—Control Codes

Control Character	Notation	xMH control code	Nx25GBASE-PQ control code
Idle	/I/	0x07	0x00
Inter-Envelope Idle	/IEI/	0x08	0x08
Rate Adjust	/RA/	0x09	0x09
Inter-Burst Idle	/IBI/	0x0A	0x0A
Start	/S/	0xFB	Encoded by block type field
Terminate	/T/	0xFD	Encoded by block type field
Error	/E/	0xFE	0x1E

142.2.2 Scrambler

The Nx25G PCS scrambles the payload of each 66-bit block. It then accumulates 66-bits blocks into groups of four and transcodes each group into a single 257-bit block. The payload of each 66-bit block is scrambled using the scrambling function defined in 49.2.6.

In the ONU, at the beginning of each burst, the scrambler is initialized with the value of $0x3\text{-}(FF)_7$, i.e., each of the bits S0 through S7 is set to 1 (see Figure 49–8).

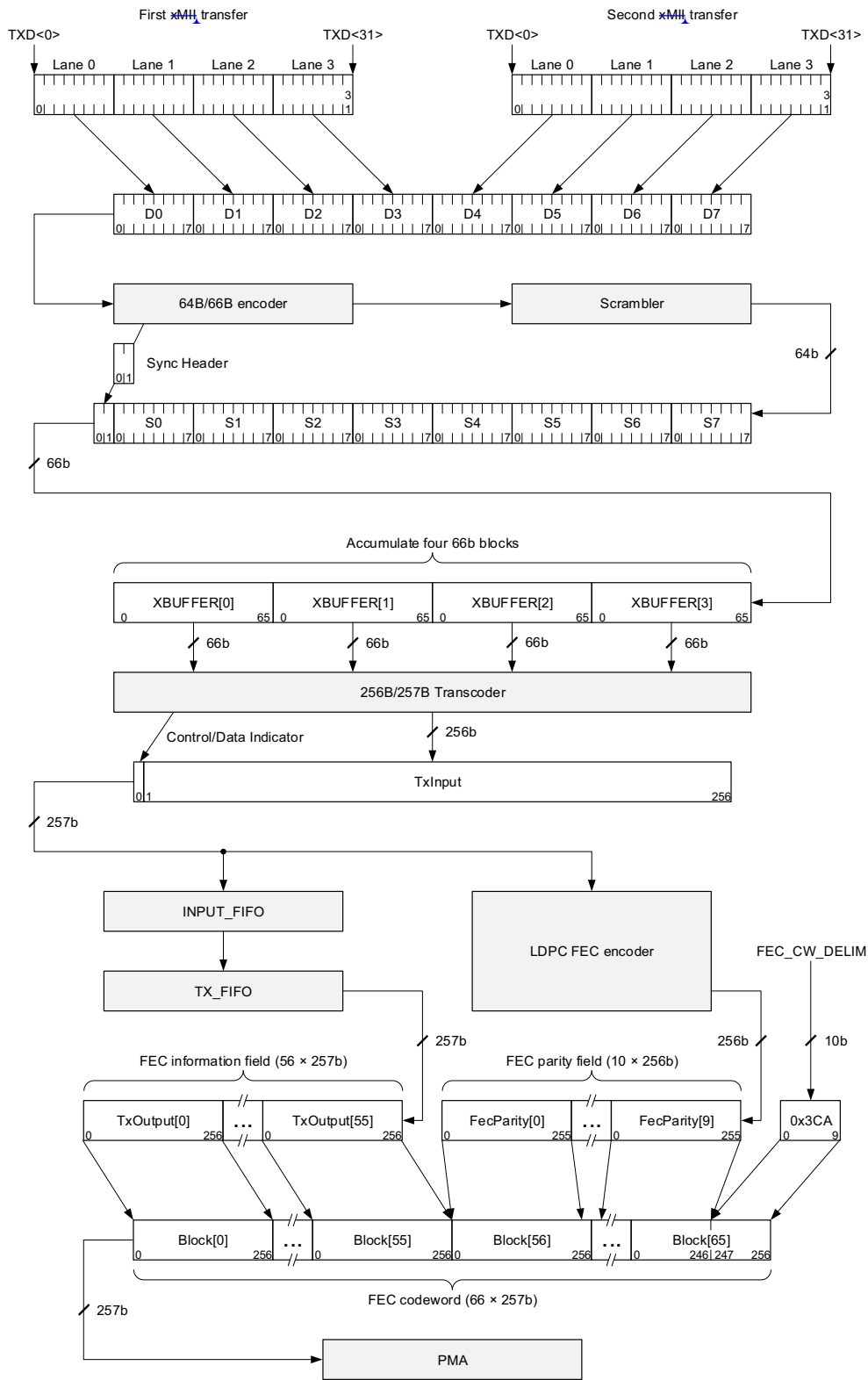


Figure 142-5—Transmit bit ordering

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

IBI258	1
Type: 258-bit block	2
Description: The <i>IBI258</i> constant represents an inter-burst idle block that is generated by the PCS Framing Process in the absence of any burst framing blocks, data blocks, or FEC Parity blocks.	3
Value: 0x0-(0A) ₃₂	4
	5
	6
IBI_EQ	7
See 143.3.3.3	8
	9
PAR_PLACEHLDR	10
Type: 258-bit block	11
Description: The <i>PAR_PLACEHLDR</i> constant represents the value of a 258-bit block inserted into the data stream by the PCS Framing Process in order to reserve the location where FEC Parity and the 10-bit FEC codeword delimiter is to be inserted into the data stream by the PCS Transmit Process.	12
Value: 0x0-(09) ₃₂	13
	14
	15
	16
	17
RATE_ADJ_EQ	18
See 143.3.3.3	19
	20
SCRAMBLED	21
Type: binary	22
Description: This constant indicates that the contents of the 257-bit block are scrambled. When the bit TxInput<257> or TxOutput<257> is set to 1, then bits TxInput<256:0> or TxOutput<256:0> in the same block carry scrambled data.	23
Value: 1	24
	25
	26
	27
142.2.5.2 Variables	28
	29
BEGIN	30
TYPE: Boolean	31
Description: This variable is used when initiating operation of the functional block state diagram. It is set to true following initialization and every reset, and it is reset to false on read.	32
	33
	34
ClkIn	35
Type: Boolean	36
Description: The clear-on-read variable <i>ClkIn</i> is set to true on each rising edge of the xMH clock.	37
	38
ClkOut	39
Type: Boolean	40
Description: The clear-on-read variable <i>ClkOut</i> is set to true once for each 257-bit block output by the PMA, i.e., the <i>ClkOut</i> tracks the transmit clock of the corresponding PMA channel (see 142.4.4).	41
	42
	43
	44
ClkXfr	45
Type: Boolean	46
Description: The clear-on-read variable <i>ClkXfr</i> is set to true once for each 257-bit block output by the PMA, i.e., the <i>ClkXfr</i> tracks the transmit clock of the corresponding PMA channel (see 142.4.4).	47
	48
	49
	50
InputFifo[]	51
Type: array of 258-bit blocks	52
Description: The <i>InputFifo</i> receives data from the PCS Input Process and hands it off to the PCS	53
	54

TxNext		1
Type: 72-bit block		2
Description: The next 72-bit block to be processed by the PCS Input Process.		3
		4
		5
TxLast		6
Type: 258-bit block		7
Description: This variable holds the 258-bit that was read from <i>TxFifo</i> in the previous <i>ClkOut</i> cycle.		8
		9
		10
TxOutput		11
Type: 258-bit block		12
Description: This variable holds one 258-bit block retrieved from the <i>TxFifo</i> .		13
		14
TxPrev		15
Type: 72-bit block		16
Description: This variable holds one 72-bit block received by the PCS Input Process from the xMH in the previous <i>ClkIn</i> cycle.		17
		18
		19
xBuffer[]		20
Type: array of 66-bit blocks		21
Description: This buffer holds four 66-bit blocks of 64B/66B encoded data to be transcoded into one 257-bit block.		22
		23
		24
xIndex		25
Type: Integer		26
Description: An index into the <i>xBuffer</i> indicating the number of encoded blocks contained in the buffer that are ready to be transcoded.		27
		28
		29
142.2.5.3 Functions		30
		31
		32
FecParity()		33
Upon initiation, the first call to this function returns a block containing the first 257 bits from the <i>TxParBuf</i> , i.e. <i>TxParBuf</i> <256:0>. Each subsequent call returns the subsequent 257 bits from the buffer. On the 10th call, the last 257 bits are returned, i.e., <i>TxParBuf</i> <2569:2312>, and the function resets to return <i>TxParBuf</i> <256:0> on the next call. This emulates a circular buffer of size 10 x 257-bits.		34
		35
		36
		37
		38
Encode(v)		39
This function performs 64B/66B encoding of a 72-bit block <i>v</i> per 49.2.13.2.3 and returns the result.		40
		41
FecEncode(v)		42
This function passes a 257-bit block <i>v</i> to the FEC engine for encoding.		43
		44
NextTxVector()		45
This function returns a 72-bit block carrying a single EQ as shown in Figure 142–2. The block is constructed from the data received from the xMH over two subsequent 36-bit transfers: the first transfer is on rising TX_CLK edge and the second transfer is on the falling TX_CLK edge.		46
		47
		48
		49
PassToPMA(v)		50
This function passes a 257-bit block <i>v</i> to the PMA using PMA_UNITDATA[i].request(<i>v</i>).		51
		52
		53
		54

ResetScrambler()
 Description: This function resets the scrambler/descrambler function to the value *IBI_EQ* (see 143.3.3.3).

Scramble(blk)
 This function accepts one 66-bit block *blk* and performs the scrambling operation on the 64-bit payload of the block, as described in 49.2.6. The returned value is a scrambled 66-bit block.

Transcode(a[4])
 This function transcodes four 64B/66B-encoded blocks into a single 256B/257B-encoded block per 91.5.2.5 and returns the result. It takes four 64B/66B-encoded blocks *a[4]* as an argument and returns a 257-bit block.

142.2.5.4 State Diagrams

142.2.5.4.1 PCS Input Process

The PCS Input Process accepts two consecutive 36-bit transfers from the ~~xMH~~ interface and converts them into a single 72-bit block. The Input Process discards all RATE_ADJ_EQs to allow for insertion of FEC parity blocks by the PCS Transmit Process (see 142.2.5.4.3). IBI_EQs not required to complete a 256B/257B block at the end of an upstream burst are also discarded at the Input Process. All other 72-bit blocks are encoded into 64B/66B blocks. Four 64B/66B blocks are accumulated, scrambled, and transcoded into a single 256B/257B block and copied to the FEC Encoder. A single bit indicating the accompanying 256B/257B block has been scrambled and transcoded is appended to the block which is then stored in the *InputFifo*.

The PCS Input Process shall implement the state diagram as depicted in Figure 142–10.

142.2.5.4.2 PCS Framer Process

The PCS Framer Process monitors data from the *InputFifo* and transfers it to the *TxFifo*, inserting inter-burst idle blocks (*IBI258*), *SyncPattern*, parity placeholders (*PAR_PLACEHLDR*), and *EBD258* as appropriate. While the *InputFifo* is empty, the PCS Framer Process appends *IBI258* to the *TxFifo*. When the *InputFifo* first becomes not empty, indicating the beginning of a burst, the *SyncPattern* is appended to the *TxFifo*. Once the complete *SyncPattern* is appended to the *TxFifo*, the Framer Process begins transferring data from the *InputFifo* to the *TxFifo*. When sufficient data for a full FEC payload has been transferred to the *TxFifo*, or the end of the burst is detected as indicated by an empty *InputFifo*, the PCS Framer Process appends sufficient *PAR_PLACEHLDR* blocks to the *TxFifo* to allow insertion of the contents of *TxParBuf* (FEC codeword parity and FEC codeword delimiter) into the data stream by the PCS Transmit Process. Additional FEC codewords are allowed for until the end of the transmission is indicated by an empty *InputFifo*, at which point the Framer Process appends the *EBD258* to the *TxFifo* followed by *IBI258*.

The PCS Framer Process shall implement the state diagram as depicted in Figure 142–11.

142.2.5.4.3 PCS Transmit Process

The PCS Transmit Process transfers data from the *TxFifo* or FEC Encoder to the PMA. On each transition of the *ClkOut* to true the Transmit Process retrieves one 258-bit block of data from the *TxFifo*. If the retrieved 258-bit block indicates the start of the burst and the ONU is currently not transmitting, the laser is turned on and data is sent towards the PMA for transmission. If the retrieved 258-bit block indicates the end of the burst and the ONU is currently transmitting, the laser is turned off and end of the burst delimiter is sent towards the PMA for transmission. If the retrieved 258-bit block indicates the FEC parity placeholder, the calculated FEC parity and 10 bits of FEC codeword delimiter are sent towards the PMA for transmission. Otherwise, data from the *TxFifo* is sent towards the PMA for transmission.

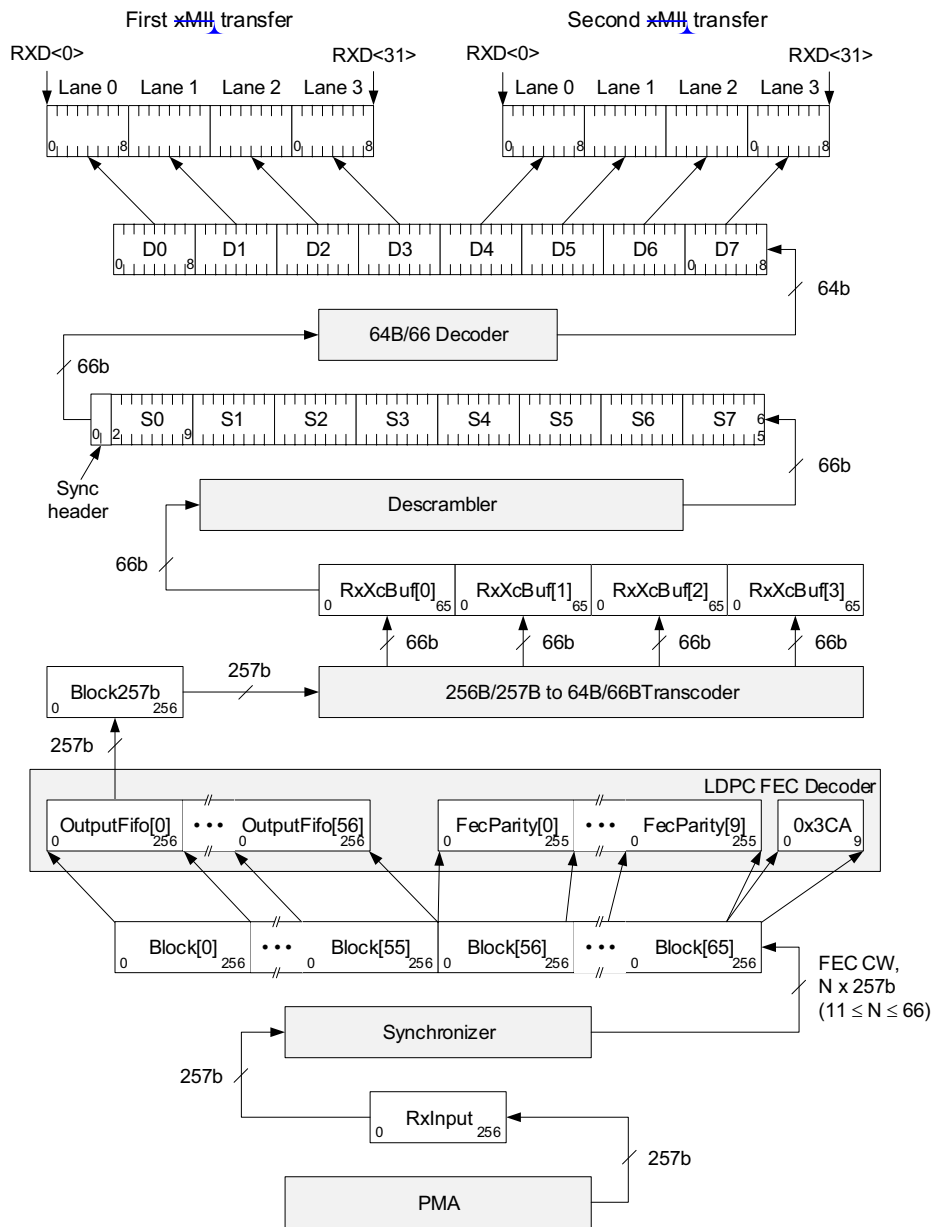


Figure 142–14—PCS receive bit ordering

142.3.5 Receive data path state diagrams

142.3.5.1 Constants

EBD257

Type: 257-bit block

Description: The *EBD257* constant holds the value of the end-of-burst delimiter.

Value: 0x00

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

BEGIN	1
See 142.2.5.2	2
	3
BerMonitorInterval	4
Indicates the length of the interval window period associated with the QC-LDPC BER monitor in units of QC-LDPC codewords (see 45.2.3.43). This value is reflected in MDIO register 3.80.	5
	6
	7
BerThreshold	8
Indicates the threshold value of invalid QC-LDPC codeword errors within the QC-LDPC BER monitor function. At the end each monitor interval period, <i>HiBer</i> is updated. The value of <i>BerThreshold</i> is reflected in MDIO register 3.82	9
	10
	11
	12
Block257b	13
Type: 257-bit block	14
Description: The <i>Block257b</i> variable temporarily holds one 257-bit block removed from the head of <i>OutputFifo</i> .	15
	16
	17
Block66b	18
Type: 66-bit block	19
Description: The <i>Block66b</i> variable temporarily holds one descrambled 66-bit block.	20
	21
Block72b	22
Type: 72-bit block	23
Description: The <i>Block72b</i> variable temporarily holds the value being passed to the xMII	24
	25
CwAvailable	26
Boolean variable that is set true when a new QC-LDPC codeword is available for testing by the BER Monitor Process and set to false when WAIT_FOR_CODEWORD state is entered. A new QC-LDPC codeword is available for testing by the BER Monitor Process when the ONU Synchronizer Process has accumulated enough blocks from the PMA to evaluate the next QC-LDPC codeword (see Figure 142–16).	27
	28
	29
	30
	31
	32
CwLeft	33
Counts the remaining number of QC-LDPC codewords within the current BER monitoring interval.	34
	35
	36
CwValid	37
Boolean indication that is set true if a received QC-LDPC codeword is valid. As an example, an QC-LDPC codeword is valid if and only if all parity checks of the QC-LDPC code are satisfied thereby terminating iterations without exceeding the maximum count (e.g., 15). The specific method for evaluating codeword validity is implementation dependent within the QC-LDPC decoder and outside the scope of this standard.	38
	39
	40
	41
	42
	43
HiBer	44
Boolean variable that is asserted true if <i>BadCwCount</i> reaches or exceeds <i>BerThreshold</i> QC-LDPC codeword errors within one BER monitor interval period, otherwise set to false. The value of <i>HiBer</i> is reflected in MDIO register 3.81.	45
	46
	47
	48
MatchCount	49
Type: Integer	50
Description: This counter tracks the number of consecutive successful detections of FEC codeword delimiters (<i>FEC_CW_DELIM</i>) while the ONU is not synchronized to the proper 257-bit block boundary.	51
	52
	53
	54

OutputFifo[]	1
Type: Array of 257-bit blocks	2
Description: The <i>OutputFifo</i> buffer holds one FEC codeword payload after it has been processed by the FEC Decoder. The <i>OutputFifo</i> supports FIFO access operations as defined in 142.1.1.6.	3
	4
	5
PayloadLeft	6
Type: Integer	7
Description: This variable holds the number of EQs remaining until one maximum length FEC codeword payload has been sent to the xMH .	8
	9
	10
PersistentFecFail	11
Type: Boolean	12
Description: This variable is set to true if the FEC decoder is unable to correct all errors in the three FEC codewords most recently received on a given channel. Otherwise, this variable is set to false. In the OLT, the <i>PersistentFecFail</i> value is reset when <i>SignalFail</i> becomes true, or the EBD is detected, i.e., the uncorrectable FEC codewords from the previous burst do not result in <i>PersistentFecFail</i> becoming true during the next burst.	13
	14
	15
	16
	17
	18
RateAdjLeft	19
Type: Integer	20
Description: This variable holds the number of EQs remaining to be generated in the PCS Output Process to fill the gap left by the removal of FEC codeword parity data from the current FEC codeword.	21
	22
	23
	24
RxCwBuf[]	25
Type: An array of 257-bit blocks	26
Description: The <i>RxCwBuf</i> is a buffer capable of storing one full FEC codeword. The <i>RxCwBuf</i> supports FIFO access operations as defined in 142.1.1.6.	27
	28
	29
RxInput	30
Type: 257-bit block	31
Description: The <i>RxInput</i> is a buffer containing the 257 bits most recently received from the PMA sublayer on a given channel.	32
	33
	34
RxXcBuf[3:0]	35
Type: Array of four 66-bit blocks	36
Description: This buffer holds four 66-bit blocks resulting from the decoding of a 257-bit block.	37
	38
SBD257	39
Type: 257-bit block	40
Description: The <i>SBD257</i> variable represents the start-of-burst delimiter, and its value is equal to either SP2 or SP3, depending on the most recently provisioned synchronization pattern (see 142.1.3.1).	41
Value: see 142.1.3.1	42
	43
	44
	45
SignalFail	46
Type: Boolean	47
Description: This Boolean variable is set based on the most recently received value of <i>PMA_SIGNAL.indication(SIGNAL_OK)</i> received on a given channel. It is true if the value of <i>SIGNAL_OK</i> was FAIL and false if the value was OK.	48
	49
	50
	51
XcIndex	52
Type: Integer	53
	54

Description: The *XcIndex* variable is an index to the *RxCwBuf* array and has a value ranging between 0 and 3, inclusively.

142.3.5.3 Functions

Decode257b(blk)

Description: This function accepts one 256B/257B encoded block *blk* and transcodes it in-to four 64B/66B encoded blocks. The result is returned as an array of four 66-bit blocks.

Decode66b(blk)

Description: This function accepts one 64B/66B encoded block *blk* and performs the decoding operation as described in 49.2.11 and Figure 49-17. The returned value is a 72-bit block.

Descramble(blk)

Description: This function accepts one 66-bit block *blk* and performs the descrambling operation on the 64-bit payload of the block, as described in 49.2.10. The returned value is a descrambled 66-bit block.

PassToFecDecoder(cw)

Description: The *PassToFecDecoder* function passes one complete FEC codeword *cw* to the FEC Decoder. The FEC codeword may be full-length or shortened. The codeword length is intrinsic to the parameter *cw*.

MatchFound(value1, value2, threshold)

Description: This function compares bit by bit its arguments *value1* and *value2* and returns a Boolean true if the number of bits that are different is less or equal to the *threshold*, otherwise the function returns false.

OutputBlock(eq)

Description: This function accepts one 72-bit block *eq* and outputs two 36-bit blocks over the ~~xMH~~. This is a blocking function and the control is not returned to the calling state until after the second 36-bit block is sent.

ResetScrambler()

See 142.2.5.3

Shift(buffer, n)

Description: This function receives 257-bit blocks from the PMA via the *PMA_UNITDATA.indication(rx_code_group<256:0>)* primitive and inserts *n* new bits at the end of the FIFO *buffer*, while removing the same number of old bits at the head of the *buffer*. The *Shift()* function is blocking and its execution takes exactly *n* bit times at the given receiving line rate.

142.3.5.4 OLT Synchronizer Process state diagram

The OLT Synchronizer Process is responsible for receiving unaligned 257-bit blocks from the PMA sublayer and aligning these blocks to the correct 257-bit block boundary. This process hunts for *SBD257* and *EBD257* values, allowing for a certain Hamming distance (see *SBD_TH* and *EBD_TH*). The 257-bit blocks that are received between the *SBD* and the *EBD* are accumulated in the *RxCwBuf* buffer. When a complete full-length or shortened FEC codeword is stored in the *RxCwBuf*, the buffer content is passed to the FEC Decoder function (see 142.3.1).

The OLT shall implement an instance of Synchronizer Process as depicted in Figure 142–15 for every enabled receive channel.

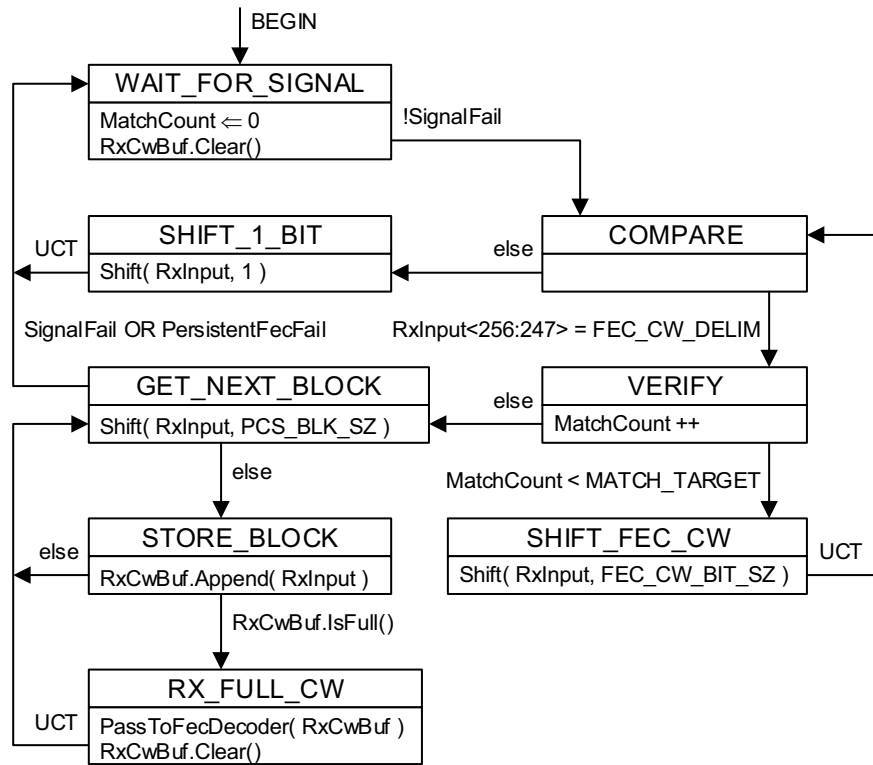


Figure 142–16—ONU Synchronizer Process state diagram

142.3.5.7 PCS Output Process

The PCS Output Process receives corrected information bits from the FEC Decoder. The FEC Decoder outputs an entire payload of a FEC codeword into the *OutputFifo* buffer. The FEC codeword payload consists of fifty-six 257-bit blocks, however, in the OLT, the payload of a last codeword in a burst may contain fewer than 56 blocks.

The PCS Output Process converts the 257-bit blocks into EQs by first transcoding each 257-bit block into four 66-bit blocks, then descrambling each block, and finally, decoding each 66-bit block into a 72-bit block. The 72-bit blocks are passed to ~~xMH~~ for transfer to the MCRS.

The PCS shall implement an instance of the Output Process as depicted in Figure 142–18 for every enabled receive channel.

142.4 Nx25G-EPON PMA

The PMA includes a downstream differential encoding option at the serial bit rate (output bits represent changes to succeeding input values rather than in respect to a given reference). This encoding technique facilitates the use of lower bandwidth receivers.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

143. Multi-Channel Reconciliation Sublayer

143.1 Overview

This clause describes the Multi-Channel Reconciliation Sublayer (MCRS) which enables multiple MACs to interface with multiple xMIIs. Figure 143–1 shows the relationship between this MCRS and the ISO/IEC OSI reference model. Generally, single-channel RS specifications enabled a single MAC to interface to a single PHY in point-to-point (P2P) links, or multiple MACs to interface to a single PHY in P2MP links (e.g., EPON architectures). This concept is expanded in this clause to allow single or multiple MACs to interface with multiple PHYs in either P2P or P2MP applications.

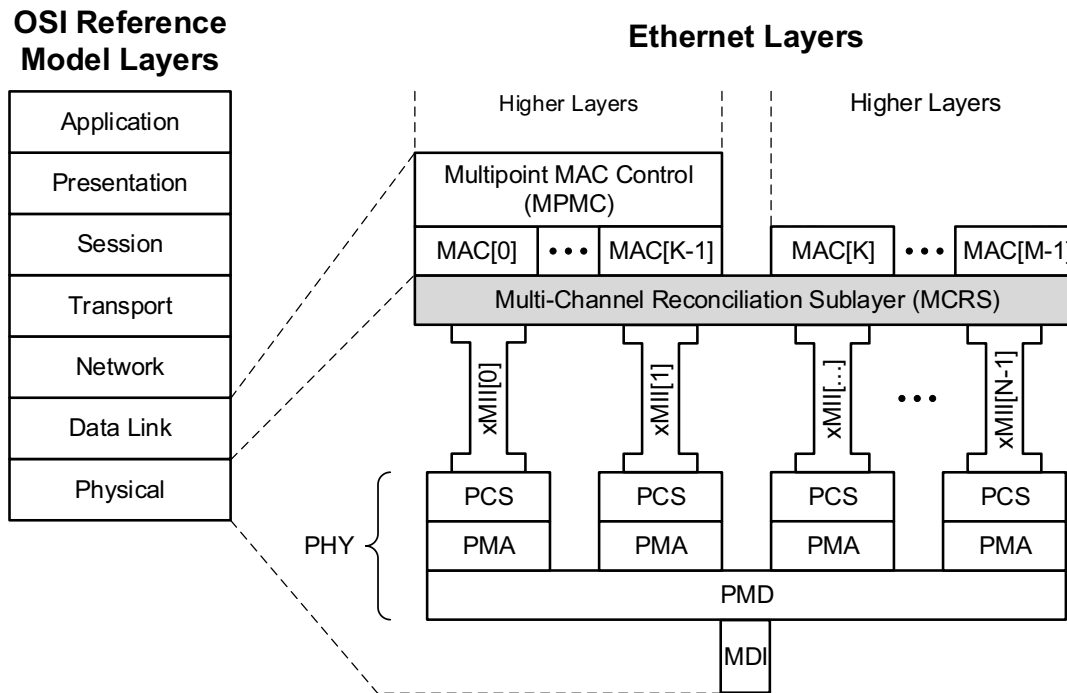


Figure 143–1—Relationship of MCRS to the OSI Reference Model

The MCRS adapts the bit-serial protocols of the MAC to the parallel format of the Physical Coding Sublayer (PCS) service interface. This clause defines an MCRS as an interface between the MAC sublayer and one or more xMIIs. In this clause, xMII is used as a generic term for the Media Independent Interfaces for implementations of 10 Gb/s and above. ~~For example, for 10 Gb/s implementations, it is called XGMII; for 25 Gb/s implementations, it is called 25GMII.~~ Though the xMII is an optional logical interface between the MAC sublayers and the Physical Layers, it is used extensively in this clause as a basis for specification.

143.2 Summary of major concepts

The following are the major concepts of the MCRS:

- a) The MCRS transmission is controlled by a higher layer (e.g., Multipoint MAC Control sublayer defined in Clause 144) via the use of MCRS_CTRL primitives, which indicate envelope start time, durations, and transmission channels.

- b) The MCRS establishes a temporary binding of a single MAC instance to one or more xMII instances with all xMIIs operating at the same rate.
- c) In the transmit direction, the MCRS converts the MAC serial data stream into the parallel data paths of multiple xMIIs servicing separate PHYs.
- d) In the receive direction, the MCRS maps the signal sets provided by the xMIIs to the PLS service primitives of individual MACs.
- e) Each direction of data transfer is independent and serviced by data, control, and clock signals.
- f) The MCRS generates continuous data or control characters in the transmit path and expects continuous data or control characters in the receive path.

143.2.1 Concept of a logical link and LLID

In point-to-multipoint architectures, such as EPON, the transmitting and receiving stations may include multiple MAC instances. Such architectures are best viewed as a collection of logical point-to-point and/or point-to-multipoint links. A point-to-point logical link connects a single MAC instance at the transmitting station to a single MAC instance at the receiving station. A point-to-multipoint logical link takes advantage of the P2MP topology and connects a single MAC instance at the transmitting station to multiple MAC instances at multiple receiving stations. The transmitting and receiving stations may be logically connected with each other via multiple logical links.

A logical link is created in the MCRS (below the MAC) by tagging each frame (or frame fragment) with a logical link identifier (LLID) value. The MCRS Transmit function inserts a specific LLID value depending on which instance of MAC has sourced the frame. The MCRS Receive function directs the received frame (or frame fragment) to the specific MAC instance mapped to this LLID value, or to multiple MAC instances, in case of point-to-multipoint logical link. The concept of a logical link is further defined in 144.3.4.

143.2.2 Concept of an MCRS channel

An MCRS channel is a single unidirectional transmission path through the MCRS. The number of channels contained within an MCRS generally corresponds to the number of xMII instances connected to the MCRS. Thus, an MCRS implementation that connects to N xMII instances contains N transmit MCRS channels and N receive MCRS channels. Some architectures (e.g., EPON) allow an xMII interface to only implement either receive or a transmit data path. In such architectures, the number of receive and transmit MCRS channels may be different. For example, in 50/10G-EPON OLT, there are two transmit MCRS channels attached to two ~~25GMII~~ and one receive MCRS channel attached to one ~~XGMII~~.

143.2.3 Binding of multiple MACs to multiple xMII instances

The key function of the MCRS is the dynamic binding of a PLS_DATA[m] interface to one or more MCRS channels (m represents the index of the MAC instance). The dynamic nature of the binding means that such a binding exists only for a predetermined interval of time during which a given MAC instance is expected to transmit or receive data. After that time, the binding no longer exists, and a different MAC instance may bind to the same MCRS channels.

The dynamic binding of MAC instances to MCRS transmit channels is controlled by the MCRS_CTRL.request() primitive described in 143.3.1.2.1. The dynamic binding of MAC instances to receive MCRS channels is determined by the LLID value of the incoming data.

Table 143–1—Mapping of PLS_DATA.request primitives

MAC operating speed	MCRS channels	Transmit interface	Signals
10 Gb/s	1	XGMII[0]	TXD[0]<31:0>, TXC[0]<3:0> and TX_CLK
25 Gb/s	1	25GMII[0]	TXD[0]<31:0>, TXC[0]<3:0> and TX_CLK
50 Gb/s	2	25GMII[0] 25GMII[1]	TXD[0]<31:0>, TXC[0]<3:0> and TX_CLK ^a TXD[1]<31:0>, TXC[1]<3:0>
Nx25 Gb/s	N	25GMII[0] 25GMII[1] 25GMII[2] ... 25GMII[N-1]	TXD[0]<31:0>, TXC[0]<3:0> and TX_CLK ^a TXD[1]<31:0>, TXC[1]<3:0> TXD[2]<31:0>, TXC[2]<3:0> ... TXD[N-1]<31:0>, TXC[N-1]<3:0>

^a All transmit 25GMII interfaces share a common clock.

Table 143–2—Mapping of PLS_DATA.indication primitives

MAC operating speed	MCRS channels	Receive interface	Signals
10 Gb/s	1	XGMII[0]	RXD[0]<31:0>, RXC[0]<3:0> and RX_CLK[0]
25 Gb/s	1	25GMII[0]	RXD[0]<31:0>, RXC[0]<3:0> and RX_CLK[0]
50 Gb/s	2	25GMII[0] 25GMII[1]	RXD[0]<31:0>, RXC[0]<3:0> and RX_CLK[0] RXD[1]<31:0>, RXC[1]<3:0> and RX_CLK[1]
Nx25 Gb/s	N	25GMII[0] 25GMII[1] 25GMII[2] ... 25GMII[N-1]	RXD[0]<31:0>, RXC[0]<3:0> and RX_CLK[0] RXD[1]<31:0>, RXC[1]<3:0> and RX_CLK[1] RXD[2]<31:0>, RXC[2]<3:0> and RX_CLK[2] ... RXD[N-1]<31:0>, RXC[N-1]<3:0> and RX_CLK[N-1]

143.3.1.1.1 Mapping of PLS_DATA[ch].request primitive

The MCRS maps the primitive PLS_DATA.request to the xMII signals TXD[ch]<31:0>, TXC[ch]<3:0>, and TX_CLK in the same way as for the XGMII as specified in 46.1.7.1.

143.3.1.1.2 Mapping of PLS_SIGNAL[ch].indication primitive

The MCRS support full duplex operation only and does not generate the PLS_SIGNAL.indication primitive.

143.3.1.1.3 Mapping of PLS_DATA[ch].indication primitive

The MCRS maps the primitive PLS_DATA.indication to the xMII signals RXD[x]<31:0>, RXC[x]<3:0> and RX_CLK[x] in the same way as for the XGMII as specified in 46.1.7.2.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

143.3.1.1.4 Mapping of PLS_DATA_VALID[ch].indication primitive

The MCRS maps the primitive PLS_DATA_VALID.indication to the xMII signals RXC[x]<3:0> and RXD[x]<31:0> in the same way as for the XGMII as specified in 46.1.7.5.

143.3.1.1.5 Mapping of PLS_CARRIER[ch].indication primitive

The MCRS supports full duplex operation only and does not generate the PLS_CARRIER.indication primitive.

143.3.1.2 MCRS Control Primitives

The MCRS inputs the MCRS_CTRL[ch].request primitives from the Multi-Point Control Protocol (MPCP) and outputs to the MPCP the MCRS_CTRL[ch].indication primitives.

143.3.1.2.1 MCRS_CTRL[ch].request(link_id, epam, env_length) primitive

The MPCP requests the MCRS to transmit the next envelope using the MCRS_CTRL[ch].request(link_id, epam, env_length) primitive. This opens an envelope on channel *ch* for the LLID specified by *link_id* with a length (in EQs) of *env_length*. If all channels are idle, the *EnvPam* variable (see 143.3.3.4) is set to the value of *epam* (see *EnvStartHeader()* function definition in 143.3.3.5).

143.3.1.2.2 MCRS_CTRL[ch].indication() primitive

The Input Process (see Figure 143–12) requests the next envelope from the MPCP after the completion of the previous envelope using the *MCRS_CTRL[ch].indication()* primitive. This primitive indicates to the MPCP that the MCRS is available for the next envelope in a given channel. In the absence of an active envelope, the *MCRS_CTRL[ch].indication()* primitive is generated continuously on every *InClk* transition (see 143.3.3.4). The MPCP may decide whether to issue a new envelope immediately adjacent to the previous envelope for envelopes that are expected to be packed in the same transmission burst. If the MPCP has determined that a transmission opportunity has ended it signals that condition by issuing an envelope with *link_id* set to 0x00-00.

143.3.1.2.3 MCRS_ECH[ch].indication(Llid) primitive

The Output Process (see Figure 143–16) generates the *MCRS_ECH[ch].indication(Llid)* primitive every time an ESH EQ is read from the *ENV_RX* buffer. This primitive causes the MPMC Control Parser Process (see 144.2.1) to generate a local timestamp (i.e., to latch the local MPCP time) representing the arrival time of ESH EQ.

143.3.1.3 XGMII interfaces

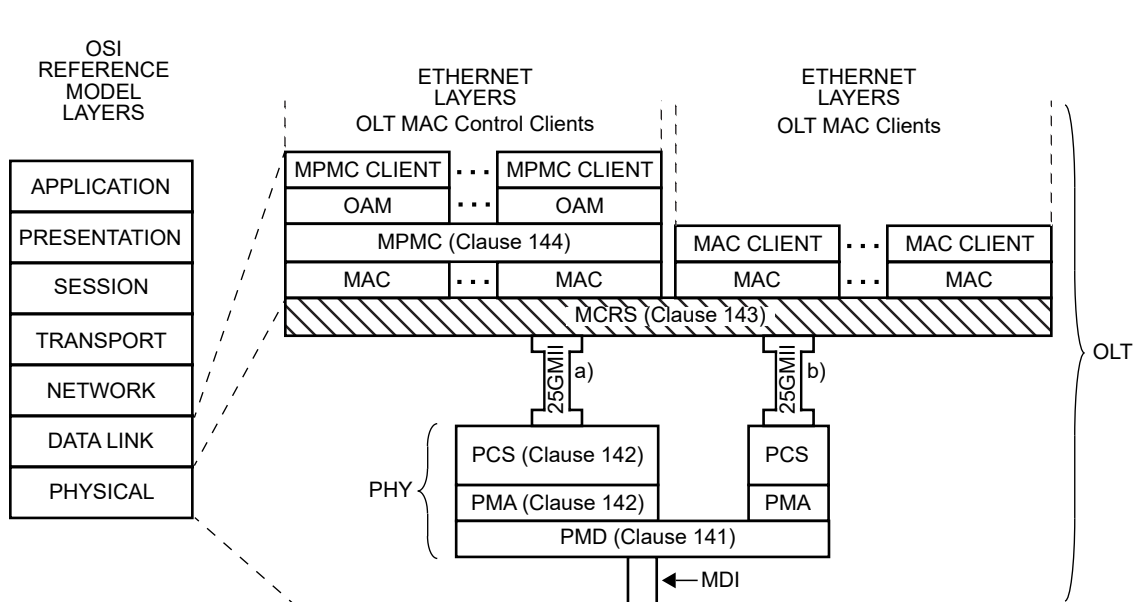
The XGMII is specified to support 10 Gb/s operation. The structure of each of the XGMII interfaces in an MCRS system is as specified in 46.1.6.

For mapping between the XGMII signals and the PLS Service interface, see 143.3.1.1.1 and 143.3.1.1.3.

For multi-channel MCRS systems the transmit XGMIIs are synchronous and only one TX_CLK is required.

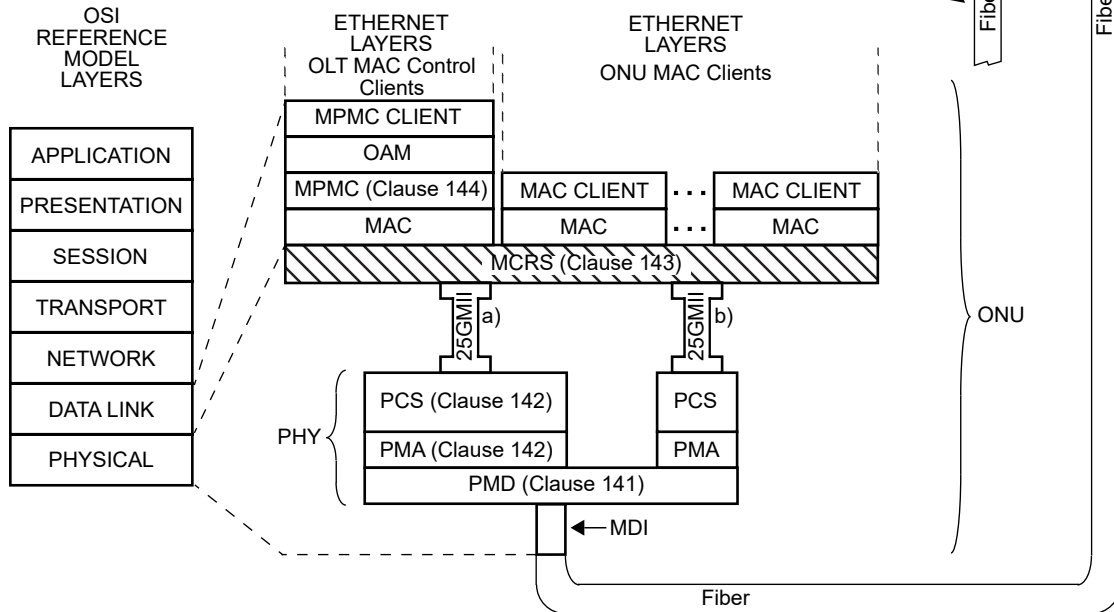
143.3.1.4 25GMII interfaces

The 25GMII is specified to support ~~25 Gb/s operation~~. The structure of each of the 25GMII interfaces in an MCRS system is identical to the XGMII structure specified in 46.1.6. The 25GMII data stream has the same



a) In some instances of Nx25-EPON one half of an XGMII (transmit or receive) may be paired with its complementary peer (receive or transmit) of a 25GMII to provide a 25-Gb/s downstream and 10-Gb/s upstream interface.

b) This interface may be absent in devices that do not support 50G-EPON PMDs.



MCRS described in this clause

25GMII= 25 GIGABIT MEDIA INDEPENDENT INTERFACE
 MDI = MEDIUM DEPENDENT INTERFACE
 OAM = OPERATIONS, ADMINISTRATION & MAINTENANCE
 OLT = OPTICAL LINE TERMINAL
 MCRS= MULTI-CHANNEL RECONCILIATION SUBLAYER
 MPMC= MULTI-POINT MAC CONTROL

ONU = OPTICAL NETWORK UNIT
 PCS = PHYSICAL CODING SUBLAYER
 PHY = PHYSICAL LAYER DEVICE
 PMA = PHYSICAL MEDIUM ATTACHMENT
 PMD = PHYSICAL MEDIUM DEPENDENT

Figure 143-17—Relationship of Nx25G-EPON P2MP PMD to the ISO/IEC OSI reference model

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

The MCRS is used with Nx25G-EPON point-to-multipoint (P2MP) networks in order to interface multiple MAC instances with one or two 25GMII channels in each direction. Figure 143–17 illustrates the relationship of the MCRS and the OSI protocol stack for Nx25G-EPON.

Nx25G-EPON OLT and ONU PMDs are defined in Clause 141, with the respective Nx25G-EPON PCS defined in 142.2 and 142.3.

The MCRS in Nx25G-EPON architecture serves as an interfaces sublayer between the MAC sublayer and 25GMII. The 25GMII interfaces have the following characteristics:

- a) The 25GMII is rate-scalable and may support rates of 25 Gb/s and 10 Gb/s.
- b) They provide independent 32-bit-wide transmit and receive data paths.
- c) They support full duplex operation only.

143.4.1.1 MCRS channels

An MCRS channel that carries information from the OLT to the ONU is referred to as the downstream channel, and the channel that carries information from an ONU to the OLT is referred to as the upstream channel.

The 25/10G-EPON and 25G/25G-EPON architectures shall implement a single MCRS channel in each direction.

The 50/10G-EPON and 50/25G-EPON architectures shall implement two MCRS channels in the downstream direction and a single channel in the upstream direction.

The 50/50G-EPON architecture shall implement two channels in each direction.

When two channels are implemented in the same direction, channel bonding of these two channels shall be supported. Table 143–6 summarizes MCRS channels for Nx25G-EPON.

Each MCRS channel is bound to a separate PCS instance via a separate xMII instance. Channels operating at 25 Gb/s are bound to 25GMII, whereas the channel operating at 10 Gb/s is bound to XGMII instance. Thus, for any given system, there is a one-to-one correspondence between the MCRS channel count and the number of xMII instances supported.

Table 143–6—MCRS channel designation and capabilities

Designation	MCRS Channel	MCRS Channel Function
DC0	Downstream channel 0	All ONUs receive this MCRS channel, broadcast
DC1	Downstream channel 1	Only ONUs capable of receiving at 50 Gb/s support this MCRS channel.
UC0	Upstream channel 0	All ONUs transmit on this MCRS channel
UC1	Upstream channel 1	Only ONUs capable of transmitting at 50 Gb/s support this MCRS channel.

143.4.1.2 Symmetric and Asymmetric Data Rates

The Nx25G-EPON architecture supports symmetric and asymmetric data rates. The symmetric data rate systems include 25/25G-EPON or 50/50G-EPON. The asymmetric rate systems include 25/10G-EPON, 50/10G-EPON, and 50/25G-EPON.

A distinction is made regarding the underlying mechanisms of achieving the asymmetric data rates. In 25/10G-EPON systems, the asymmetric data rate is achieved via the MCRS channel rate asymmetry, where

a single downstream MCRS channel DC0 operates at 25 Gb/s and a single upstream MCRS channel UC0 operates at 10 Gb/s. Additional details for MCRS implementations supporting the channel rate asymmetry are provided in 143.4.4. In 50/25G-EPON systems, the asymmetric data rate is achieved via the MCRS channel number asymmetry, where two MCRS channels are active in the downstream direction (DC0 and DC1), but only a single MCRS channel UC0 is active in the upstream direction. In 50/25G-EPON systems, upstream and downstream MCRS channels operate at the data rate of 25 Gb/s.

Editor's Note (to be removed prior to publication): content of 143.4.4 is currently missing and will be contributed as a comment against the draft - specifically, against this note.

Both the channel rate asymmetry and the channel number asymmetry mechanisms may be combined, as is the case in 50/10G-EPON systems, where there are two downstream MCRS channels operating at 25 Gb/s and a single upstream MCRS channel operating at 10 Gb/s.

An Nx25G-EPON system may serve ONUs that support different numbers of MCRS channels (see 143.4.1). Therefore, some ONUs are only able to receive and transmit data on MCRS channels DC0 and UC0, some are able to receive on DC0 and DC1 and transmit on UC0 and UC1 or just on UC0.

143.4.1.3 Nx25G-EPON application-specific parameters

For definitions of constants, variables, and functions, see 143.3.3 (transmit direction) and 143.3.4 (receive direction).

143.4.1.3.1 Constants

ADJ_BLOCK_SIZE
Value: 257

NUM_CH
Value: 1 for devices supporting only 10 Gb/s or 25 Gb/s operation over a single channel; 2 for devices supporting 50 Gb/s operation over two channels.

RATE_ADJ_SIZE
Value: 33

143.4.1.3.2 Transmit variables

EnvTx
Description: Since there is no timing jitter or channel skew to be removed at the transmitting device, the size of *EnvTx* buffer may be reduced to only two rows. If this optimization is implemented, the variables *rRow* and *wRow* are represented by 1-bit unsigned integers.

143.4.2 MCRS time synchronization

For MCRS to provide the intended skew and jitter remediation capabilities, a sufficient delay margin has to be built into the MCRS buffering at the ONU and the OLT. Such delay margin is established at the ONU registration time by proper setting of MCRS *EnvRx* read and write pointers at the OLT and the ONU.

Upon power-up or reset, an unregistered ONU synchronizes to the received clock and aligns to 257-bit block and FEC codeword boundaries on each of its active (enabled) receive channels (see ONU Synchronizer Process, 142.3.5.5). After that, the received data is passed to FEC decoder, which introduces a near-constant delay. Corrected data from the FEC decoder is passed to ~~xMH~~ and is received into ONU MCRS *EnvRx* buffer.

The following are the ONU rules for setting the *EnvRx* write and read pointers:

- 1) Write pointer
 - a) The ONU MCRS always sets the write pointer for the *EnvRx* buffer to equal the EPAM value in any envelope header it receives, regardless of the LLID value in that envelope header.
 - b) If multiple receive channels are active, the write pointers are set independently for each channel based on EPAM values in envelope headers received on each channel.
- 2) Read Pointer
 - a) The read pointer increments synchronously with the *LocalTime* counter, which is locked to the ~~xMH~~ receive clock of an active (enabled) receive channel with the lowest index.
 - b) In an unregistered ONU, upon every update of a write pointer associated with the receive channel with the lowest index, the read pointer is also updated according to the following equation:

$$ReadPointer = WritePointer \text{ XOR } 0x20$$

In the OLT, the PCS receiver synchronizes on start-of-burst delimiter (see OLT Synchronizer Process, 142.3.5.5) independently on each active (enabled) receive channel. After that, the received data is passed to FEC decoder, which introduces a near-constant delay. Corrected data from the FEC decoder is passed to the ~~xMH~~ and is received into the OLT MCRS *EnvRx* buffer. The following are the OLT rules for setting the *EnvRx* write and read pointers:

- 1) Write pointer
 - a) When receiving an envelope from a registered ONU, the OLT MCRS sets the write pointer for the *EnvRx* buffer to equal the EPAM value in the envelope header.
 - b) When receiving an envelope from an unregistered ONU, the OLT MCRS sets the write pointer according to the following equation:

$$WritePointer = ReadPointer \text{ XOR } 0x20$$

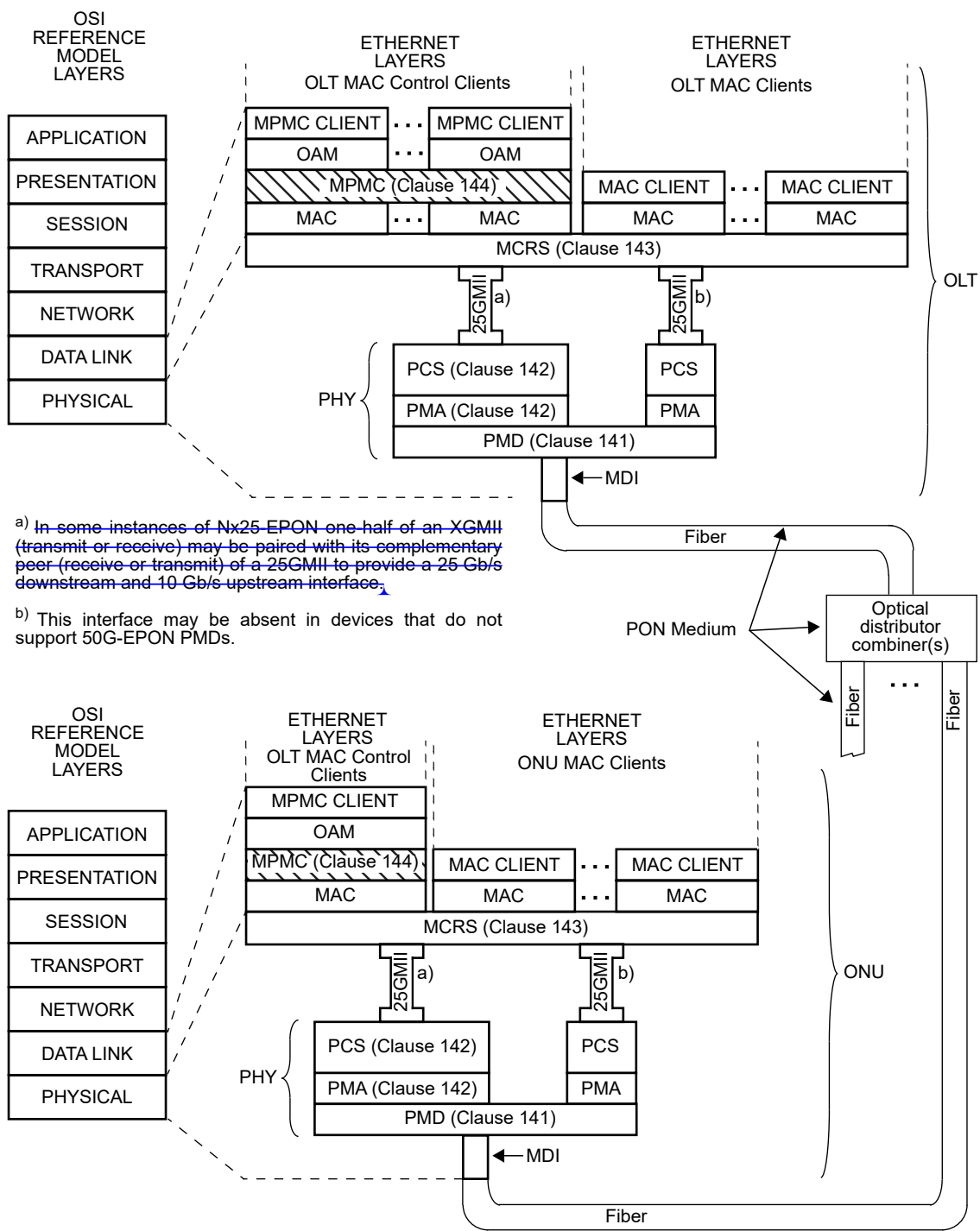
NOTE—The OLT MCRS determines that an envelope is from an unregistered ONU by either checking the LLID value in the envelope header (DISC_PLID) or by checking that an envelope header is received during the discovery window (see 144.1.1.3). Otherwise, the envelope is assumed to be received from a registered ONU.

- 2) Read Pointer
 - a) The read pointer increments synchronously with the *LocalTime* counter, which is locked to the ~~xMH~~ transmit clock.
 - b) If the OLT implements multiple transmit channels, all these channels share the same ~~xMH~~ transmit clock. Correspondingly, the read pointers for all channels increment synchronously and maintain equal values.

The above set of rules ensures that a delay of 32 EQT is built into the ONU MCRS receive path and a similar delay of 32 EQT is built into the OLT MCRS receive path. Therefore, the total round-trip delay measured by the MPCP (see 144.3.1.1) during an ONU discovery and registration includes a built-in margin of 64 EQT that is used to eliminate skew between different channels or the timing jitter within a channel.

143.4.3 Delay variability constraints

The MPCP relies on strict timing based on the distribution of timestamps. The MCRS is designed to allow a delay variability of up to 64 EQTs. During the normal operation of a registered ONU, the delay any EQ experiences in the *EnvRx* buffer is complementary to the accumulated skew and jitter that this EQ encounters after leaving the *EnvTx* buffer in the transmitting MCRS, such that the sum of the two delays remains constant.



a) In some instances of Nx25-EPON one half of an XGMII (transmit or receive) may be paired with its complementary peer (receive or transmit) of a 25GMII to provide a 25 Gb/s downstream and 10 Gb/s upstream interface.

b) This interface may be absent in devices that do not support 50G-EPON PMDs.



MPMC described in this clause

25GMII=25 GIGABIT MEDIA INDEPENDENT INTERFACE
 MDI = MEDIUM DEPENDENT INTERFACE
 OAM = OPERATIONS, ADMINISTRATION & MAINTENANCE
 OLT = OPTICAL LINE TERMINAL
 MCRS= MULTI-CHANNEL RECONCILIATION SUBLAYER
 MPMC= MULTI-POINT MAC CONTROL

ONU = OPTICAL NETWORK UNIT
 PCS = PHYSICAL CODING SUBLAYER
 PHY = PHYSICAL LAYER DEVICE
 PMA = PHYSICAL MEDIUM ATTACHMENT
 PMD = PHYSICAL MEDIUM DEPENDENT

Figure 144-2—Relationship of EPON P2MP PMD to the ISO/IEC OSI reference model and the IEEE 802.3 Ethernet model

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54