802.3dj D1.4
Comment Resolution
Logic Track

Gary Nicholl (Cisco), Logic Track Lead Editor
Eugene Opsasnick (Broadcom), Logic Editor
Matt Brown (Alphawave Semi), 802.3dj Chief Editor

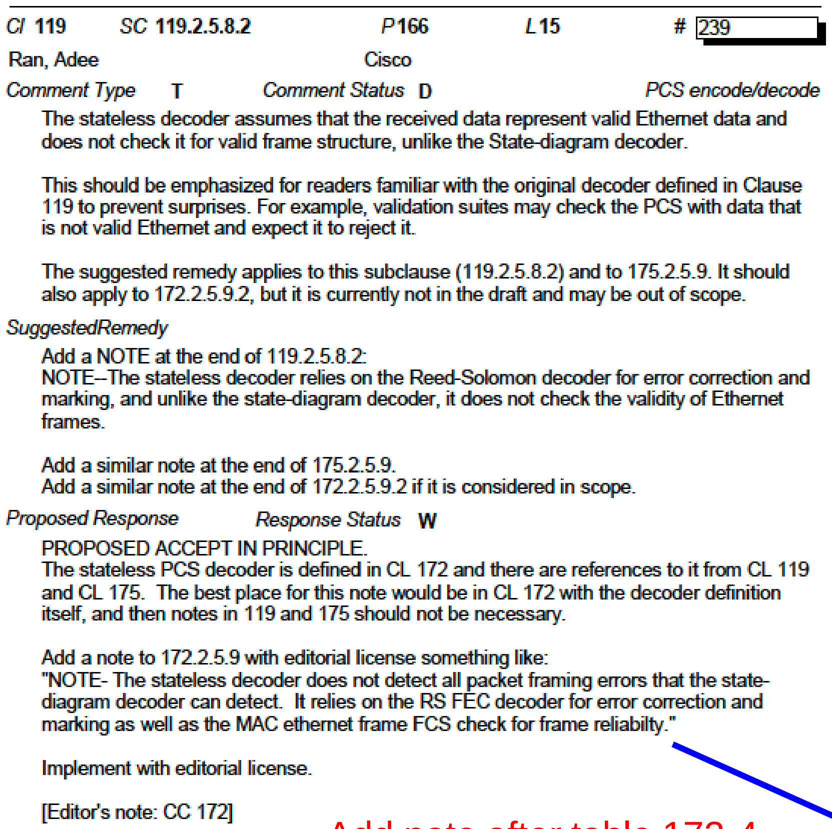
Introduction

- This slide package was assembled by the 802.3dj editorial team to provide background and detailed resolutions to aid in comment resolution.
- Specifically, these slides are for the logic track comments

PCS Decode

Comment #239

Comment #239



Cl 119 *SC* 119.2.5.8.2 *P*166 *L*15 # 239

Ran, Adee Cisco

Comment Type T *Comment Status* D *PCS encode/decode*

The stateless decoder assumes that the received data represent valid Ethernet data and does not check it for valid frame structure, unlike the State-diagram decoder.

This should be emphasized for readers familiar with the original decoder defined in Clause 119 to prevent surprises. For example, validation suites may check the PCS with data that is not valid Ethernet and expect it to reject it.

The suggested remedy applies to this subclause (119.2.5.8.2) and to 175.2.5.9. It should also apply to 172.2.5.9.2, but it is currently not in the draft and may be out of scope.

SuggestedRemedy

Add a NOTE at the end of 119.2.5.8.2:
NOTE--The stateless decoder relies on the Reed-Solomon decoder for error correction and marking, and unlike the state-diagram decoder, it does not check the validity of Ethernet frames.

Add a similar note at the end of 175.2.5.9.
Add a similar note at the end of 172.2.5.9.2 if it is considered in scope.

Proposed Response *Response Status* W

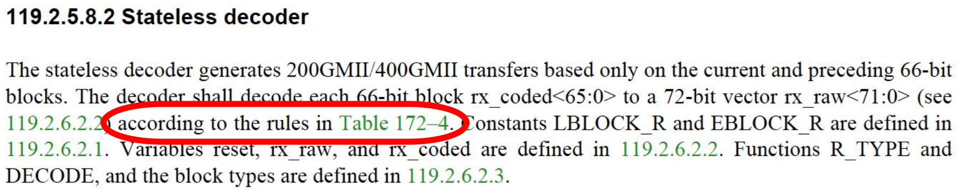
PROPOSED ACCEPT IN PRINCIPLE.
The stateless PCS decoder is defined in CL 172 and there are references to it from CL 119 and CL 175. The best place for this note would be in CL 172 with the decoder definition itself, and then notes in 119 and 175 should not be necessary.

Add a note to 172.2.5.9 with editorial license something like:
"NOTE- The stateless decoder does not detect all packet framing errors that the state-diagram decoder can detect. It relies on the RS FEC decoder for error correction and marking as well as the MAC ethernet frame FCS check for frame reliabilty."

Implement with editorial license.

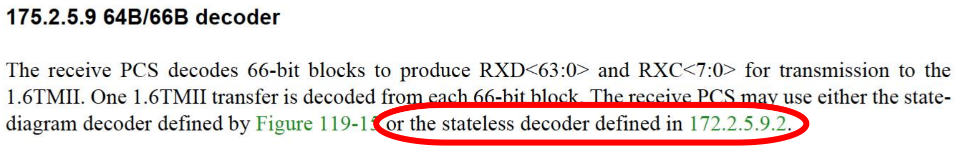
[Editor's note: CC 172]

Add note after table 172-4.



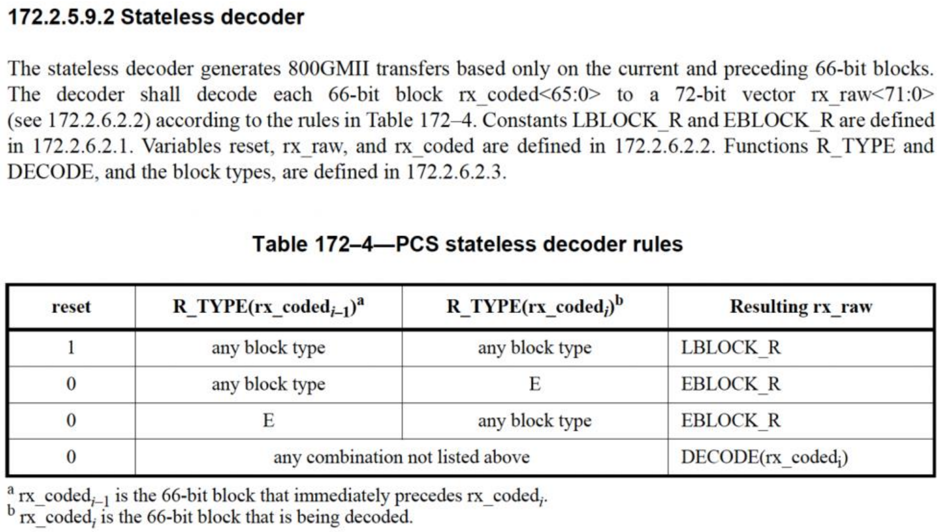
119.2.5.8.2 Stateless decoder

The stateless decoder generates 200GMII/400GMII transfers based only on the current and preceding 66-bit blocks. The decoder shall decode each 66-bit block rx_coded<65:0> to a 72-bit vector rx_raw<71:0> (see 119.2.6.2.2) according to the rules in Table 172–4. Constants LBLOCK_R and EBLOCK_R are defined in 119.2.6.2.1. Variables reset, rx_raw, and rx_coded are defined in 119.2.6.2.2. Functions R_TYPE and DECODE, and the block types are defined in 119.2.6.2.3.



175.2.5.9 64B/66B decoder

The receive PCS decodes 66-bit blocks to produce RXD<63:0> and RXC<7:0> for transmission to the 1.6TMII. One 1.6TMII transfer is decoded from each 66-bit block. The receive PCS may use either the state-diagram decoder defined by Figure 119-15 or the stateless decoder defined in 172.2.5.9.2.



172.2.5.9.2 Stateless decoder

The stateless decoder generates 800GMII transfers based only on the current and preceding 66-bit blocks. The decoder shall decode each 66-bit block rx_coded<65:0> to a 72-bit vector rx_raw<71:0> (see 172.2.6.2.2) according to the rules in Table 172–4. Constants LBLOCK_R and EBLOCK_R are defined in 172.2.6.2.1. Variables reset, rx_raw, and rx_coded are defined in 172.2.6.2.2. Functions R_TYPE and DECODE, and the block types, are defined in 172.2.6.2.3.

Table 172–4—PCS stateless decoder rules

reset	R_TYPE(rx_coded$_{i-1}$)[a]	R_TYPE(rx_coded$_i$)[b]	Resulting rx_raw
1	any block type	any block type	LBLOCK_R
0	any block type	E	EBLOCK_R
0	E	any block type	EBLOCK_R
0	any combination not listed above		DECODE(rx_coded$_i$)

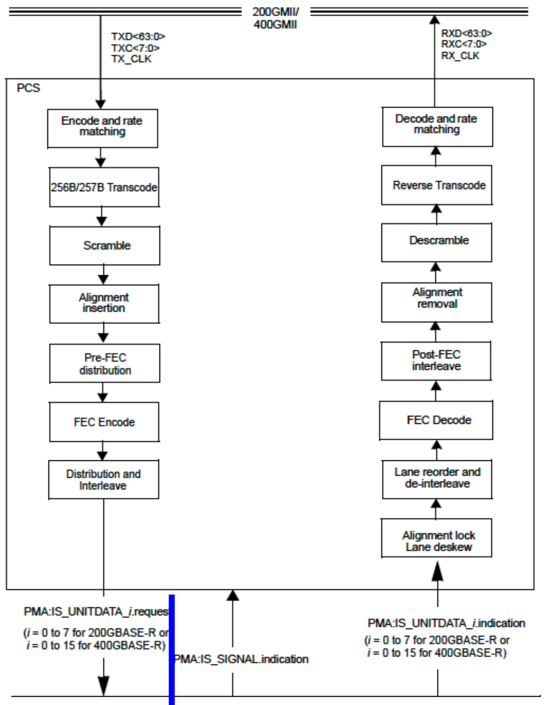
[a] rx_coded$_{i-1}$ is the 66-bit block that immediately precedes rx_coded$_i$.
[b] rx_coded$_i$ is the 66-bit block that is being decoded.

Note - The stateless decoder does not detect …

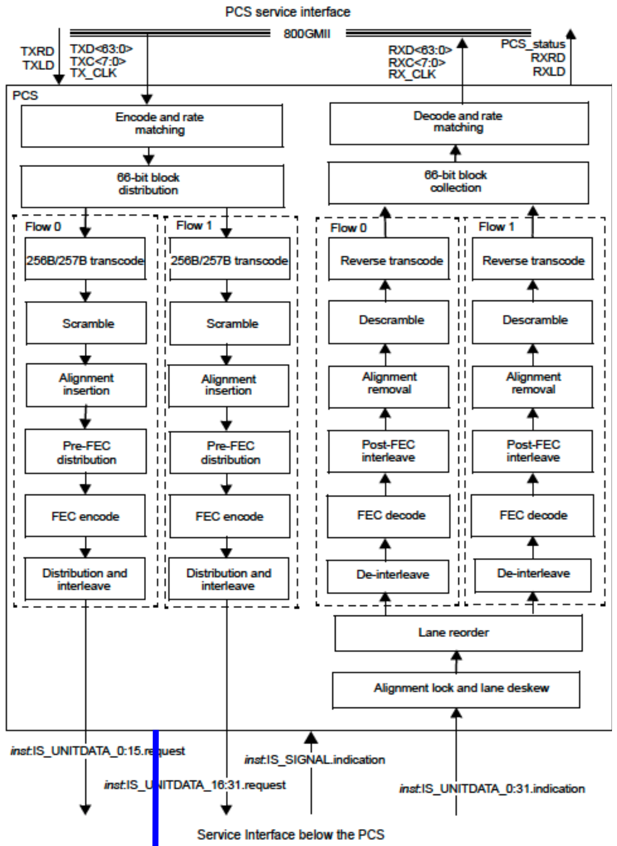
PCS IS_SIGNAL.request

Comments #248, #251, #236

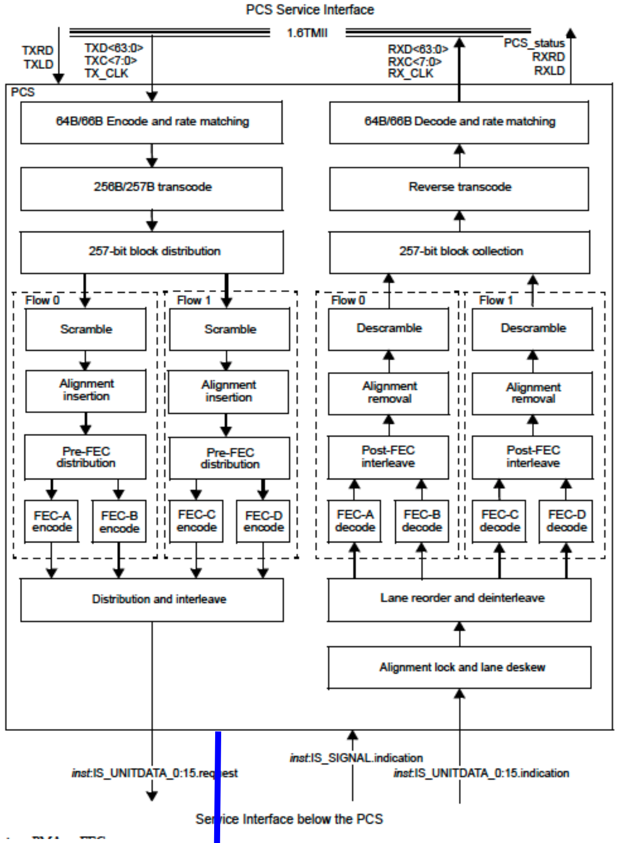
Comment #248 - 200G/400G PCS, 800G PCS, 1.6T PCS



PMA:IS_SIGNAL.request



inst:IS_SIGNAL.request



inst:IS_SIGNAL.request

Comment #251 - PMA IS_SIGNAL Generation

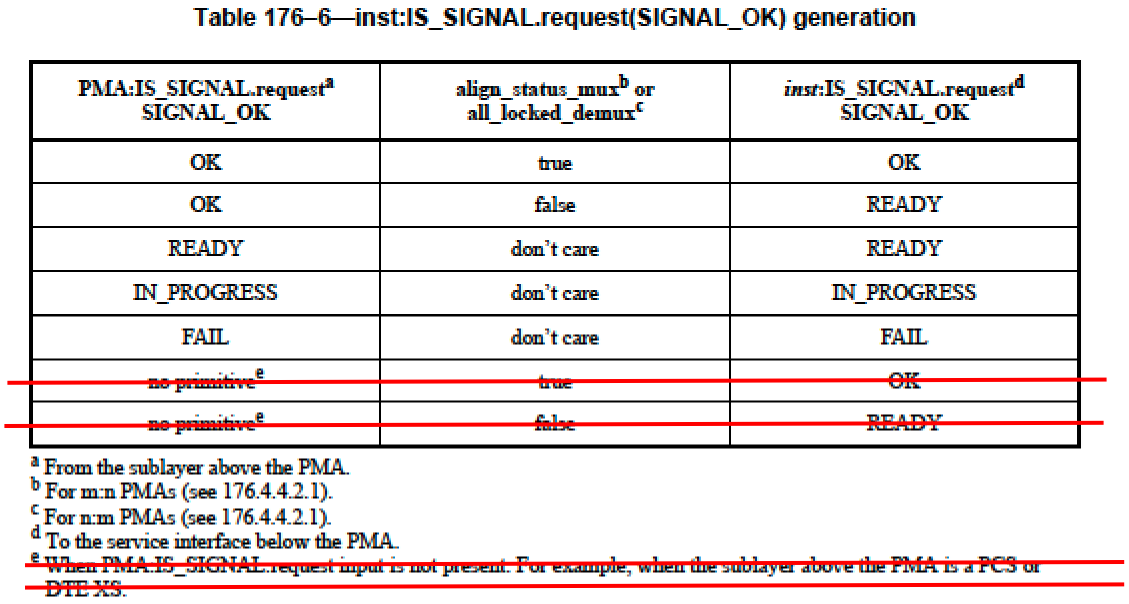


Table 176–6—inst:IS_SIGNAL.request(SIGNAL_OK) generation

PMA:IS_SIGNAL.request[a] SIGNAL_OK	align_status_mux[b] or all_locked_demux[c]	*inst*:IS_SIGNAL.request[d] SIGNAL_OK
OK	true	OK
OK	false	READY
READY	don't care	READY
IN_PROGRESS	don't care	IN_PROGRESS
FAIL	don't care	FAIL
~~no primitive[e]~~	~~true~~	~~OK~~
~~no primitive[e]~~	~~false~~	~~READY~~

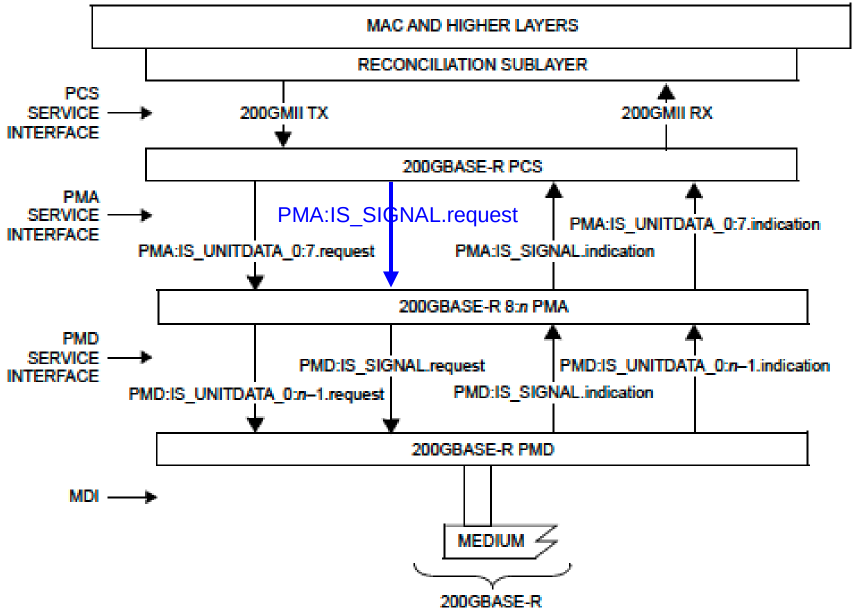
[a] From the sublayer above the PMA.
[b] For m:n PMAs (see 176.4.4.2.1).
[c] For n:m PMAs (see 176.4.4.2.1).
[d] To the service interface below the PMA.
~~[e] When PMA:IS_SIGNAL.request input is not present. For example, when the sublayer above the PMA is a PCS or DTE XS.~~

PMA:IS_SIGNAL.request (from the sublayer above the PMA) can come from:

- Another PMA
- AUI
- PCS (newly added input)
- DTE XS (newly added input)

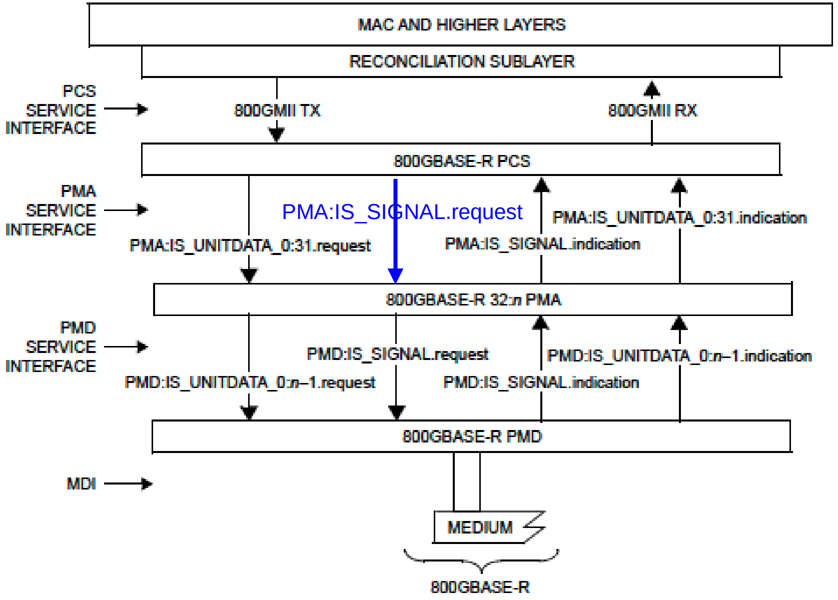
inst:IS_SIGNAL.request generation is more clear and consistent.

Comment #236 - Example overview figures for 200G and 800G



200GMII = 200 Gb/s MEDIA INDEPENDENT INTERFACE
MAC = MEDIA ACCESS CONTROL
MDI = MEDIUM DEPENDENT INTERFACE
PCS = PHYSICAL CODING SUBLAYER
PMA = PHYSICAL MEDIUM ATTACHMENT
PMD = PHYSICAL MEDIUM DEPENDENT
n = NUMBER OF PARALLEL STREAMS OF DATA UNITS

Figure 116–2—200GBASE-R inter-sublayer service interfaces



800GMII = 800 Gb/s MEDIA INDEPENDENT INTERFACE
MAC = MEDIA ACCESS CONTROL
MDI = MEDIUM DEPENDENT INTERFACE
PCS = PHYSICAL CODING SUBLAYER
PMA = PHYSICAL MEDIUM ATTACHMENT
PMD = PHYSICAL MEDIUM DEPENDENT
n = NUMBER OF PARALLEL STREAMS OF DATA UNITS

Figure 169–2—800GBASE-R inter-sublayer service interfaces not including 800GMII Extender

Comment #236 - examples with extender

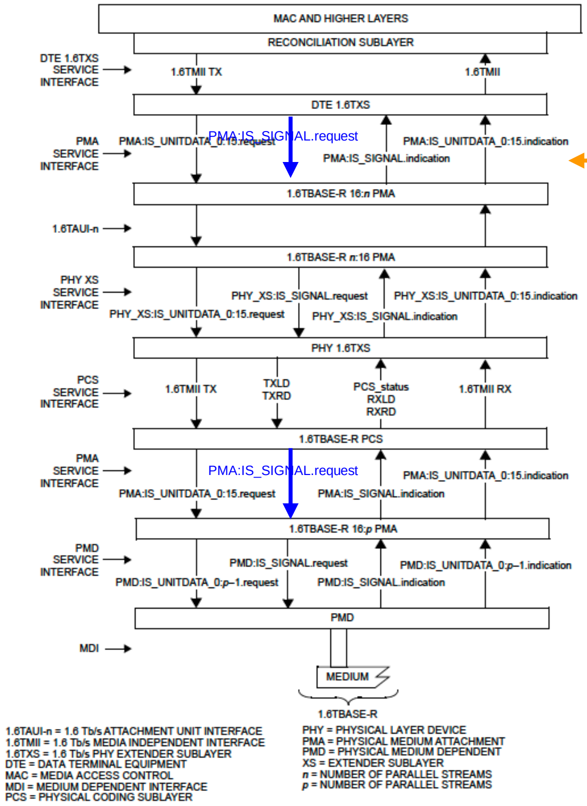

MAC AND HIGHER LAYERS
RECONCILIATION SUBLAYER
DTE 1.6TXS SERVICE INTERFACE
1.6TMII TX
1.6TMII
DTE 1.6TXS
PMA SERVICE INTERFACE
PMA:IS_UNITDATA_0:15.request
PMA:IS_SIGNAL.request
PMA:IS_UNITDATA_0:15.indication
PMA:IS_SIGNAL.indication
1.6TBASE-R 16:n PMA
1.6TAUI-n
1.6TBASE-R n:16 PMA
PHY XS SERVICE INTERFACE
PHY_XS:IS_SIGNAL.request
PHY_XS:IS_UNITDATA_0:15.indication
PHY_XS:IS_UNITDATA_0:15.request
PHY_XS:IS_SIGNAL.indication
PHY 1.6TXS
PCS SERVICE INTERFACE
1.6TMII TX
TXLD
TXRD
PCS_status
RXLD
RXRD
1.6TMII RX
1.6TBASE-R PCS
PMA SERVICE INTERFACE
PMA:IS_SIGNAL.request
PMA:IS_UNITDATA_0:15.indication
PMA:IS_UNITDATA_0:15.request
PMA:IS_SIGNAL.indication
1.6TBASE-R 16:p PMA
PMD SERVICE INTERFACE
PMD:IS_SIGNAL.request
PMD:IS_UNITDATA_0:p–1.indication
PMD:IS_UNITDATA_0:p–1.request
PMD:IS_SIGNAL.indication
PMD
MDI
MEDIUM
1.6TBASE-R
1.6TAUI-n = 1.6 Tb/s ATTACHMENT UNIT INTERFACE
1.6TMII = 1.6 Tb/s MEDIA INDEPENDENT INTERFACE
1.6TXS = 1.6 Tb/s PHY EXTENDER SUBLAYER
DTE = DATA TERMINAL EQUIPMENT
MAC = MEDIA ACCESS CONTROL
MDI = MEDIUM DEPENDENT INTERFACE
PCS = PHYSICAL CODING SUBLAYER
PHY = PHYSICAL LAYER DEVICE
PMA = PHYSICAL MEDIUM ATTACHMENT
PMD = PHYSICAL MEDIUM DEPENDENT
XS = EXTENDER SUBLAYER
n = NUMBER OF PARALLEL STREAMS
p = NUMBER OF PARALLEL STREAMS

Figure 174–3—Inter-sublayer service interfaces for a 1.6TBASE-R PHY and 1.6TMII Extender

Fig. 174-3 Is a typical xBASE-R PHY with extender (Copper/IMDD PHYs).

- IS_SIGNAL.request added to both DTE and PCS Tx outputs

Fig. 185-2 and 185-3 are a special case for coherent PHY 800GBASE-LR1

- Coherent links do not support ILT
- Per comment #21, Fig. 185-3 is removed Instructions for changes to Fig. 185-3 in comment #236 can be ignored.
- Per comment #21, Fig. 185-2 is adding an 800G-AUI.
- It is unclear how IS_SIGNAL.request should be added to the new inter-sublayer interfaces in Fig. 185-2

(Should pull #21 from bucket and resolve along with comments #236 and #248).

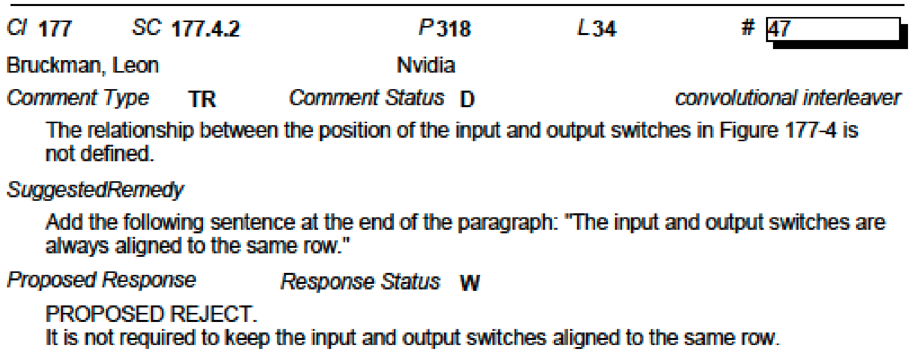
Add new proposed diagram for Figure 185-2 here. (Per comment #21)

Include any new IS_SIGNAL.request and IS_SIGNAL.indication

Convolutional Interleaver

Comment #47

Comment #47 - Clause 177, Convolutional Interleaver



Cl 177	*SC* 177.4.2	*P* 318	*L* 34	# 47

Bruckman, Leon Nvidia

Comment Type **TR** *Comment Status* **D** *convolutional interleaver*

The relationship between the position of the input and output switches in Figure 177-4 is not defined.

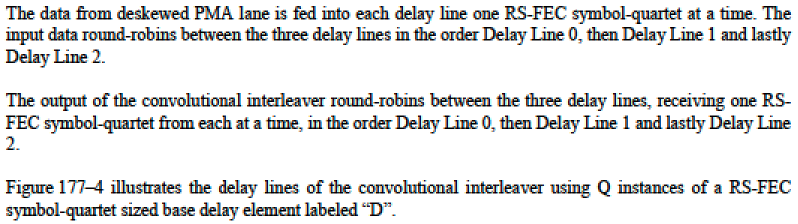
SuggestedRemedy

Add the following sentence at the end of the paragraph: "The input and output switches are always aligned to the same row."

Proposed Response *Response Status* **W**

PROPOSED REJECT.
It is not required to keep the input and output switches aligned to the same row.

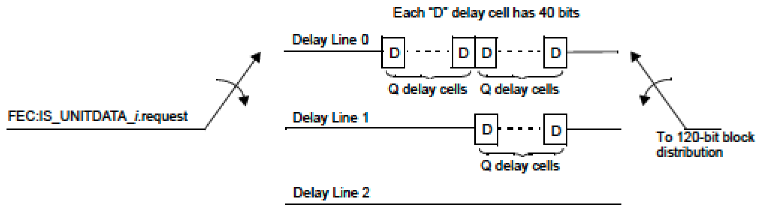
From 177.4.2:



The data from deskewed PMA lane is fed into each delay line one RS-FEC symbol-quartet at a time. The input data round-robins between the three delay lines in the order Delay Line 0, then Delay Line 1 and lastly Delay Line 2.

The output of the convolutional interleaver round-robins between the three delay lines, receiving one RS-FEC symbol-quartet from each at a time, in the order Delay Line 0, then Delay Line 1 and lastly Delay Line 2.

Figure 177–4 illustrates the delay lines of the convolutional interleaver using Q instances of a RS-FEC symbol-quartet sized base delay element labeled "D".



Each "D" delay cell has 40 bits
Delay Line 0
D
D
D
D
Q delay cells
Q delay cells
FEC:IS_UNITDATA_i.request
Delay Line 1
D
D
Q delay cells
To 120-bit block distribution
Delay Line 2

Figure 177–4—Convolutional interleaver

Comment #47 - Comparison to CL 184 Convolutional Interleaver

From 184.4.5:

The following list describes the convolutional interleaver process:

a) The input and output switches are always aligned to the same row *b*, where *b* = 0 to 2
b) A block of 40 bits is read from row *b*.
c) The contents of row *b* are shifted to the right by 40 bits.
d) A block of 40 bits is written to row *b*.
e) The position of the switches is updated to (*b* + 1) mod 3

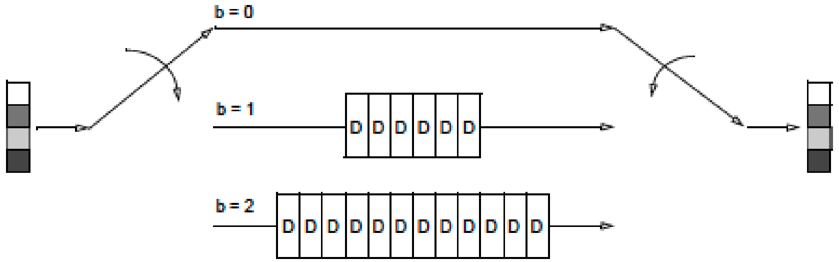


Figure 184–4—Convolutional interleaver

184.4.5: Convolutional Interleaver description

Part a) Input and Output switches are always aligned to the same row.

Is this not necessary?

ER1 MDIO

Comments #106, #107

Variables related to the inverse FEC sublayer

General strategy: add references to the clause 186 subclause for the related function, as well as to the clause 172 subclause that defines the variable

Comment #106 and #107