

# Logic Baseline proposal for 800G single-wavelength coherent PHY with concatenated FEC

Kishore Kota – Marvell

Eric Maniloff – Ciena

Arnon Loewenthal – Alphawave

P802.3dj Plenary Meeting, Berlin, Germany, July 2023

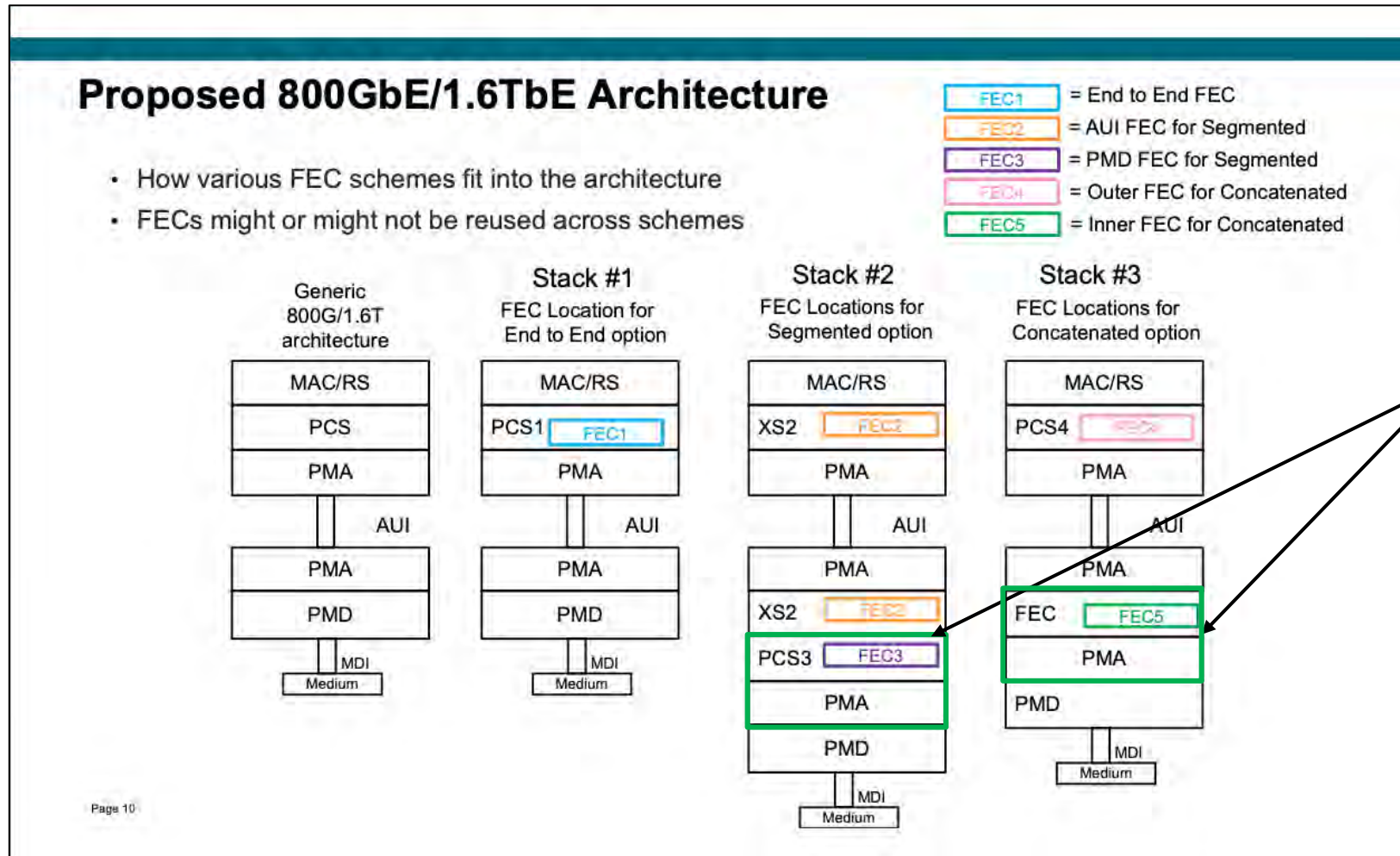
# Supporters

- Ian Betty – Ciena
- Paul Brooks - Viavi
- David Cassan – Alphawave
- Tony Chan Carusone – Alphawave
- Frank Chang – Source Photonics
- Piers Dawe - Nvidia
- Arash Farhood – Marvell
- Sebastien Gareau – Ciena
- Ali Ghiasi – Ghiasi Quantum
- Tao Gui – Huawei
- James Harley – Ciena
- Xiang He – Huawei
- Kechao Huang – Huawei
- Hideki Isono – Fujitsu Optical Components
- Madhu Krishnamurthy – Lumentum
- Michael Strunz-Kroll - euNetworks
- Cedric Lam – Google
- Samuel Liu – Marvell
- Wendell Liu – AT&T
- Xiang Liu – Huawei
- Jeff Maki – Juniper
- Guangcan Mi - Huawei
- Ernest Muhigana – Lumentum
- Jeff Nichols – Ciena
- Gurinder Parhar– Source Photonics
- Charles Park - Juniper
- Lenin Patra – Marvell
- Atul Strivastava - NEL
- Peter Stassar – Huawei
- Dan Tauber – Lumentum
- Or Vidal – Alphawave
- Chongjin Xie - Alibaba
- Shuang Yin - Google
- Bo Zhang – Marvell
- Xiang Zhou – Google
- Yanjun Zhu - Hisense

# Logic Baseline Requirements

- Support 800GE MAC
- Support single-wavelength 800G coherent line interface
- Adhere to the adopted logic architecture ([gustlin 3df 01a 220517](#))
- Support up to 2xAUIs per link side
  - AUIs are lane rate independent (for e.g. one AUI at 200G/lane and another at 100G/lane)
- Support FEC requirements of single-wavelength coherent interfaces
- Support requirements which are unique to single-wavelength coherent interfaces (such as burst error tolerance, unequal bit error rates, laser phase noise tolerance, ability to recover carrier offset etc.)

# Adopted logic architecture for reference

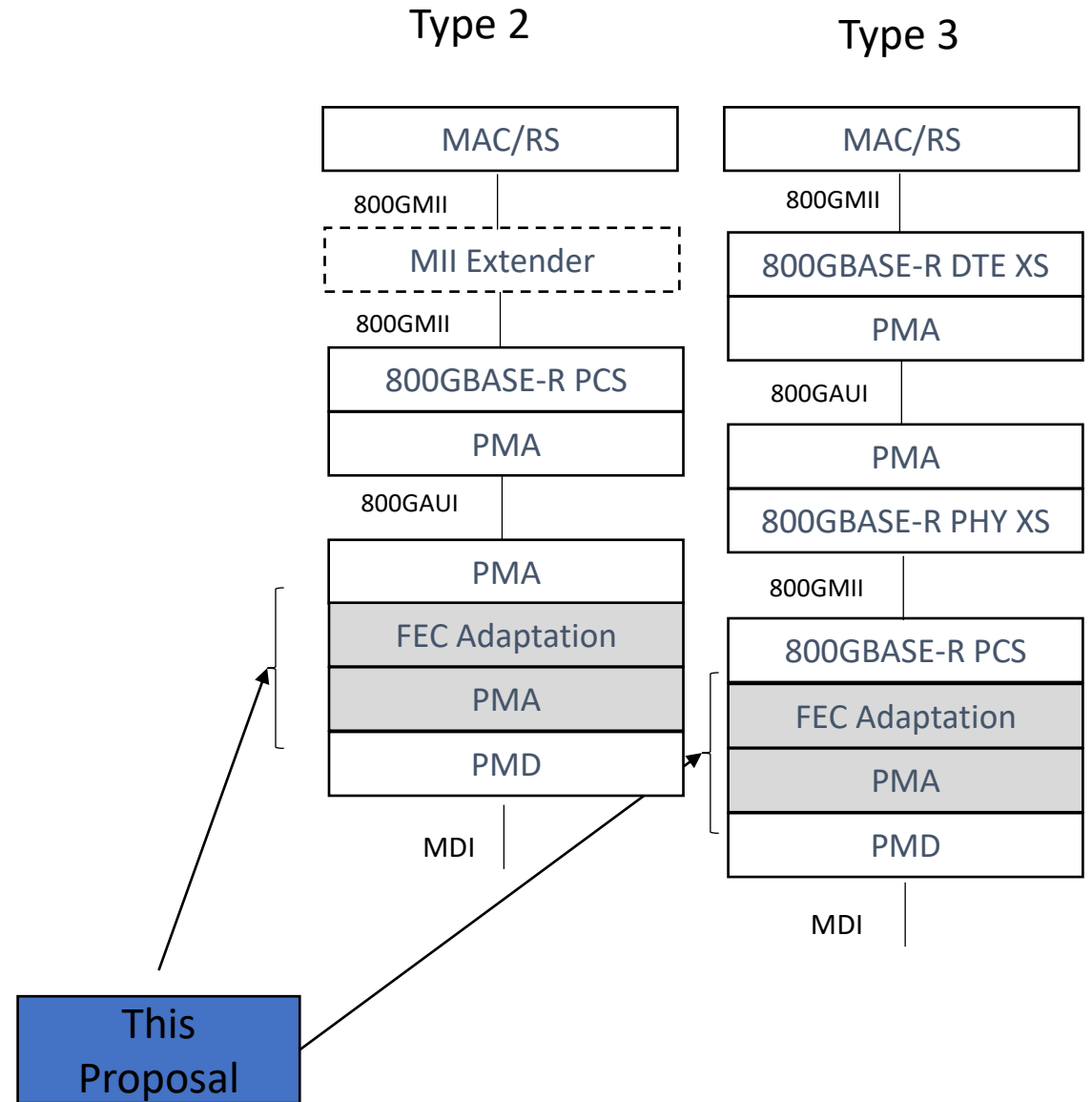


This proposal covers the FEC/PMA sublayer definition for single-wavelength 800G coherent interfaces

[gustlin\\_3df\\_01a\\_220517.pdf](#)

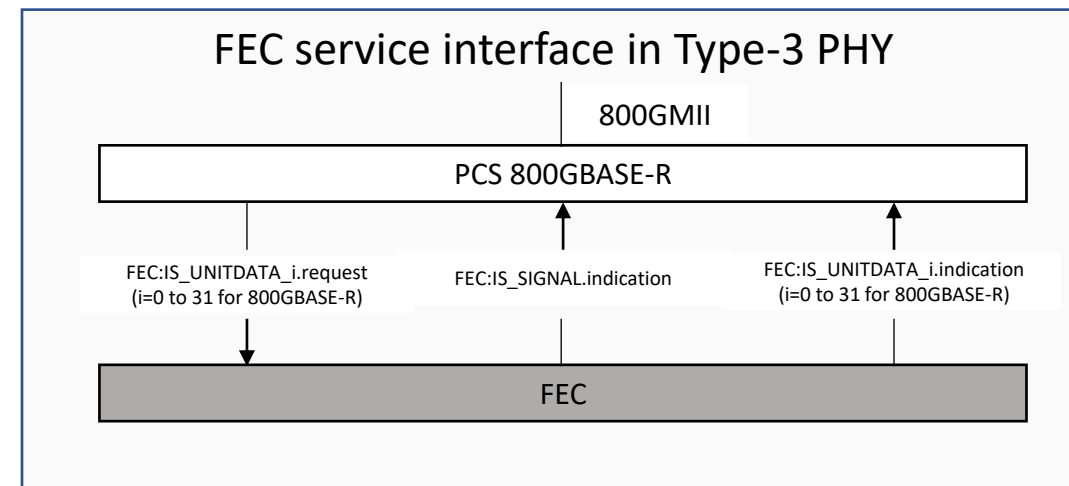
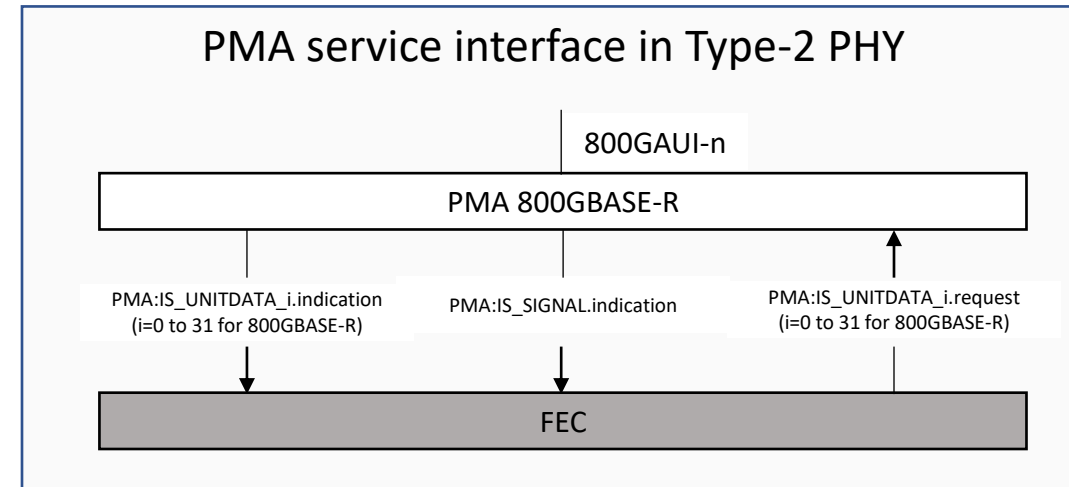
# This proposal

- Based on a Type 2 FEC/PHY scheme (“concatenated PHY”) as described in [brown 3dj optx adhoc 01a 230222](#)
  - Same architecture as [patra 3dj 01b 2303](#)
  - Supports Type 3 (“segmented PHY”) using extended sublayer, if needed
- AUIs are supported using the 800GMII Extender
  - 800GAUI-8 defined in 802.3df
  - 800GAUI-4 will be defined in 802.3dj

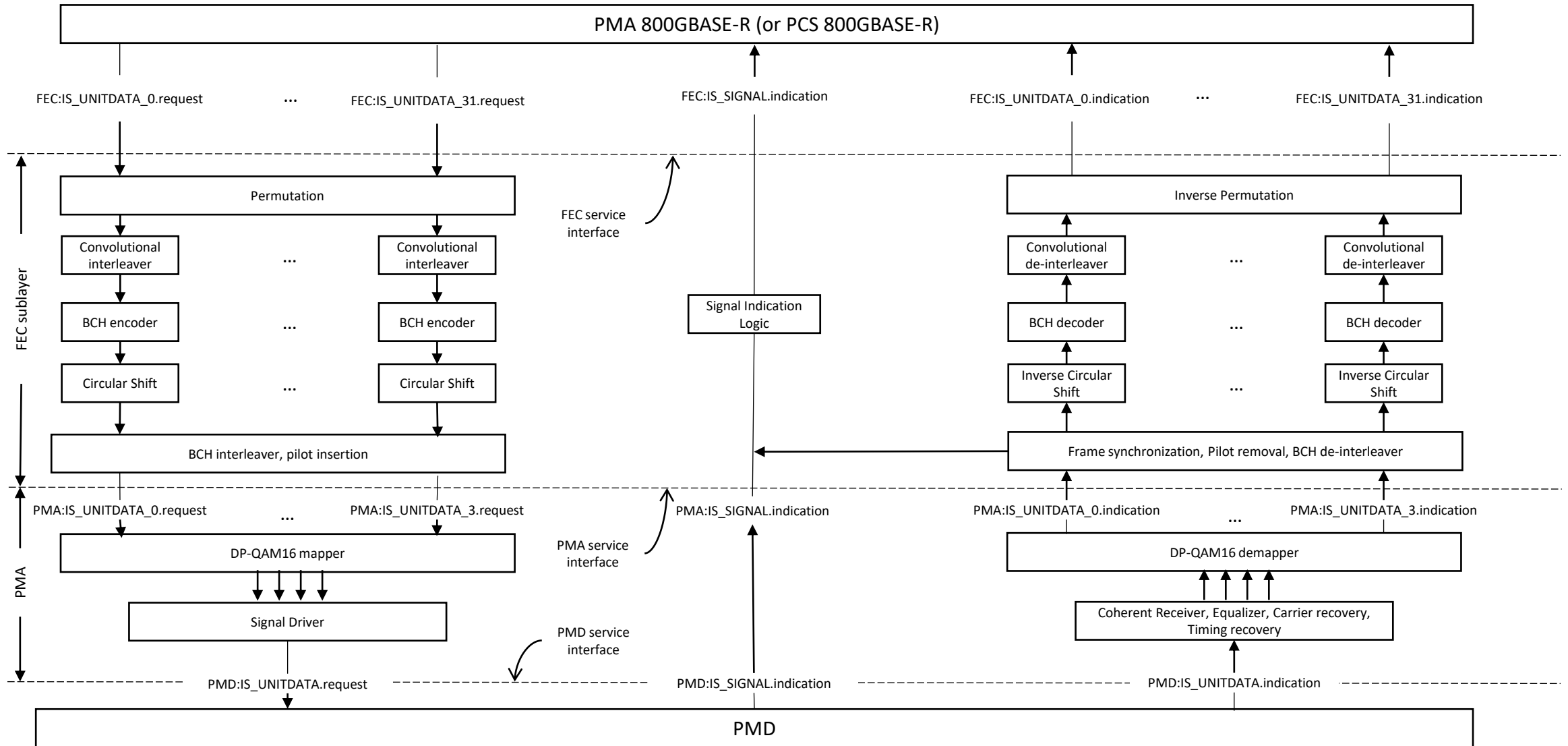


# FEC Service interface

- The FEC service interface allows the 800GBASE-R PCS to transfer information to and from the 800G PHY in Type-3 (“segmented”) PHYs
- This interface supports the exchange of encoded data between the PCS and the FEC sublayer
- This interface is identical to the 800GBASE-R PMA service interface



# FEC/PMA Sublayer Block Diagram

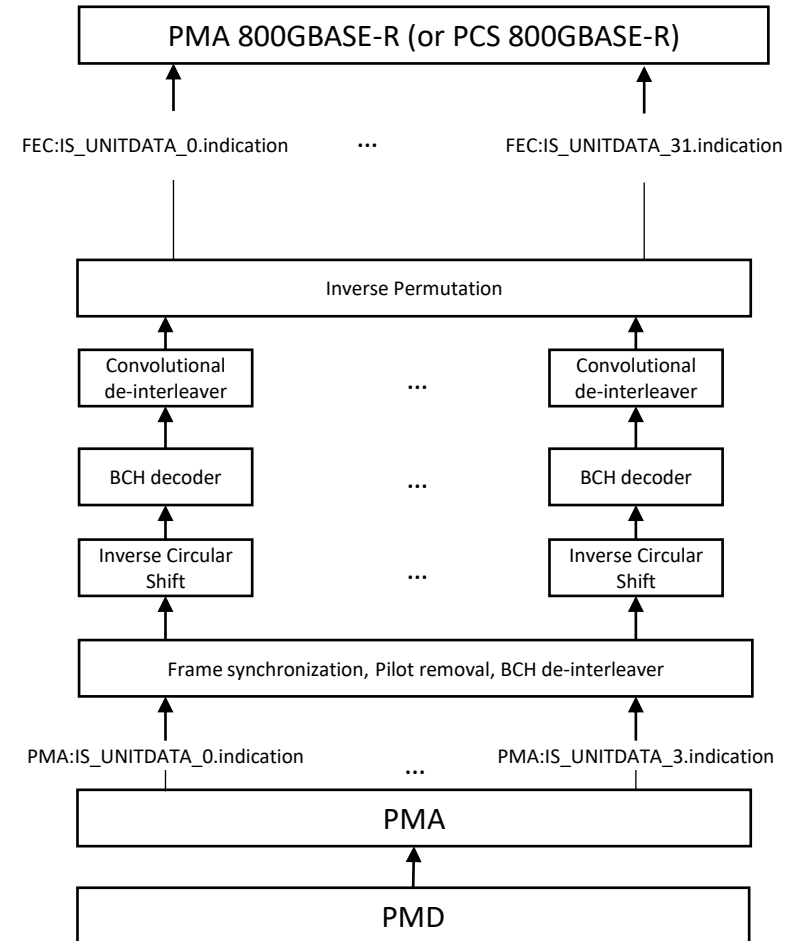




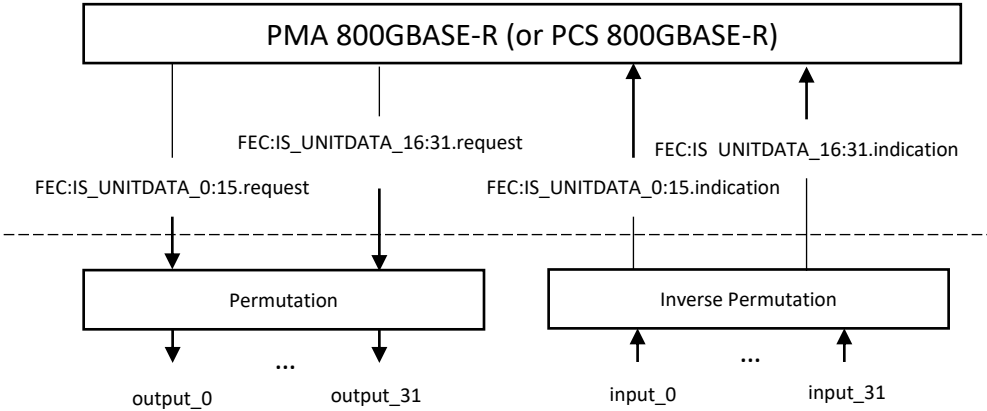


# FEC Sublayer Receive Path

- Decode 4 physical lanes (XI/XQ/YI/YQ) into 32 PCS lanes
- Most of the decoding in the FEC layer is done per PCS lane
- Frame synchronization can be done based on pilot symbols



# Permutation/Inverse Permutation Functions



## Pictorial Description of the permutation function

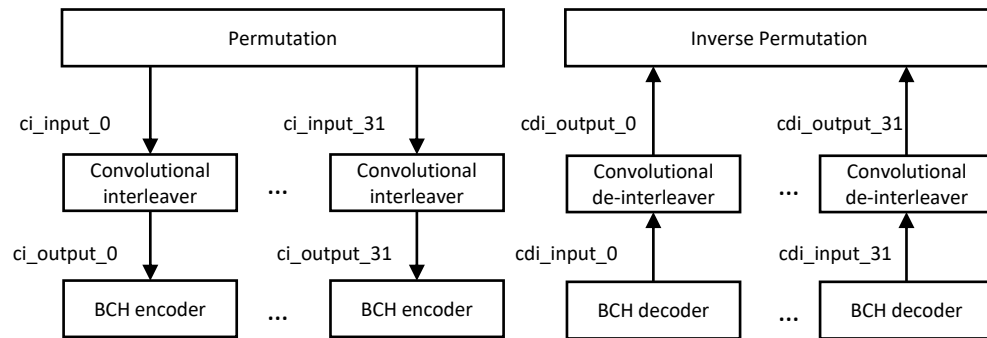
|         | 0 | 1  | 2  | 3  |
|---------|---|----|----|----|
| Lane 0  |   |    |    |    |
| Lane 1  | 0 | 8  | 16 | 24 |
| Lane 2  |   |    |    |    |
| Lane 3  |   |    |    |    |
| Lane 4  |   |    |    |    |
| Lane 5  | 1 | 9  | 17 | 25 |
| Lane 6  |   |    |    |    |
| Lane 7  |   |    |    |    |
| Lane 8  |   |    |    |    |
| Lane 9  | 2 | 10 | 18 | 26 |
| Lane 10 |   |    |    |    |
| Lane 11 |   |    |    |    |
| Lane 12 |   |    |    |    |
| Lane 13 | 3 | 11 | 19 | 27 |
| Lane 14 |   |    |    |    |
| Lane 15 |   |    |    |    |
| Lane 16 |   |    |    |    |
| Lane 17 | 4 | 12 | 20 | 28 |
| Lane 18 |   |    |    |    |
| Lane 19 |   |    |    |    |
| Lane 20 |   |    |    |    |
| Lane 21 | 5 | 13 | 21 | 29 |
| Lane 22 |   |    |    |    |
| Lane 23 |   |    |    |    |
| Lane 24 |   |    |    |    |
| Lane 25 | 6 | 14 | 22 | 30 |
| Lane 26 |   |    |    |    |
| Lane 27 |   |    |    |    |
| Lane 28 |   |    |    |    |
| Lane 29 | 7 | 15 | 23 | 31 |
| Lane 30 |   |    |    |    |
| Lane 31 |   |    |    |    |



|         | 0 | 1  | 2  | 3  |
|---------|---|----|----|----|
| Lane 0  |   |    |    |    |
| Lane 1  | 0 | 8  | 20 | 28 |
| Lane 2  |   |    |    |    |
| Lane 3  |   |    |    |    |
| Lane 4  |   |    |    |    |
| Lane 5  | 1 | 9  | 21 | 29 |
| Lane 6  |   |    |    |    |
| Lane 7  |   |    |    |    |
| Lane 8  |   |    |    |    |
| Lane 9  | 2 | 10 | 22 | 30 |
| Lane 10 |   |    |    |    |
| Lane 11 |   |    |    |    |
| Lane 12 |   |    |    |    |
| Lane 13 | 3 | 11 | 23 | 31 |
| Lane 14 |   |    |    |    |
| Lane 15 |   |    |    |    |
| Lane 16 |   |    |    |    |
| Lane 17 | 4 | 12 | 16 | 24 |
| Lane 18 |   |    |    |    |
| Lane 19 |   |    |    |    |
| Lane 20 |   |    |    |    |
| Lane 21 | 5 | 13 | 17 | 25 |
| Lane 22 |   |    |    |    |
| Lane 23 |   |    |    |    |
| Lane 24 |   |    |    |    |
| Lane 25 | 6 | 14 | 18 | 26 |
| Lane 26 |   |    |    |    |
| Lane 27 |   |    |    |    |
| Lane 28 |   |    |    |    |
| Lane 29 | 7 | 15 | 19 | 27 |
| Lane 30 |   |    |    |    |
| Lane 31 |   |    |    |    |

- Purpose: Provides 10bit symbols from 4 interleaved RS(544,514) codewords on each of 32 PCS lanes
- Prerequisites: 10bit symbol alignment across lanes. Lane reorder and FEC codeword deskew across lanes is optional
- Operates on 4 RS-symbol boundaries across 32 PCS lanes
- In the transmit direction, the permutation function maps the bits received on thirty-two PCS lanes through the FEC:IS\_UNITDATA\_0:31.request primitives and maps them to output lanes *output\_0:31*
  - Denote the index of the PCSL as  $p$  ( $p=0$  to  $31$ ), where the bits for the  $p^{th}$  lane are obtained through  $FEC:IS\_UNITDATA\_p.request$
  - Denote the index of the aligned input 10-bit block across the PCSLs as  $i$  and the bit index within block  $i$  as  $j$  ( $j=0$  to  $9$ ).
  - Denote the index of the output lanes as  $q$  ( $q=0$  to  $31$ )
  - The mapping between the bit sequences on the PCSL and the  $output\_q$  is:
    - $output\_q[10i + j] = PCSL[(q + 16[i/2])\%32, 10i + j]$
- In the receive direction, the inverse permutation function maps the bits received on input lanes *input\_0:31* to thirty-two PCS lanes and provides this data through the FEC:IS\_UNITDATA\_0:31.indication primitives. Using similar definitions as the transmit, the mapping between the bit sequences on  $input\_q$  lanes and the PCSL is:
  - $PCSL[(q + 16[i/2])\%32, 10i + j] = input\_q[10i + j]$

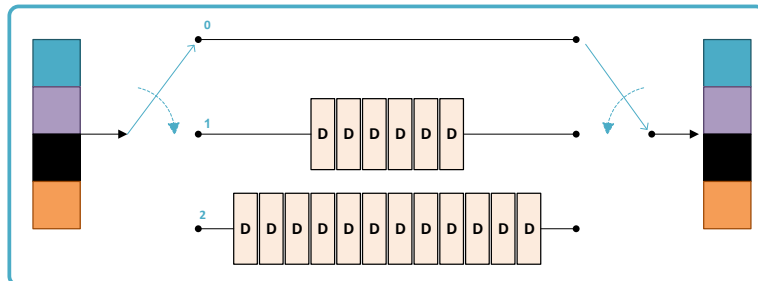
# Convolutional interleaver/de-interleaver Functions



- Purpose: Ensure each BCH codeword contributes no more than one 10-bit RS symbol to each RS codeword.
  - One Convolutional Interleaver per PCS lane.
  - Interleaver has three parallel delay lines
  - D represents storage of 40b
  - Word width is 4 symbols or 40 bits at I/O

- In the transmit direction, the convolutional interleaver function maps the bits received on ci\_input\_0:31 lanes from the permutation function and maps them to output lanes ci\_output\_0:31
  - Denote the input and output of the convolutional interleaver of the  $p^{\text{th}}$  lane as  $ci\_input\_p$  and  $ci\_output\_p$  respectively ( $p=0$  to 31)
  - Denote the index of the aligned output 40-bit block as  $i$  and the bit index within block  $i$  as  $j(j=0$  to 39). The block index  $i$  is synchronized to the pilot insertion function described in a subsequent slide. The 40-bit blocks are aligned to block boundaries of the permutation function.
  - The mapping between the bit sequences on  $ci\_input\_p$  and  $ci\_output\_p$  is:
    - $ci\_output\_p[40i + j] = ci\_input\_p[40 * (i - 18 * (i\%3)) + j]$

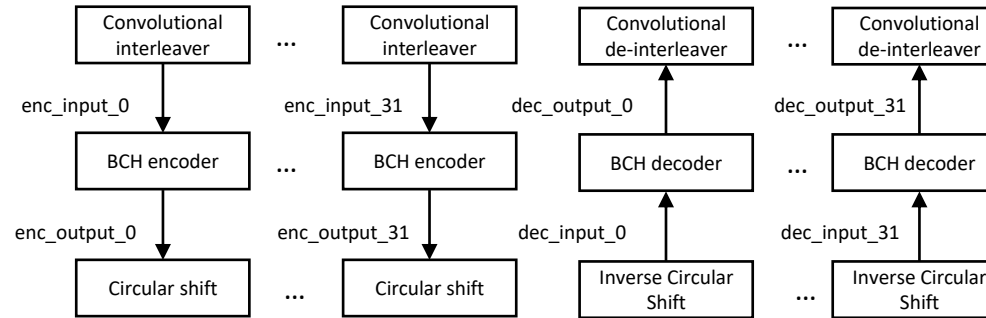
## Pictorial Description of the convolutional interleaver function



Interleaver Latency of 55 nanoseconds

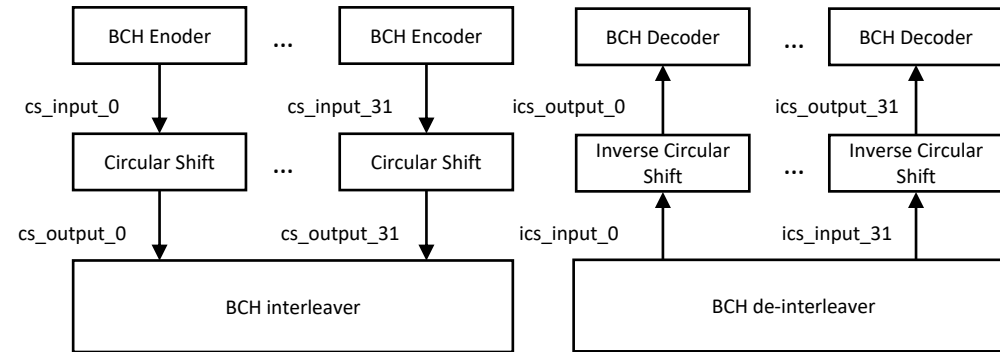
- In the receive direction, the convolutional de-interleaver function maps the bits received on cdi\_input\_0:31 lanes from the BCH decoder function and maps them to output lanes cdi\_output\_0:31
  - Denote the input and output of the convolutional de-interleaver of the  $p^{\text{th}}$  lane as  $cdi\_input\_p$  and  $cdi\_output\_p$  respectively ( $p=0$  to 31)
  - Denote the index of the aligned input 40-bit block as  $i$  and the bit index within block  $i$  as  $j(j=0$  to 39). The index  $i$  is synchronized to the frame synchronization function described in a subsequent slide
  - The mapping between the bit sequences on  $cdi\_input\_p$  and  $cdi\_output\_p$  is:
    - $cdi\_output\_p[40 * (i + 18 * (2 - i\%3)) + j] = cdi\_input\_p[40 * i + j]$

# BCH Encoder/Decoder Functions



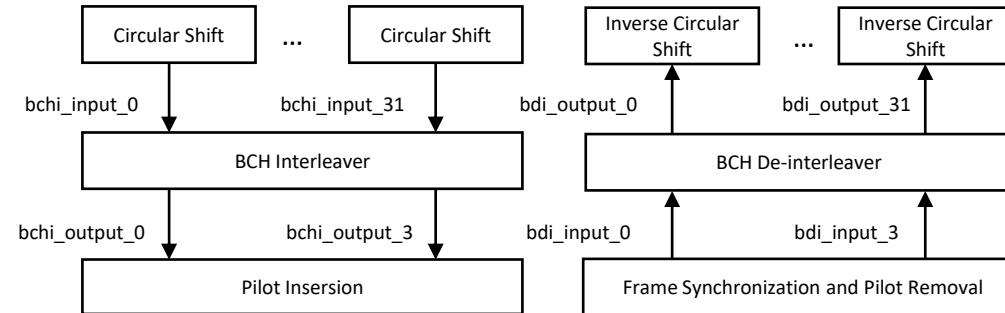
- Purpose: Inner code works in conjunction with outer KP4 FEC to provide a high-performance FEC for 800LR
  - BCH (126,110) code operates on 11 RS symbols at output of convolutional interleaver
  - Simple Chase decoders can achieve a pre-FEC BER threshold of  $\sim 1.1e-2$  for a post-KP4 BER of  $1e-15$
- Code definition:
  - Define  $c$  as a binary vector of length 126, and  $c(x)$  a polynomial of degree 125 with the coefficients defined by  $c$  (where the bit 0 of  $c$  represents the coefficient of power 125).
  - Then  $c$  is a codeword of the BCH(126,110) code if  $c(x)$  is divisible (module 2) by the defined binary generator polynomial  $g(x)$
- In the transmit direction, the BCH encoder function maps the bits received on `enc_input_0:31` lanes from the convolutional interleaver function and maps them to output lanes `enc_output_0:31`
  - The bits of `enc_input_p` ( $p=0$  to 31) are obtained from the convolutional interleaver output lanes `ci_output_p`. Denote the index of the aligned 40-bit block at the convolutional interleaver output as  $i$  and the bit index within block  $i$  as  $j(j=0$  to 39) where  $i$  and  $j$  are identical to the indices used in the definition of the convolutional interleaver function.
  - Denote the index of the BCH encoder block as  $u$  and the bit index within the input block is  $v$  ( $v=0$  to 109). The indices  $u$  and  $v$  are related to  $i$  and  $j$  and defined as  $u \triangleq \lfloor (40i + j)/110 \rfloor$  and  $v \triangleq (40i + j) \% 110$
  - The encoding of each block  $u$  on lane  $p$  is defined as follows:
    - A message polynomial  $m(x)$  of degree 109 is defined where the coefficient of the  $x^{109-v}$  term is `enc_input_p[110*u+v]` for  $v=0$  to 109
    - A generator polynomial  $g(x)$  of degree 16 is defined as  $g(x) = x^{16} + x^{14} + x^{11} + x^{10} + x^9 + x^7 + x^5 + x^3 + x + 1$
    - A parity polynomial  $p(x)$  of degree 15 is defined as the remainder from the division (module 2) of  $m(x) * x^{16}$  by the generator polynomial  $g(x)$ .  $p(x) = p_{15}x^{15} + p_{14}x^{14} + \dots + p_1x + p_0$
    - The bit mapping of the encoder output `enc_output_p` is defined as follows
      - $$enc\_output\_p[126u + v] = \begin{cases} enc\_input\_p[110u + v] & \text{for } v = 0 \text{ to } 109 \\ p_{125-v} & \text{for } v = 110 \text{ to } 125 \end{cases}$$
- In the receive direction, the BCH decoder function uses the signal received on `dec_input_0:31` lanes from the frame synchronization/pilot removal function and maps them to output bit lanes `dec_output_0:31`

# Circular Shift/Inverse Circular Shift Functions



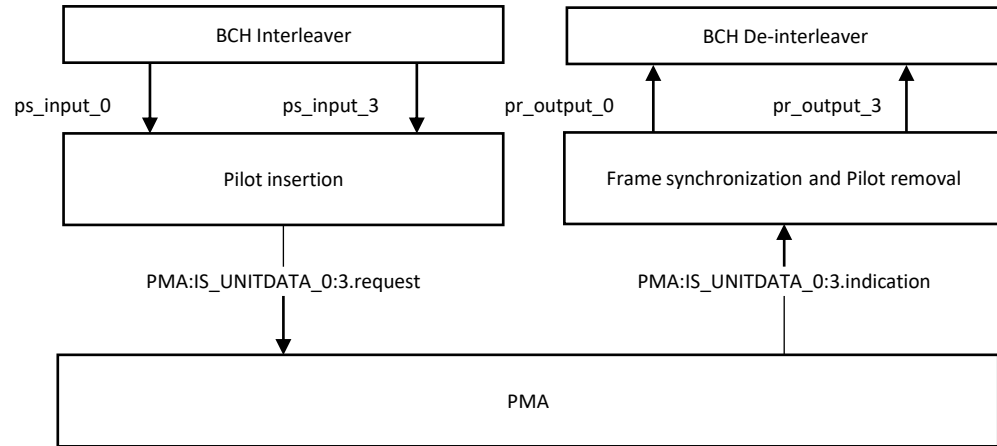
- Purpose: Works in conjunction with the BCH interleaver function to improve robustness to burst errors by spreading out the bursts over more RS(544,514) codewords
- In the transmit direction, the circular shift function maps the bits received on cs\_input\_0:31 lanes from the BCH encoder function and maps them to output lanes cs\_output\_0:31
  - Denote the input and output of the circular shift function of the  $p^{\text{th}}$  lane as  $cs\_input\_p$  and  $cs\_output\_p$  respectively ( $p=0$  to 31). The bits of  $cs\_input\_p$  are the bits at the output of the BCH encoder  $enc\_output\_p$ .
  - Denote the index of the 126-bit BCH codeword block aligned across all lanes as  $i$  and the bit index within block  $i$  as  $j$  ( $j=0$  to 125) where  $j=0$  to 109 are the information bits and bits  $j=110$  to 125 are the check bits inserted by the systematic BCH encoder.
  - The mapping between the bit sequences on the  $ci\_input\_p$  and the  $ci\_output\_p$  ( $p=0$  to 31) is:
    - $$ci\_output\_p[j] = \begin{cases} ci\_input\_p[(j - 20p)\%110] & \text{if } j < 110 \\ ci\_input\_p[j] & \text{otherwise} \end{cases}$$
- In the receive direction, the inverse circular shift function maps the signal received on ics\_input\_0:31 lanes from the frame BCH de-interleaver function and maps them to output lanes ics\_output\_0:31. The mapping of this function is the inverse of the circular shift

# BCH Interleaver/De-interleaver Function



- Purpose: Works in conjunction with the circular shift to improve robustness to burst errors and unequal bit error rates of the bits extracted from the DP-QAM16 symbols
  - The interleaver works by ensuring that the bits of each BCH codeword are spread evenly across all dimensions of the DP-QAM16 symbols after mapping in the PMA sublayer
  - The BCH interleaver works on a block of 32 synchronized BCH codewords generated on the 32 PCS lanes
- In the transmit direction, the BCH interleaver function maps the bits received on bchi\_input\_0:31 lanes from the circular shift function and maps them to output lanes bchi\_output\_0:3
  - Denote  $k$  as the index of the 126-bit codeword block aligned across all lanes  $p$  ( $p=0$  to  $31$ ) at the output of the circular shift blocks.
  - Denote  $q$  ( $q=0$  to  $3$ ) as the index of the output lane bchi\_output\_ $q$
  - Denote  $(2l+j)$  as an index ( $l=0$  to  $503$  and  $j=0$  to  $1$ ) of the bits within the  $k^{\text{th}}$  block of output bits formed from the  $k^{\text{th}}$  input bit block
  - The mapping between the bit sequences on bchi\_input\_ $p$  and bchi\_output\_ $q$  is:
 
$$bchi\_output\_q[2l + j] = bchi\_input\_p \left( 4 * \left\lfloor \frac{l}{63} \right\rfloor + \left( 2l + \left\lceil \frac{l}{2} \right\rceil \% 2 + q \right) \% 4 \right) [2 * l \% 63 + (l + j) \% 2]$$
- In the receive direction, the BCH de-interleaver performs the inverse function to map the information received on input lanes bdi\_input\_0:3 to the correct location within the output lanes bdi\_output\_0:31

# Pilot Insertion/Removal and Frame Synchronization Functions



- Purpose: Pilot symbols are implemented as the outer symbols of the DP-16QAM constellation, allowing robust framing to a DP-QPSK constellation
  - Pilots are inserted every 64 symbols (one pilot symbol, 63 payload symbols.)
  - 63 payload symbols is an easy multiple for BCH(126,110), no RES (padding) needed
  - A DSP Frame is  $96 \times 64 = 6144$  symbols including 96 pilots.
  - The pilot sequence is a PRBS9 pattern initialized at the beginning of the DSP Frame.
  - Seed value for pilot reset is chosen to ensure DC balance.

- In the transmit direction, the pilot inserter function maps the bits received on  $ps\_input\_0:3$  lanes from the BCH interleaver function and maps them to output lanes  $ps\_output\_0:3$  and provides these output bits through the PMA:IS\_UNITDATA\_0:3.request

- Denote  $u$  as the index of the 126-bit codeword block at the input of the pilot inserter function aligned across all lanes  $q$  ( $q=0$  to  $3$ ) and aligned to the boundaries of the BCH encode function. The index  $u$  is identical to the index  $u$  used in the definition of the BCH encode function.

- The pilot sequence  $ps[q,k]$  for lane  $q$  ( $q=0$  to  $3$ ) and index  $k$  ( $k=0$  to  $95$ ) is defined as the output of a PRBS9 generator. The sequences are zero-balanced in each lane and have low correlation across lanes.

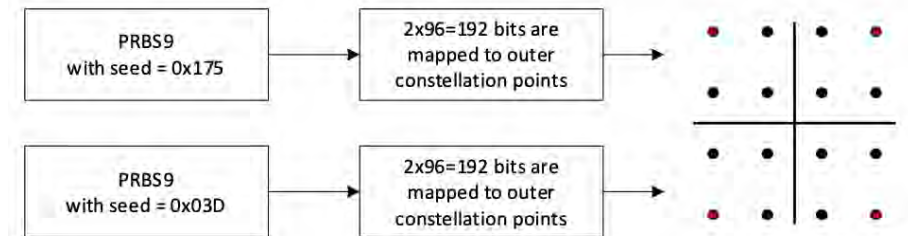
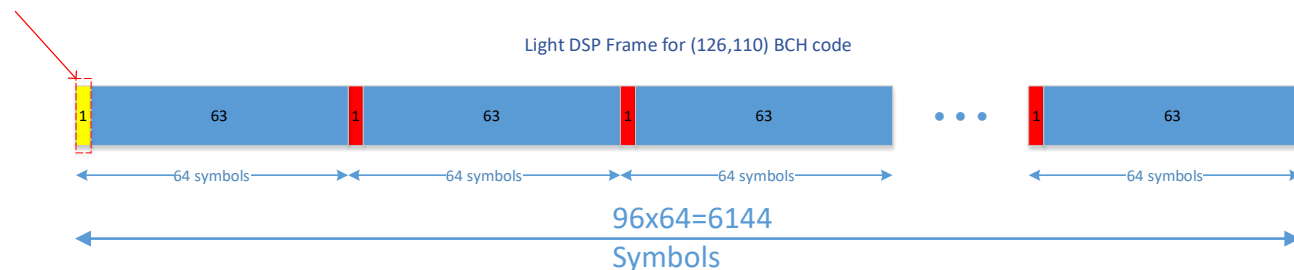
- The mapping between the bit sequences on the  $ps\_input\_q$  and the  $ps\_output\_q$  ( $q=0$  to  $3$ ) is:

$$ps\_output\_q[128u + v] = \begin{cases} ps[q, \lfloor u/96 \rfloor] & v = 0 \\ 0 & v = 1 \\ ps\_input\_q[126u + v - 2] & v = 2 \text{ to } 128 \end{cases}$$

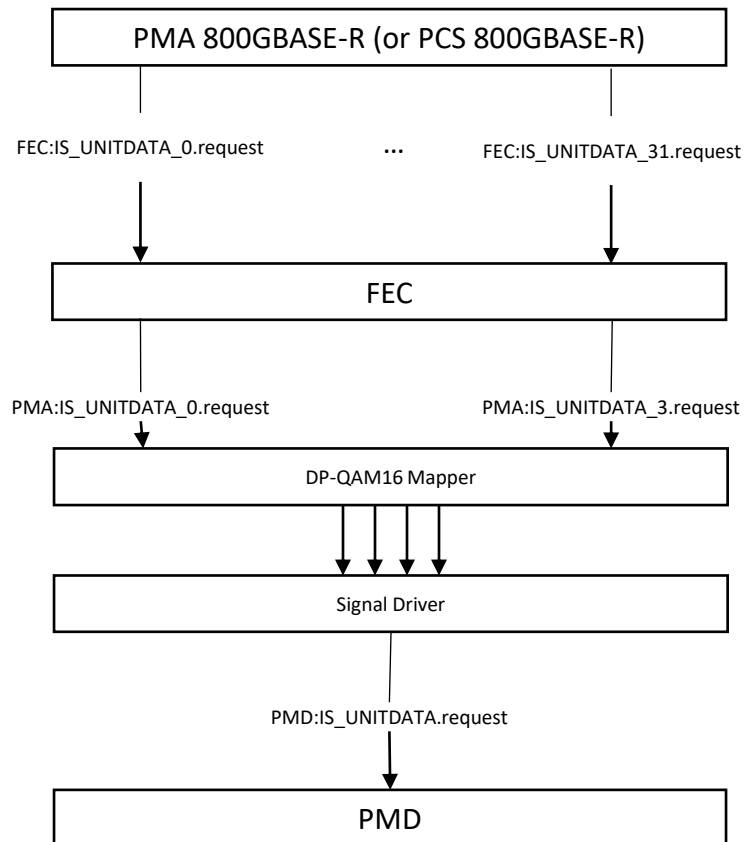
- In the receive direction, the pilot sequence bits are used to synchronize the frame to determine the BCH decoder boundaries and initialize the convolutional de-interleavers. These symbols are removed prior to BCH de-interleaver function.

## Pictorial Description of pilots within the DP-QAM16 symbols

Reset Pilot Sequence



# PMA Sublayer Transmit Path

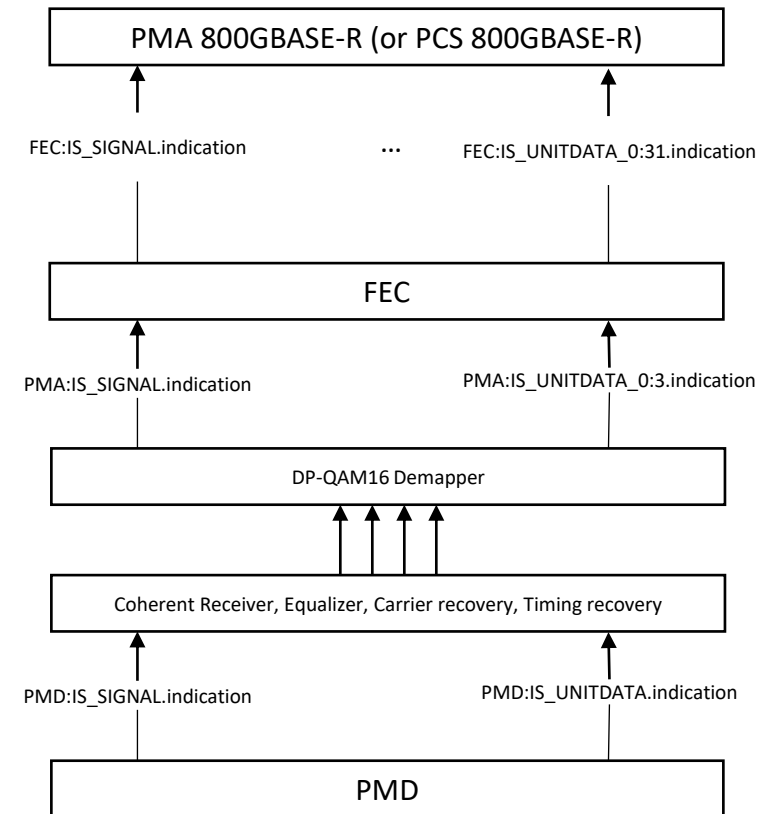


- Encode 4 FEC output lanes into 1 optical signal lane
- DP-QAM16 (Dual-polarization QAM16) mapper is used to map the four bit lanes received from the FEC sublayer through the PMA:IS\_UNITDATA.request primitives into four lanes of 4-level signals used to modulate the XI/XQ/YI/YQ of optical signal
- Signal driver function includes all the electro-optical processing required to convert the mapper output into an optical signal
- Details of the mapper function is provided in subsequent slide. Optical specifications required to implement the signal driver function will be covered in subsequent contributions.

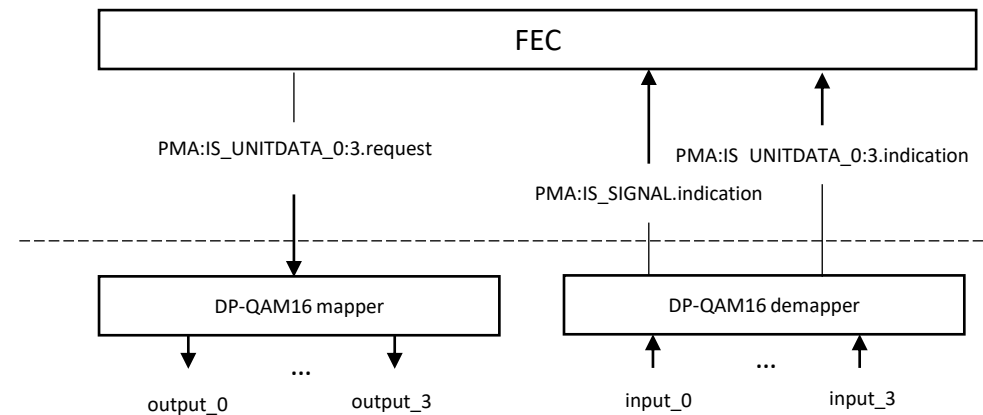


# PMA Sublayer Receive Path

- Decode the optical signal received at the PMD into 4 output lanes provided through the PMA:IS\_SIGNAL.indication and PMA:IS\_UNITDATA\_0:3.indication primitives
- The coherent receiver implements all functions required to convert the optical signal into into four lanes of 4-level signals
- The DP-QAM16 demapper function converts four lanes of 4-level signals into four bit sequences provided to the FEC sublayer through the PMA:IS\_UNITDATA\_0:3.indication primitives
- Details of the demapper function is provided in subsequent slide. Optical specifications required to implement the coherent receiver will be covered in subsequent contributions.



# DP-QAM16 mapper/demapper Functions



- Purpose: Convert the bit sequences at the PMA service interface to and from four lanes of four-level signals which modulate the XI/XQ/YI/YQ optical signal at the PMD
- Prerequisites: Maintain 2-bit alignment to the functions in the FEC sublayer
- In the transmit direction, the DP-QAM16 mapper function maps the bits received on four PMA lanes through the PMA:IS\_UNITDATA\_0:3.request primitives and maps them to output lanes *output\_0:3*
  - Denote the index of the PMAL as  $p$  ( $p=0$  to  $3$ ), where the bits for the  $p^{\text{th}}$  lane are obtained through PMA:IS\_UNITDATA\_ $p$ .request
  - Denote the index of the aligned input 2-bit block across the PCSs as  $i$  and the bit index within block  $i$  as  $j$  ( $j=0$  to  $1$ ). The 2-bit boundaries are maintained through the processing in the FEC sublayer.
  - The mapping between the bit sequences on the PMAL and the output\_ $p$  is:
    - $output\_p[i] = 4 * (PMAL[p, 2i]) + 2 * (PMAL[p, 2i] \wedge PMAL[p, 2i + 1]) - 3$
- In the receive direction, the DP-QAM16 demapper function maps the four signals extracted from the XI/XQ/YI/YQ of the optical signal on input lanes *input\_0:3* to four bit lanes and provides this data through the PMA:IS\_UNITDATA\_0:3.indication primitives. The demapper implements the inverse of the DP-QAM16 mapper function.

# Summary

- This proposal provides a complete logic baseline (FEC and PMA) for single-wavelength 800G coherent interfaces using an inner FEC
- The details were provided previously in [maniloff 3dj 01a 2305](#). The description has been updated to follow the document structure and conventions in this task force (for e.g. [ran 3dj 01a 2303](#), [nicholl 3dj 01a 2305](#))
- The proposal provides a low complexity, low power, low latency solution for coherent interfaces without sacrificing performance
- The proposal leverages many aspects of similar proposals for the IMDD objectives in this task force (see [patra 3dj 01b 2303](#))