ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

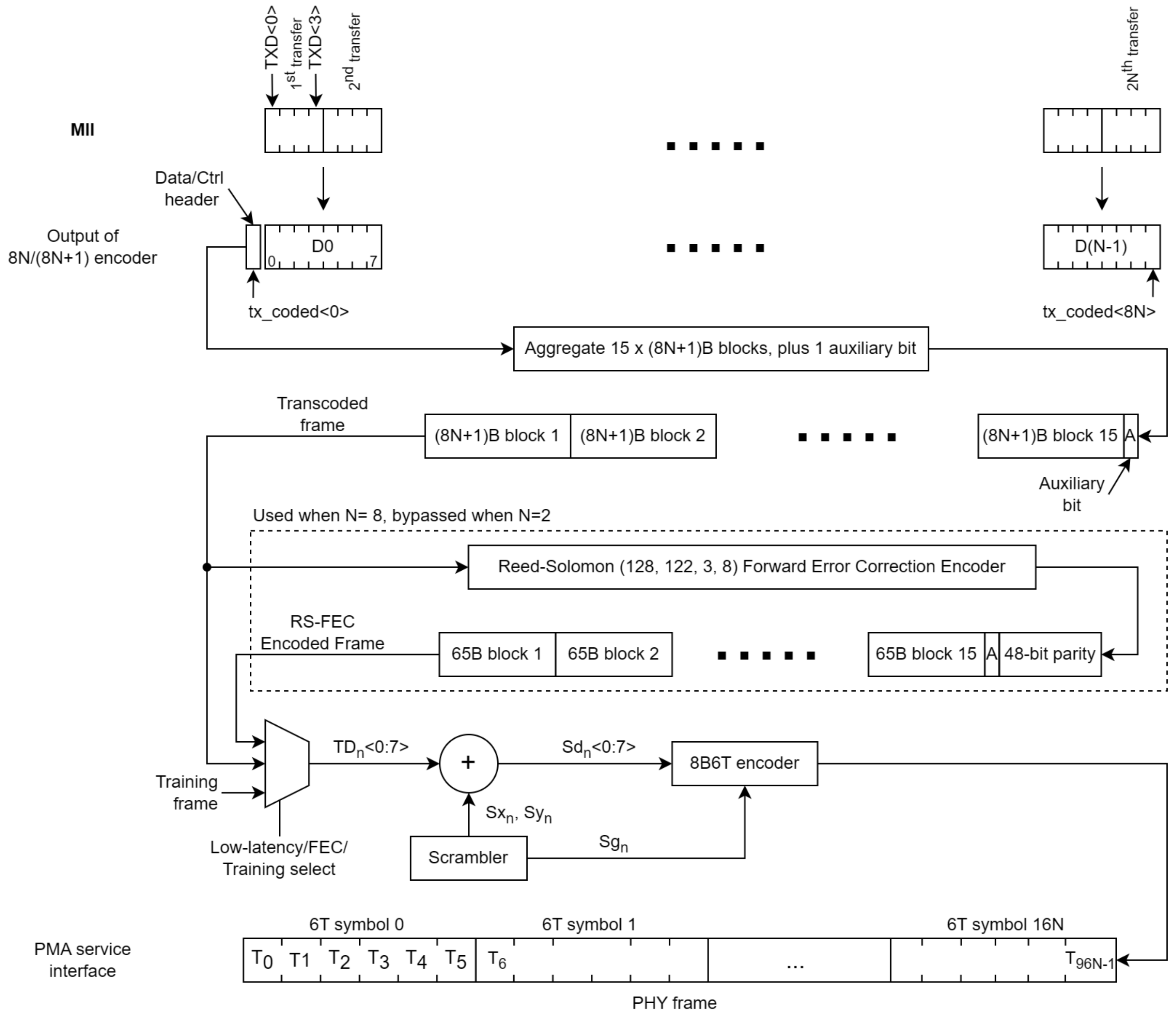
Proposal for a 100BASE-T1L PCS

Jacobo Riesco
Brian Murray

Introduction

- This a proposal for the 100BASE-T1L PCS
  - At the July meeting the 802.3dg group approved a motion to adopt PAM3 modulation at 80 Msymbol/s using 8b6T coding in conjunction with an 8N/8N+1 block code and with a low latency mode and a burst error protection mode
  - See Murray_3dg_01a_07152024
  - This a proposal for the PCS to be used with this PHY

PCS TX Block Diagram and Bit Ordering



N=2: 16B/17B low latency

N=8: 64B/65B with RS FEC

8b6T Coding and Block Codes

- PAM-3 8b6T coding using 8N/(8N+1) block codes has been proposed for 100BASE-T1L with two different modes supported:
  - Low latency mode using a 16B/17B (N=2) block code
  - Burst error protection mode using a 64B/65B (N=8) block code and RS FEC
    - Line rate is 80 Msymbol/s in both cases
  - 8b6T coding is described in:
    - https://grouper.ieee.org/groups/802/3/dg/public/May_2024/Curran_3dg_01a_07152024.pdf
    - https://grouper.ieee.org/groups/802/3/dg/public/May_2024/Murray_3dg_01a_07152024.pdf
  - 8N/(8N+1) encoding is described in:
    - https://www.ieee802.org/3/dg/public/May_2022/Lo_3dg_01_012524.pdf
    - https://grouper.ieee.org/groups/802/3/dg/public/May_2024/Lo_3dg_01a_0724.pdf
  - RS FEC is described in
    - https://grouper.ieee.org/groups/802/3/dg/public/May_2024/Tingting_3dg_02_16_07_2024.pdf

8N/(8N+1) Encoding

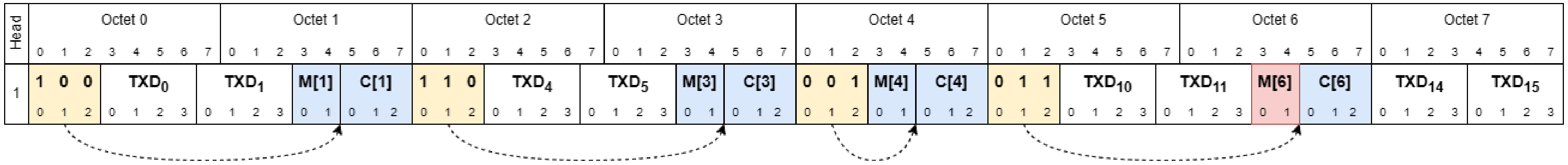
- Defined by the following pseudo-code (Modifications from Clause 97, per Lo_3dg_01a_0724), where N is the number of octets encoded in a block
  - N = 8 when the Reed-Solomon FEC is used and N = 2 when it is not used
  - Octets within a block are numbered using and increasing index n, from 0 to N-1, with n = 0 being the first octet of the block presented on the MII interface.

```
TC[n]           : 0 if octet n is a data octet, i.e., if the octet n contains two MII data nibbles, TXD[2n][0:3] and  TXD[2n+1][0:3]; 1 otherwise.
TC[-1]          : 1 by definition
TD[n][0:7]      : MII octet n (TD[n][0:3] = TXD[2n][0:3], TD[n][4:7] = TXD[2n+1][0:3]) if TC[n] = 0
C[n][0:2]       : MII control code n, corresponding to MII data nibbles 2n, 2n+1 (and 2n+2) as per Table 1 above.
B[0:8N]         :  8N+1 block. Bit 0 transmitted first
OR(n)           : Bitwise OR of TC[n:N-1]
OR(N)           : 0 by definition
NEXT(n)[0:2]    : bit position of lowest bit in TC[n:N-1] that is a 1. Bit 3 is MSB.
M(n)[0:1]       : MII mode n, corresponding to MII data nibbles 2n, 2n+1 (and 2n+2).
                  M[n][0] = 1 if encoded symbol is CD; else 0.
                  M[n][1] = TXD2n[0] if encoded symbol is CD; else OR(n+1)

B[0]            = OR(0)
B[8n+1:8n+3]    = TD[n][0:2]      if OR(n) = 0
                  NEXT(n)[0:2]    if OR(n) = 1 AND TC[n-1] = 1
                  TD[n-1][5:7]    if OR(n) = 1 AND TC[n-1] = 0
B[8n+4:8n+5]    = TD[n][3:4]      if OR(n) = 0
                  M[n][0:1]       if OR(n) = 1 AND TC[n] = 1
                  TD[n][0:1]      if OR(n) = 1 AND TC[n] = 0
B[8n+6:8n+8]    = TD[n][5:7]      if OR(n) = 0
                  C(n)[0:2]       if OR(n) = 1 AND TC[n] = 1
                  TD[n][2:4]      if OR(n) = 1 AND TC[n] = 0
```

8N/(8N+1) Encoding

- Example encoding for a N = 8 block containing the following MII octets $D_0/C_1/D_2/C_3/C_4/D_5/C_6/D_7$
  - Octets are shown in the transmit order, and bits within an octet and its sub-fields are also shown in the transmit order, i.e. LSB first



Scrambler

- Data and training modes use the same 33 bits side-stream scrambler approach used in many other previous PHY technologies, Clauses 40, 146, 96, 149, 165
- Generator polynomials for Master and Slave are:
  - Master: $g_M(x) = 1 + x^{13} + x^{33}$
  - Slave: $g_S(x) = 1 + x^{20} + x^{33}$

  NOTE— IEEE Std 802.3, Annex K defines optional alternative terminology for “master” and “slave”
- Scrambling is done per 8b/6T octet
  - The bits stored in the scrambler shift register delay line at time n are denoted by $Scr_n[32:0]$
  - At each octet period, the shift register is advanced by one bit, and a new bit represented by $Scr_n[0]$ is generated
  - Encoding rules are based on the generation, at time n, of nine bits: $Sx_n[3:0]$, $Sy_n[3:0]$ and Sg_n
    - The eight $Sx_n[3:0]$ and $Sy_n[3:0]$ bits are used to decorrelate the octet $TD_n[7:0]$ during transmission
    - Sg_n is used to randomize the sign of the NND 6T symbol in the 8b6T encoder

Scrambler. Generation of $Sd_n[7:0]$ and Sg_n

- $Sx_n[3:0]$, $Sy_n[3:0]$ and Sg_n are generated using three uncorrelated bits, X_n, Y_n and $Scr_n[0]$, and an auxiliary generator polynomial, g(x), as in Clause 40:
  - The bits X_n and Y_n derived from elements of the same maximum-length shift register sequence of length $2^{33}-1$ as $Scr_n[0]$, but shifted in time
  - **The associated delays are all large and different so that there is no short-term correlation among the bits X_n, Y_n and $Scr_n[0]$**
  - They are generated as follows:

$$X_n = Scr_n[4] \wedge Scr_n[6]$$
$$Y_n = Scr_n[1] \wedge Scr_n[5]$$

  - The generator polynomial is:

$$g(x) = x^3 \wedge x^8$$

  - $Sx_n[3:0]$, $Sy_n[3:0]$ and Sg_n are generated as follows:

$$Sy_n[0] = Scr_n[0]$$
$$Sy_n[1] = g(Scr_n[0]) = Scr_n[3] \wedge Scr_n[8]$$
$$Sy_n[2] = g^2(Scr_n[0]) = Scr_n[6] \wedge Scr_n[16]$$
$$Sy_n[3] = g^3(Scr_n[0]) = Scr_n[9] \wedge Scr_n[14] \wedge Scr_n[19] \wedge Scr_n[24]$$
$$Sg_n = Y_n = Scr_n[1] \wedge Scr_n[5]$$

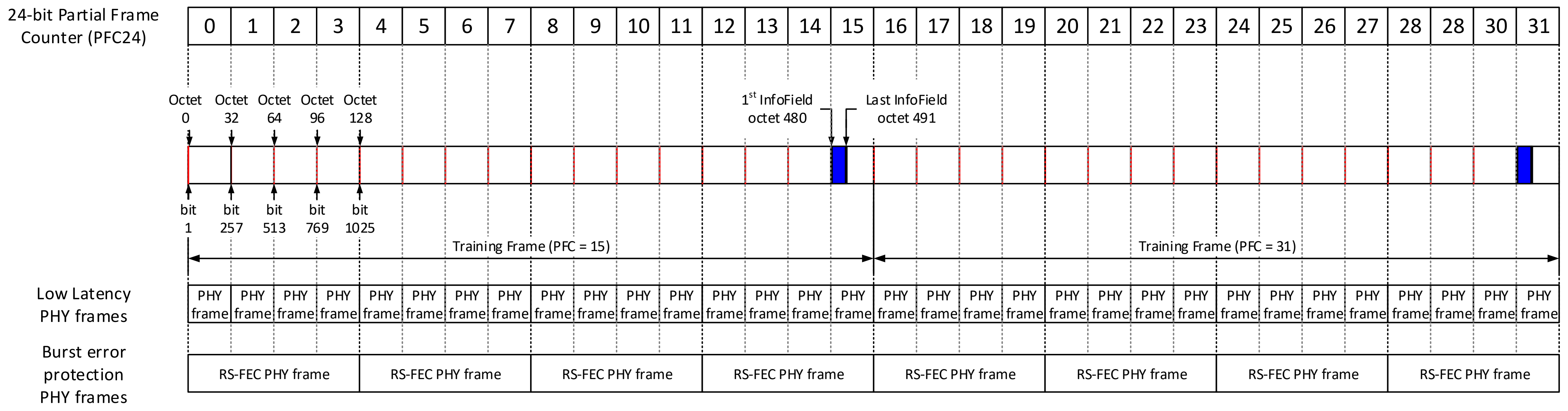
$$Sx_n[0] = X_n = Scr_n[4] \wedge Scr_n[6]$$
$$Sx_n[1] = g(X_n) = Scr_n[7] \wedge Scr_n[9] \wedge Scr_n[12] \wedge Scr_n[14]$$
$$Sx_n[2] = g^2(X_n) = Scr_n[10] \wedge Scr_n[12] \wedge Scr_n[20] \wedge Scr_n[22]$$
$$Sx_n[3] = g^3(X_n) = Scr_n[13] \wedge Scr_n[15] \wedge Scr_n[18] \wedge Scr_n[20] \wedge Scr_n[23] \wedge Scr_n[25] \wedge Scr_n[28] \wedge Scr_n[30]$$

- The scrambled data octet, $Sd_n[7:0]$ is generated as follows:

$$Sd_n[7:4] = Sx_n[7:4] \wedge TD_n[7:4]$$
$$Sd_n[3:0] = Sy_n[3:0] \wedge TD_n[3:0]$$

PMA Training Frame

- The PMA training frame is the same approach used in Clauses 149 &165
- Training frames with indicators are sent during PMA training to establish the alignment of the PHY frames



- Each partial PHY frame is 32 octets (256 bits) long, beginning at Sd_n where (n mod 32) = 0
- Each training frame is composed of 16 partial PHY frame

PMA Training Frame: Scrambling

- All the bits in the training frame except the second bit in each partial frame (in red in the previous figure), which is set to 1, and the InfoField (in blue in the previous figure) are zero. Therefore:
  - The second bit is inverted in the first 15 partial frames
    - $Scr_n[0]$ is sent on the first bit to facilitate scrambler acquisition at the receiver
  - The first 12 octets (96 bits) of the 16th partial frame are XORed with the contents of the InfoField[0:95]
  - The scrambled octets during training are:

$$m = 8 \cdot (n \, mod \, 32)$$

$$Sd_n[7:4] = \begin{cases} Sx_n[7:4] \wedge InfoField[m+7:m+4] & 480 \leq (n \, mod \, 512) \leq 491 \\ Sx_n[7:4] & otherwise \end{cases}$$

$$Sd_n[3:2] = \begin{cases} Sy_n[3:2] \wedge InfoField[m+3:m+2] & 480 \leq (n \, mod \, 512) \leq 491 \\ Sy_n[3:2] & otherwise \end{cases}$$

$$Sd_n[1] = \begin{cases} Sy_n[1] \wedge InfoField[m+1] & 480 \leq (n \, mod \, 512) \leq 491 \\ Sy_n[1] \wedge 1 & else \, if \; m = 1 \\ Sy_n[1] & otherwise \end{cases}$$

$$Sd_n[0] = \begin{cases} Sy_n[0] \wedge InfoField[m] & 480 \leq (n \, mod \, 512) \leq 491 \\ Sy_n[0] = Scr_n[0] & otherwise \end{cases}$$

- The scrambled training octet $Sd_n[7:0]$ is 8b/6T encoded as in normal operation, so that ternary symbols (PAM3) with bounded running disparity are also used during training

PMA Training Frame: PHY Control InfoField

▶ PHY control information is exchanged between link partners during training using a 12-octet InfoField (IF). Same approach as in Clauses 97, 149 and 165

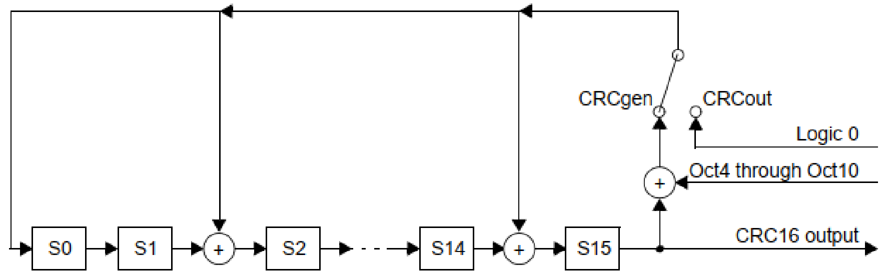
Octet 1	Octet 2	Octet 3	Octet 4	Octet 5	Octet 6	Octet 7	Octet 8	Octet 9	Octet 10	Octet 11	Octet 12
0xBB	0x7A	0x00		PFC24		Message		PHY capability / DataSwPFC24		CRC16	

PMA_state		loc_rcvr_status	en_slave_tx (master) timing_lock_ok (slave)	reserved			
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	1	1	1	0	0	0	0

Octet 8								Octet 9								Octet 10							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
VendorSpecificData																reserved	EEEen	LPIen	reserved				RSen

PHY capability (*Exact contents may change)

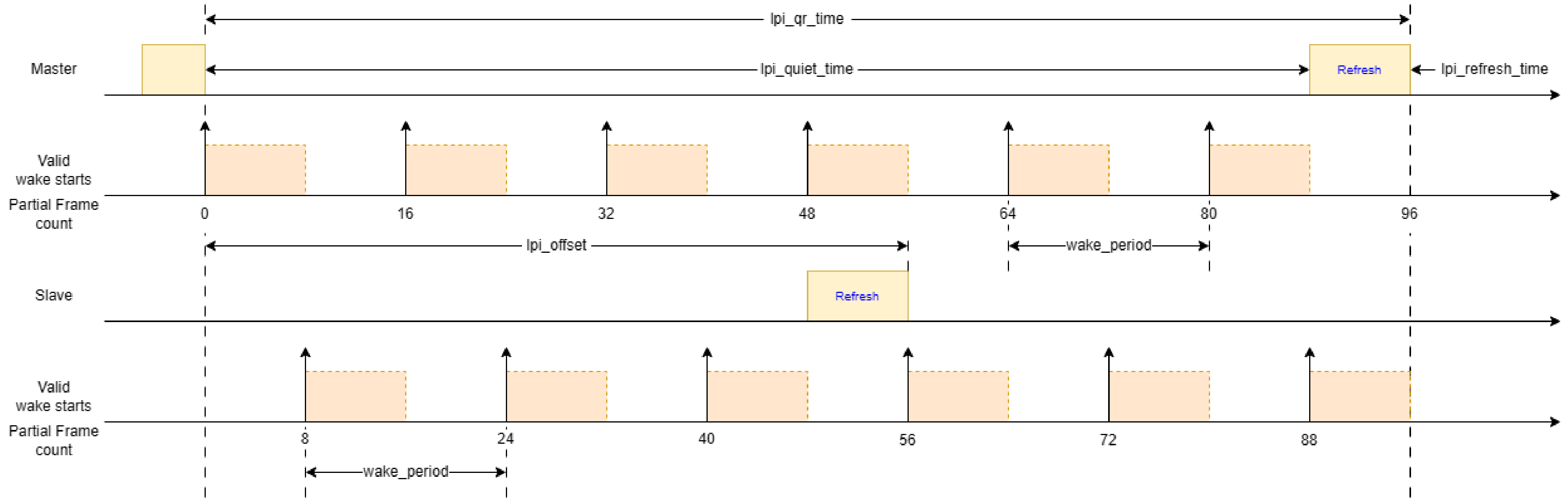
- Octets and Fields are sent LSB first
- RSen selects burst error protection mode (RS-FEC) when set in both link partners



CRC16 same as Clauses 97, 149 and 165

Optional EEE Capability

- EEE uses the same approach as in Clauses 97, 149 &165
- EEE abilities advertised in InfoField (EEEen and LPIen bits in PHY Capability)
  - LPI only entered when both link partners indicate EEEen = 1 and LPIen = 1
  - If EEEen = 1 and LPIen = 0, *PCS encoding transparently carries the LPI signalling MII to MII*, but the PHY does not enter LPI quiet-refresh cycle
  - If EEEen = 0, LPI is encoded as a normal inter frame (Idle), and decoded as error
- LPI consists of alternating quiet and refresh periods
- Each direction of the link can enter and exit LPI mode independently



LPI Synchronization and Signalling

- LPI synchronization
  - Quiet-refresh cycle stablished from the master partial frame count (PFC24)
    - Slave PHY shall synchronize its PFC24 to the master's during training

Variable	Master	Slave
tx_refresh_active	lpi_quiet_time ≤ mod(PFC24, lpi_qr_time)	lpi_offset-lpi_refresh_time ≤ mod(PFC24, lpi_qr_time) < lpi_offset
tx_wake_start	mod(PFC24, wake_period) = 0	mod(PFC24, wake_period) = wake_period/2

- LPI signalling
  - During quiet periods, the PCS transmitter passes zero ternary symbols to the PMA
  - During the staggered out of phase refresh periods, the PCS transmitter operates as in normal mode, with PCS transmit data (tx_coded) set to zero (or encoded LPI)
    - No PHY health information. Quiet refresh cycle shall be sufficient to ensure PHY SNR
  - During wake-up, the PCS transmitter operates as in normal mode, with the PCS transmit data (tx_coded) containing (8N/8N+1) encoded normal inter-frame symbols
    - No alert signal (same as clause 97)

LPI Timing Parameters and Wake-up Time

Parameter	Number of partial frame periods (*Values may change)	µs
lpi_offset	56	
lpi_qr_time	96	230.4
lpi_quiet_time	88	211.2
lpi_refresh_time	8	19.2
sleep	8	19.2
wake_period	16	

- Quiet time determined by the requirement to preserve timing and keep the eye open
- Refresh allows enough time to update the filter coefficients to maintain the link quality

lpi_wake_time	Wake starts before sleep is complete		Wake starts after sleep is complete	
Partial frames	Partial frames	µs	Partial frames	µs
8	32	76.8	24	57.6

OAM

- OAM is not supported
  - No side-channel communications
  - Only 1 auxiliary bit is available in the proposed encoding for the low latency mode and the burst error protection mode
- This bit will be used as a beacon set to 1 when the link is up
  - Additional information (TBD) might be passed if required, but no full OAM allowing the transport of vendor specific messages between PCSs

Sequence Ordered Sets

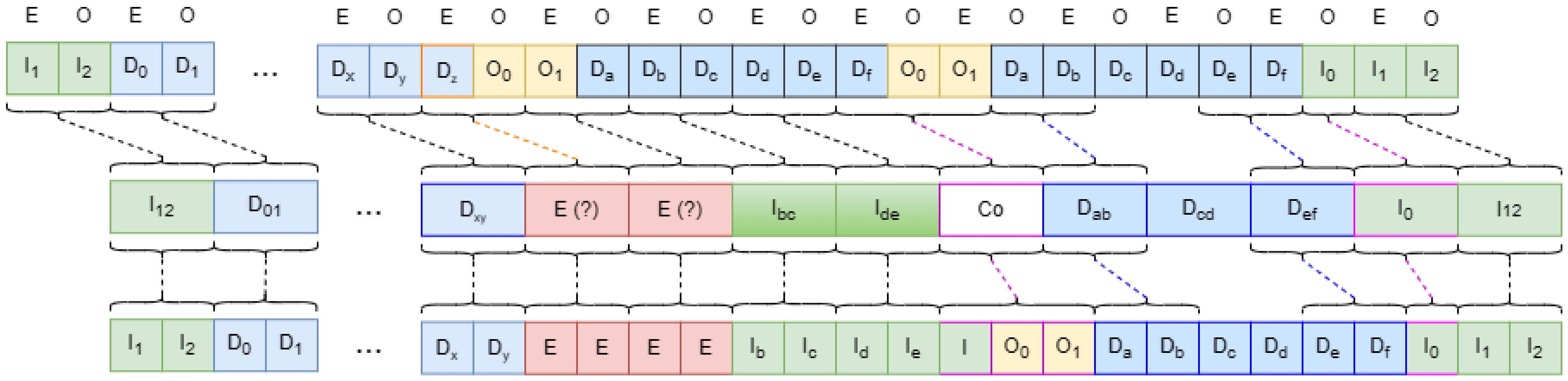
- Adding sequence ordered sets to MII and the 8N/(8N+1) encoding has been adopted
    - https://grouper.ieee.org/groups/802/3/dg/public/May_2024/Lo_3dg_01a_0724.pdf
- Defined in XGMII and higher speed MIIs, but never retrofitted into MII *nor GMII*
    - XGMII: uses a control character in lane 0 and data characters in lanes 1 to 3
    - MII: could use two ordered set nibbles (TX_EN=0, TX_ER=1, TXD=0x4) followed by 6 normal-inter packet nibbles (TX_EN=TX_ER=0) with TXD set to the ordered set code
- Could be used for link fault signalling between local and remote Reconciliation Sublayers (RS)
    - Three sequence ordered set codes (out of a possible 2^{24}) are defined:
        - ***Local Fault:*** *Signalled* by local PHY to local RS
            - Local RS stops transmit data/LPI from local MAC and generates remote fault on transmit path (*possibly truncating a MAC frame*)
        - ***Remote Fault:*** *Signalled* by local RS to remote RS
            - Remote RS stops transmit data/LPI from remote MAC and generates inter-frame indications (idles)
        - ***Link Interruption* (optional):** Signalled by local PHY to local RS
            - Local RS *May* set CARRIER_STATUS to CARRIER_ON
- But, as shown in Murray_3dg_01_09162024, none of the existing 100M silicon or commonly used MAC interfaces support Local / Remote Fault / Link Interruption or sequence ordered sets
    - Remote faults cannot be conveyed reliably to link partner when the link degrades

Issue with Previously Adopted Proposal for 8N/8N+1

- The previously adopted proposal to achieve constant latency groups 3 nibbles at the encoder if a packet or a sequence ordered set starts on an odd cycle, and then ungroups at the decoder
- However, there is an issue with the previously adopted proposal in some cases
  - When a sequence ordered set follows a packet and starts on an odd cycle there is a problem since there are not enough control symbols available to encode it
  - A sequence ordered set may start on an odd cycle
    - If a packet with and odd number of nibbles is followed by a sequence ordered set
    - If a packet is interrupted by a sequence on an odd cycle
  - Lo_3dg_01a_0724 does not define how to handle these cases
- The following slide explains this problem in more detail

Issue with Previously Adopted Proposal for 8N/8N+1

- Packet with an odd number of nibbles followed by a sequence
  - This is a real use case, not a corner case
- In MII, the transmit clock, TX_CLK, is sourced from the PHY to the MAC
  - In general, TX_CLK will be plesiochronous to the MAC clock
    - The RS will need to adapt between the MAC and PHY transmit clock rates
    - Usually this is done by simply adding or deleting Idle *nibbles*
    - But, in addition, most MACs will enforce the minimum IPG at the transmit side
  - For packets separated by the minimum IPG, sometimes, **the first preamble nibble is replaced with an idle nibble, resulting in a valid packet with an odd number of nibbles**



E O E O
I₁ I₂ D₀ D₁ …
E O E O E O E O E O E O E O E O E O E O E O
Dₓ D_y D_z O₀ O₁ D_a D_b D_c D_d D_e D_f O₀ O₁ D_a D_b D_c D_d D_e D_f I₀ I₁ I₂
I₁₂ D₀₁ …
D_xy E (?) E (?) I_bc I_de Co D_ab D_cd D_ef I₀ I12
I₁ I₂ D₀ D₁ …
Dₓ D_y E E E E I_b I_c I_d I_e I O₀ O₁ D_a D_b D_c D_d D_e D_f I₀ I₁ I₂

Proposed Constant Latency MII to 8N/8N+1 Encoding

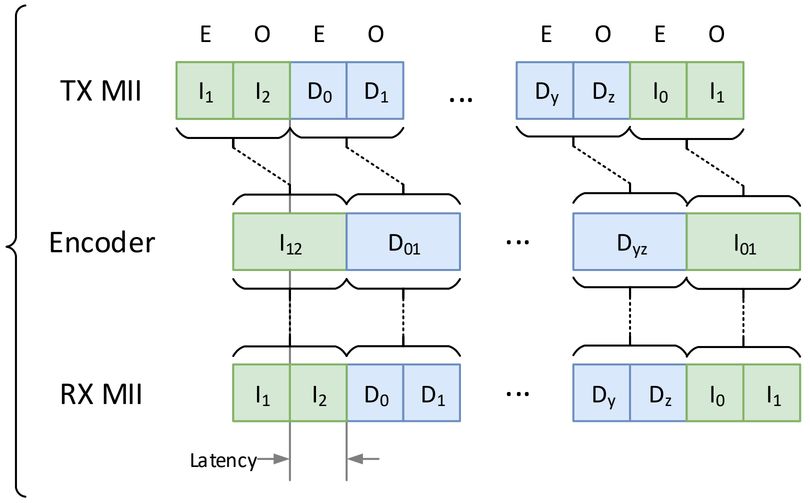
- This issue is addressed by simplifying the scheme for constant latency
- We propose the following for the constant latency MII to 8N/8N+1 encoding
  - Do not include sequence ordered sets in the PCS 8N/8N+1 encoding
    - Precludes end to end communication of sequence ordered sets
    - This does not preclude using sequence ordered sets between the PHY and the RS
  - Do not group/ungroup 3 nibbles at the start and do not expand/compress one idle at the end of the packet
    - This avoids an increase of the transmit latency by one MII clock cycle (40 ns)
    - This results in constant **lower** latency for packets
    - We can use state-less encoding which is simpler
  - The Codes for sequence ordered sets starting on an odd cycle (Co) are not needed
  - Do not have to deal with unexpected behaviour that may occur in case of errors
  - Encoder and decoder do not need to remain synchronized
- Simpler and more robust state-less encoding
  - Always decodes an octet into two nibbles
  - As it is state-less dealing with the case of errors is simpler

Start on Even Cycle, Packet Even Number of Nibbles

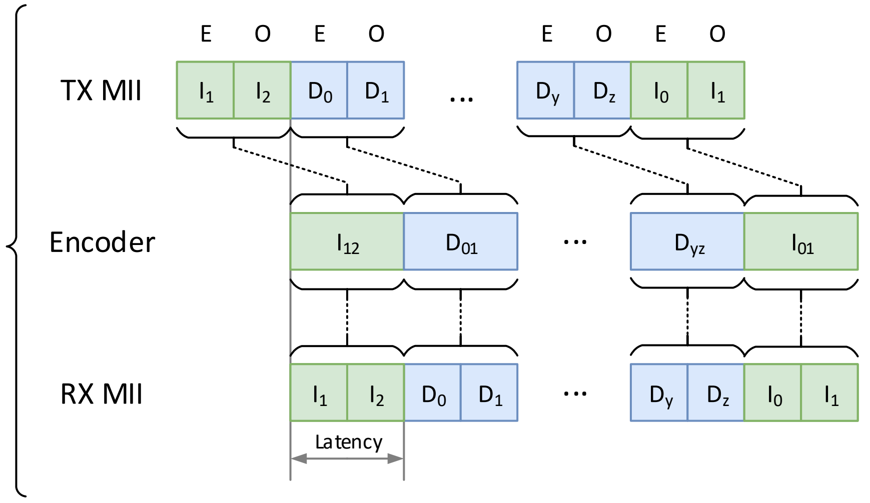
- Encoding: combine even + odd nibbles into octet presented to encoder
- Decoding: split octet to nibbles

➢ Only difference wrt previously adopted proposal is latency

New proposal



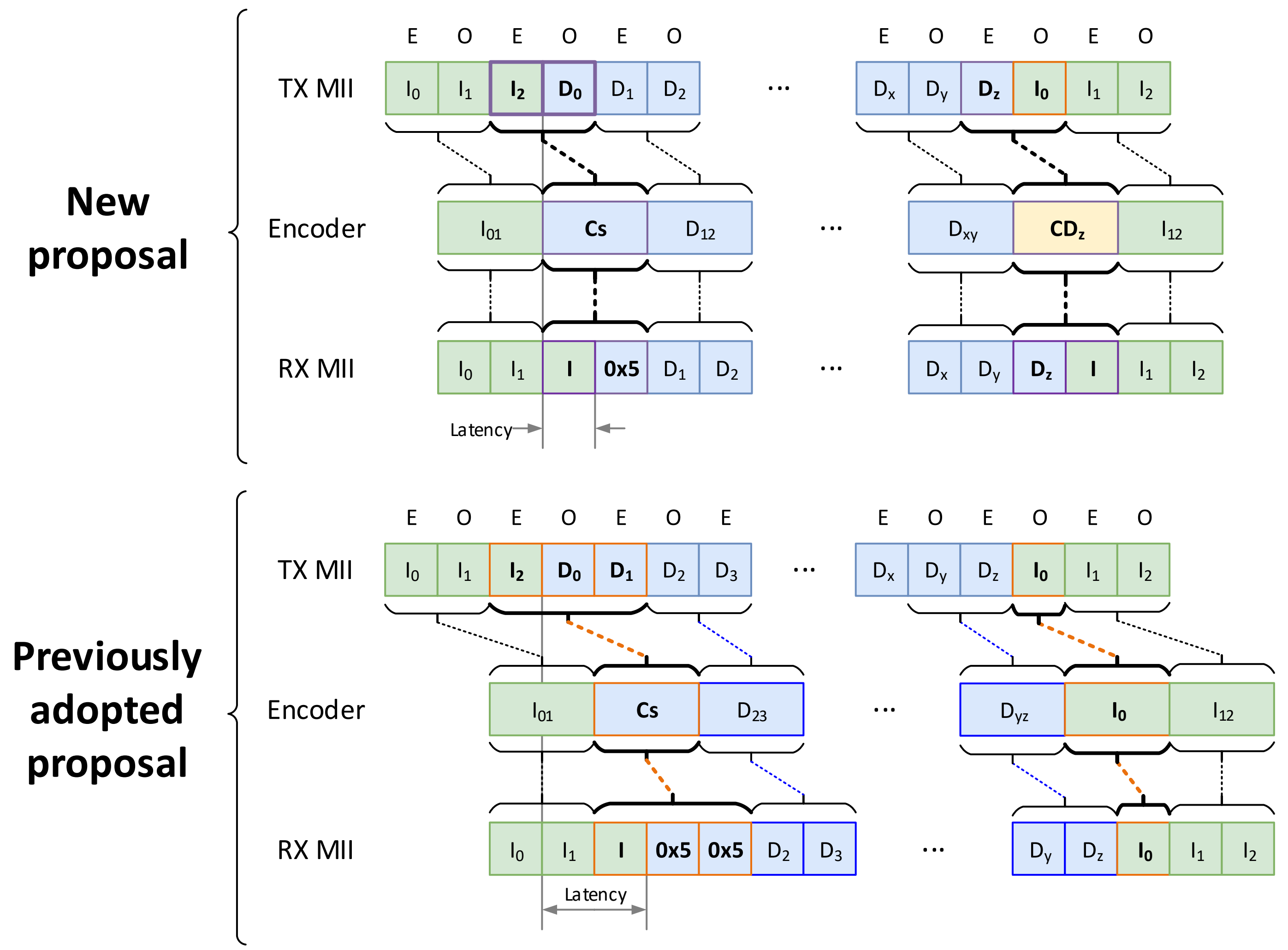
Previously adopted proposal



Start on Odd Cycle, Packet Even Number of Nibbles

- Encoding: combine even + odd nibbles into octet presented to encoder
  - Idle nibble followed by first preamble data nibble in octet, encoded to Cs
  - Data nibble (D_z) followed by idle nibble in octet, encoded to CD_z
- Decoding: Split octet to nibbles
  - Cs decoded to idle nibble followed by 0x5 data nibble
  - CD_z decoded to D_z data nibble followed by idle nibble

➢ Only difference wrt previously adopted proposal is latency and we don't need to group/ungroup 3 nibbles or expand/compress one idle

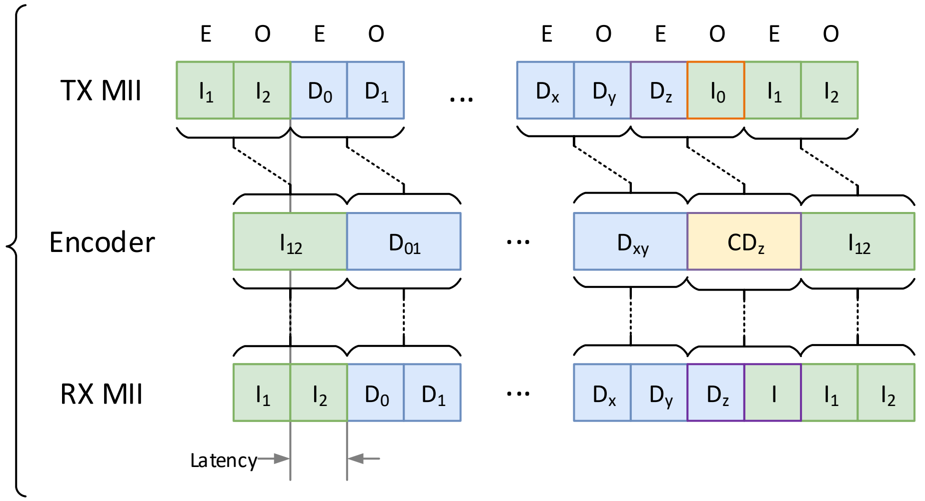

New proposal
E O E O E O
TX MII
I0 I1 I2 D0 D1 D2 … Dx Dy Dz I0 I1 I2
Encoder
I01 Cs D12 … Dxy CDz I12
RX MII
I0 I1 I 0x5 D1 D2 … Dx Dy Dz I I1 I2
Latency
Previously adopted proposal
E O E O E O E
TX MII
I0 I1 I2 D0 D1 D2 D3 … Dx Dy Dz I0 I1 I2
Encoder
I01 Cs D23 … Dyz I0 I12
RX MII
I0 I1 I 0x5 0x5 D2 D3 … Dy Dz I0 I1 I2
Latency

Start on Even Cycle, Packet Odd Number of Nibbles

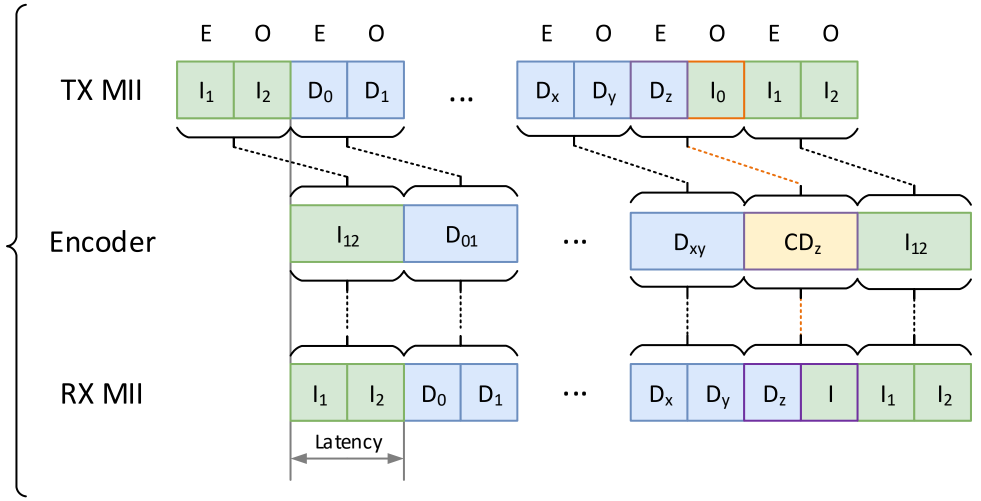
- Encoding: combine even + odd nibbles into octet presented to encoder
  - Data nibble (D_z) followed by Idle nibble in octet, encoded to CD_z
- Decoding: Split octet to nibbles
  - CD_z decoded to D_z data nibble followed by Idle nibble

- Only difference wrt previously adopted proposal is latency

New proposal



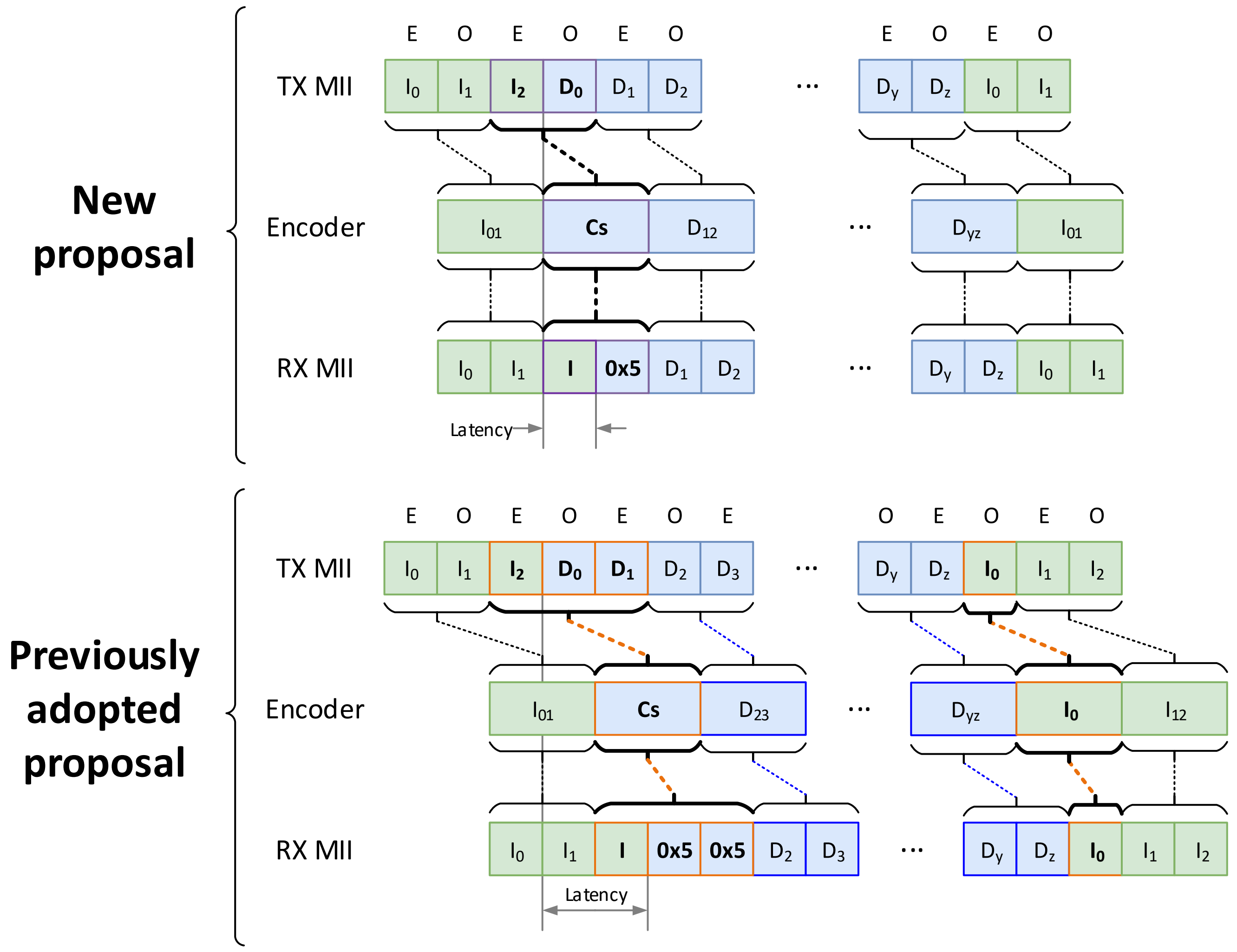
Previously adopted proposal



Start on Odd Cycle, Packet Odd Number of Nibbles

- Encoding: combine even + odd nibbles into octet presented to encoder
  - Idle nibble followed by first preamble data nibble in octet, encoded to Cs
- Decoding: Split octet to nibbles
  - Cs decoded to idle nibble followed by 0x5 data nibble

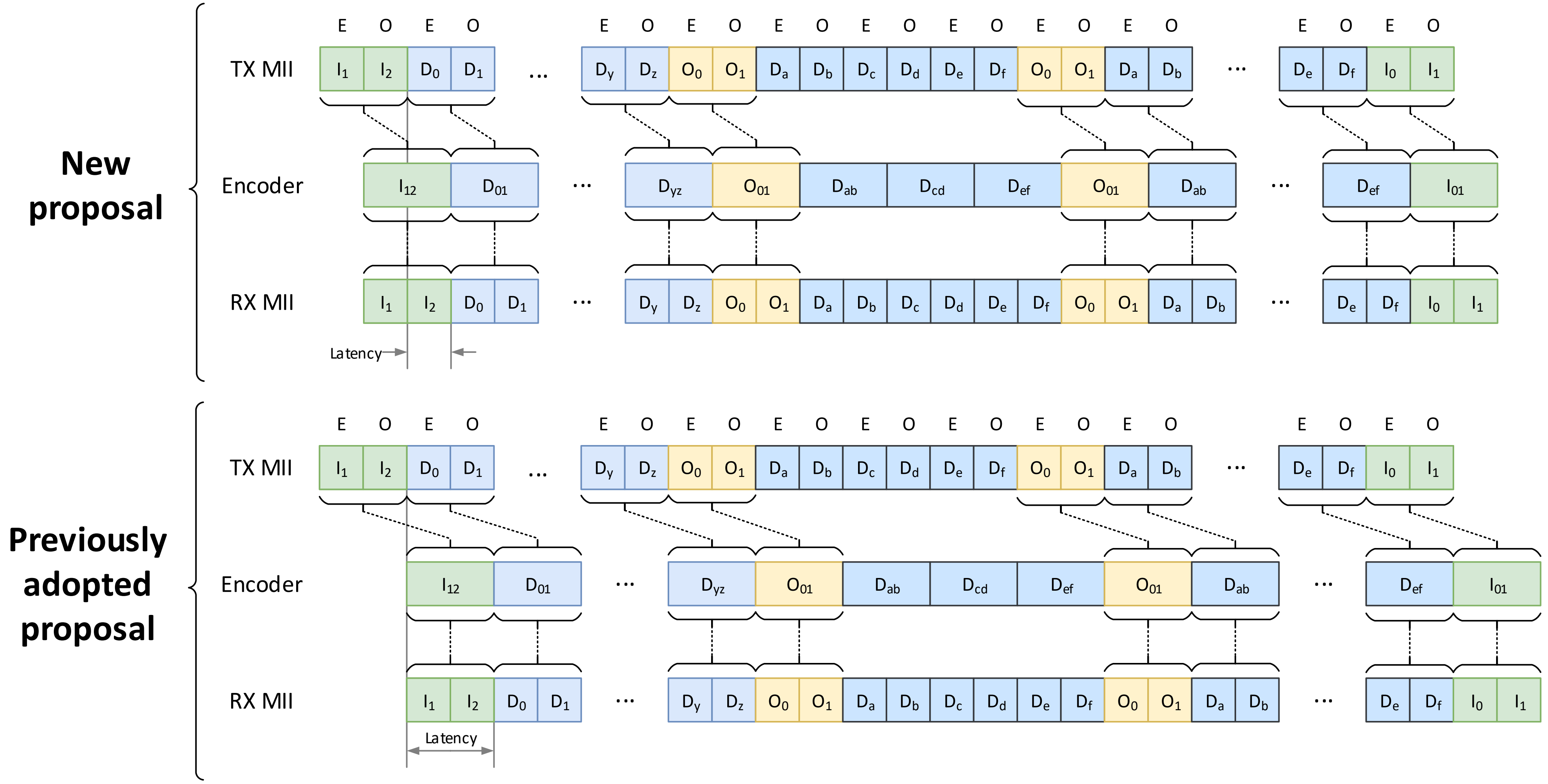
➢ Only difference wrt previously adopted proposal is latency and we don't need to group/ungroup 3 nibbles or expand/compress one idle



Supporting Sequence Ordered Sets with 8N/8N+1

- We can extend the above scheme to include sequence ordered sets
  - We would do this if there is a reason to support end to end communication of sequence ordered sets

Start on Even Cycle, Packet Even Number of Nibbles

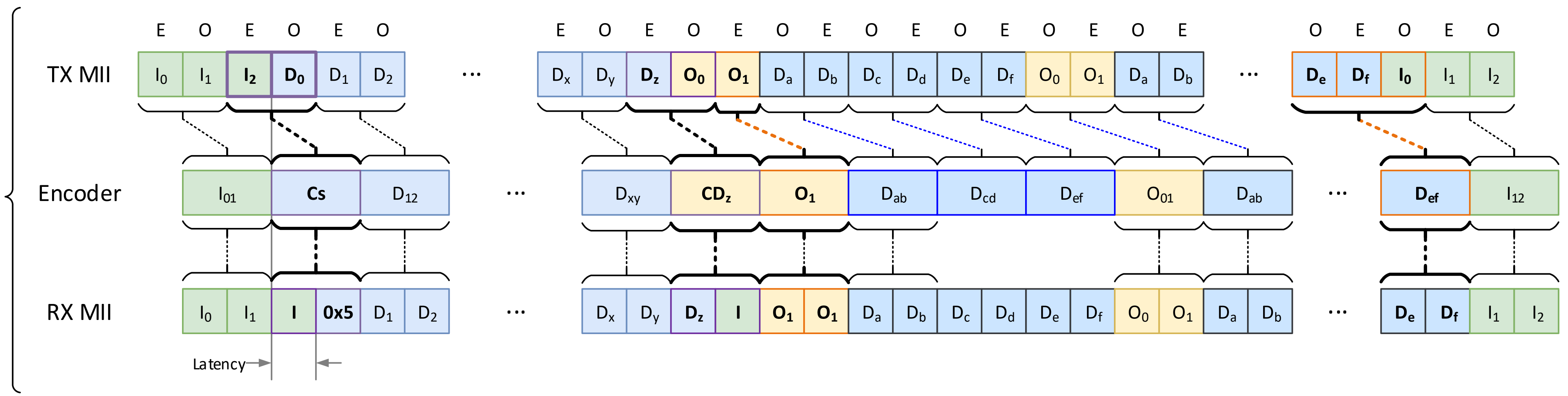


New proposal
Previously adopted proposal
TX MII
Encoder
RX MII
E O E O
I_1 I_2 D_0 D_1
D_y D_z O_0 O_1 D_a D_b D_c D_d D_e D_f O_0 O_1 D_a D_b
D_e D_f I_0 I_1
I_{12} D_{01}
D_{yz} O_{01} D_{ab} D_{cd} D_{ef} O_{01} D_{ab}
D_{ef} I_{01}
Latency

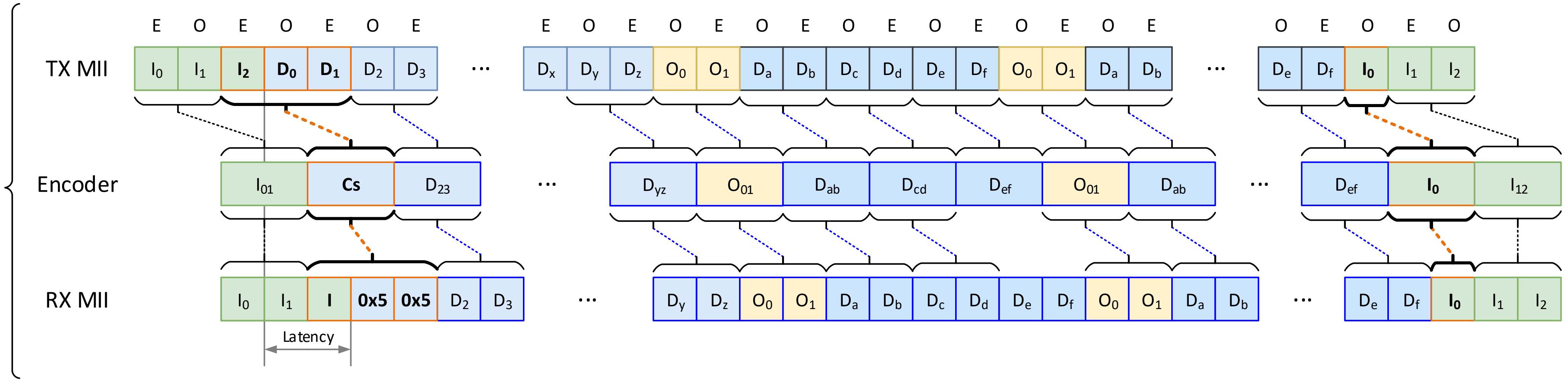
- Only difference is latency

Start on Odd Cycle, Packet Even Number of Nibbles

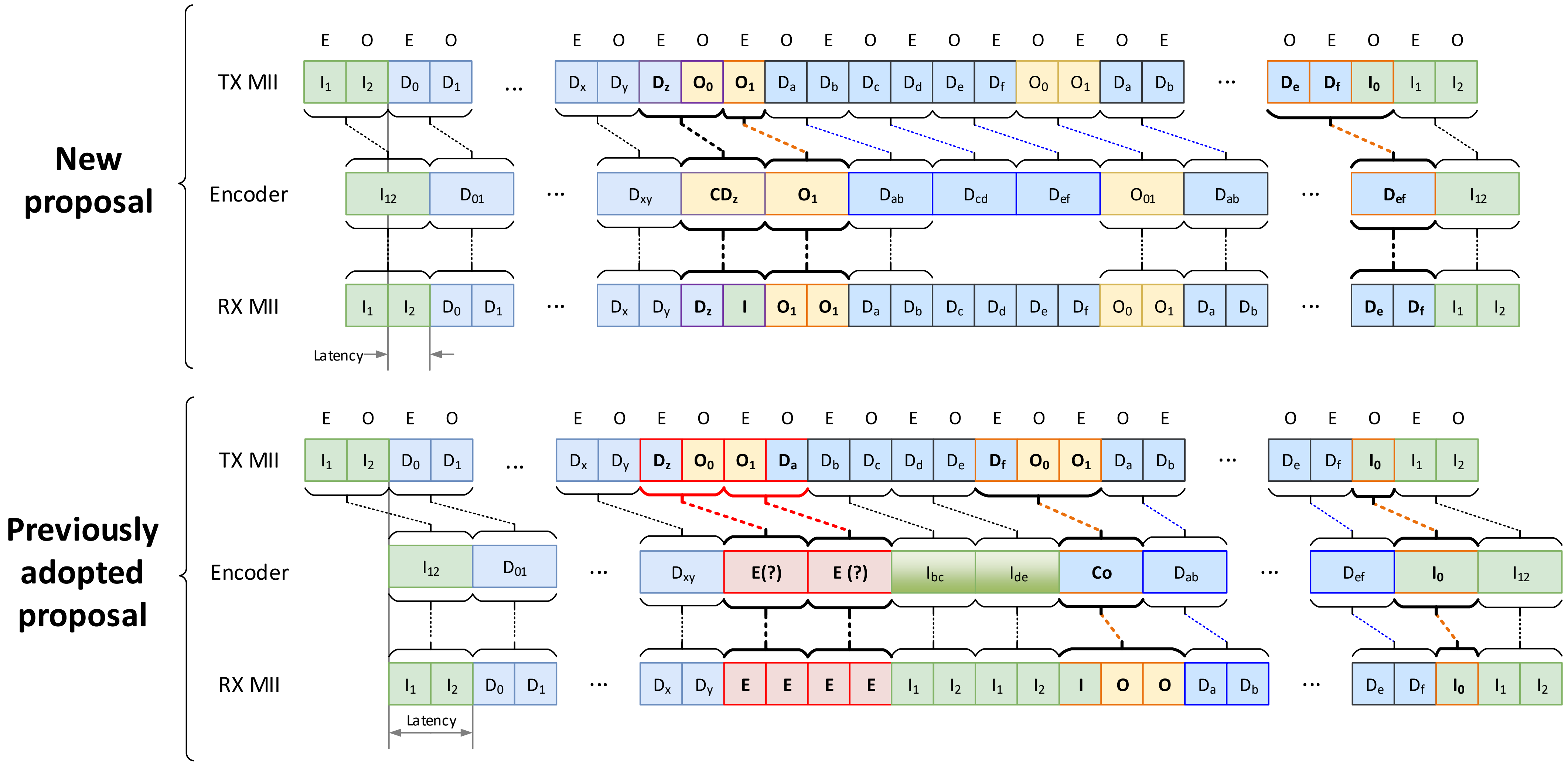
New proposal


E O E O E O
TX MII
Encoder
RX MII
I0 I1 I2 D0 D1 D2
I01 Cs D12
I0 I1 I 0x5 D1 D2
Latency
Dx Dy Dz O0 O1 Da Db Dc Dd De Df O0 O1 Da Db
Dxy CDz O1 Dab Dcd Def O01 Dab
Dx Dy Dz I O1 O1 Da Db Dc Dd De Df O0 O1 Da Db
De Df I0 I1 I2
Def I12
De Df I1 I2

Previously adopted proposal

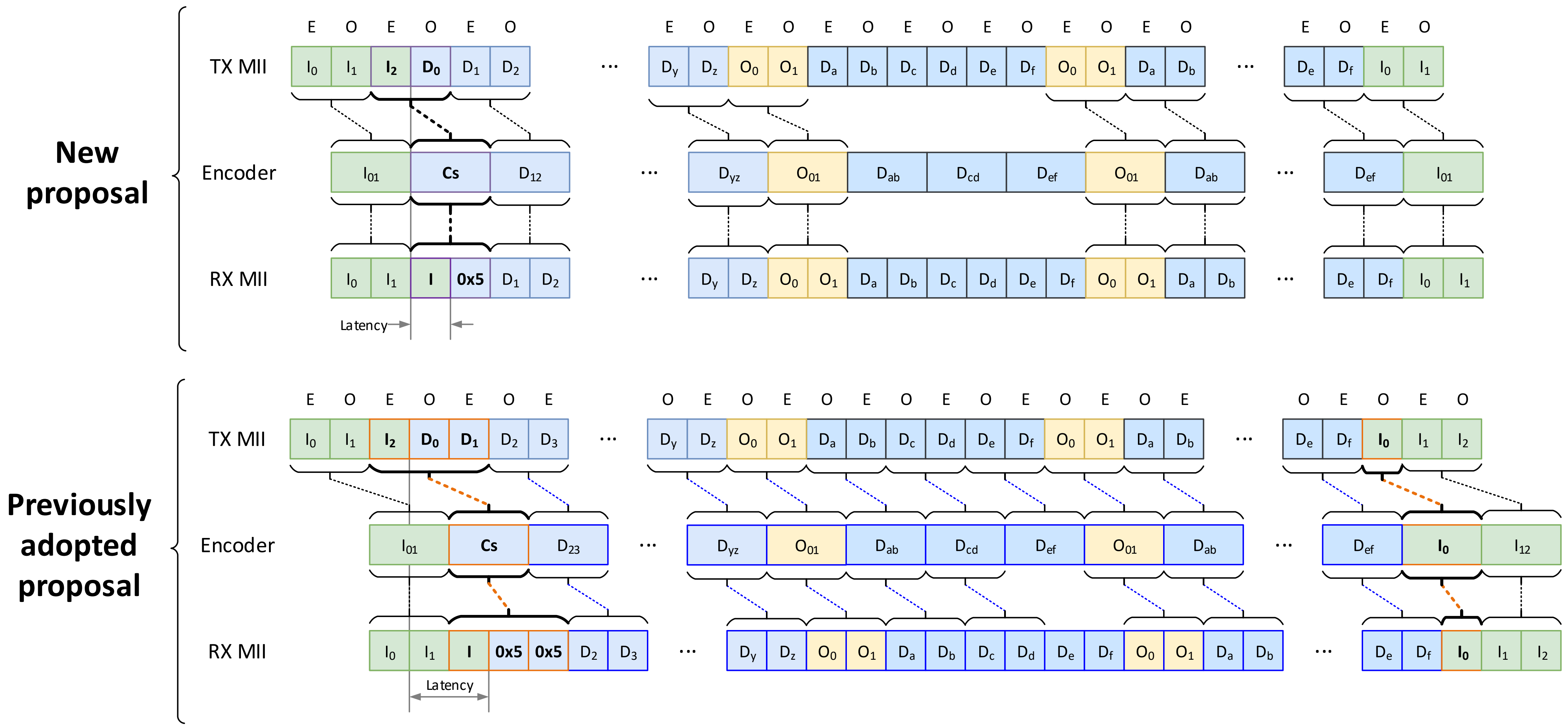


Start on Even Cycle, Packet Odd Number of Nibbles



- Undefined real use case in previously adopted proposal

Start on Odd Cycle, Packet Odd Number of Nibbles


New proposal
Previously adopted proposal
TX MII
Encoder
RX MII
E O E O E O
I_0 I_1 I_2 D_0 D_1 D_2
D_y D_z O_0 O_1 D_a D_b D_c D_d D_e D_f O_0 O_1 D_a D_b
D_e D_f I_0 I_1
I_{01} Cs D_{12}
D_{yz} O_{01} D_{ab} D_{cd} D_{ef} O_{01} D_{ab}
D_{ef} I_{01}
I_0 I_1 I 0x5 D_1 D_2
Latency
I_0 I_1 I_2 D_0 D_1 D_2 D_3
I_{01} Cs D_{23}
D_{ef} I_0 I_{12}
I_0 I_1 I 0x5 0x5 D_2 D_3
D_e D_f I_0 I_1 I_2
Latency

100BASE-T1L PCS Summary

- 8N/8N+1 Encoding and Decoding
  - Constant latency MII to 8N/8N+1 encoding using simple and robust state-less encoding and decoding
  - Do not add the complexity of transferring sequence ordered sets across PCS
- Scrambler:
  - Use a 33 bits side-stream scrambler, as per Clauses 40, 146, 96, 149 and 165 with a new scrambler value generated every 8b6T octet
    - Generate 9 uncorrelated bits from the scrambler each octet; $Sx_n[3:0]$, $Sy_n[3:0]$ and Sg_n
- PMA Training Frame:
  - Same approach as in Clauses 149 and 165; 2nd bit is inverted in the first 15 partial frames
- Energy Efficient Ethernet:
  - Same approach as in Clauses 97, 149 and 165; with 211.2 μs of quiet and 19.2 μs of refresh time
  - PCS encoding transparently carries the LPI signalling MII to MII, even if we don't enter LPI
- OAM
  - OAM is not supported; no vendor specific messages transferred between local and remote PCS

Questions ?