ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Proposal for a 100BASE-T1L PCS

Jacobo Riesco
Brian Murray
Philip Curran

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 MSym/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
  - At the September meeting the 802.3dg group approved a motion to adopt a 33-bit PCS Tx Scrambler and the basic structure of a PMA Training Frame
    - See Murray_3dg_02_09172024
  - At the September meeting the 802.3dg group approved a motion that IEEE 802.3dg not include an OAM channel and set the auxiliary bit in the PHY frame to zero
    - See spmd_TF_minutes_00_091624_4
- This presentation is a continuation of the PCS proposal in

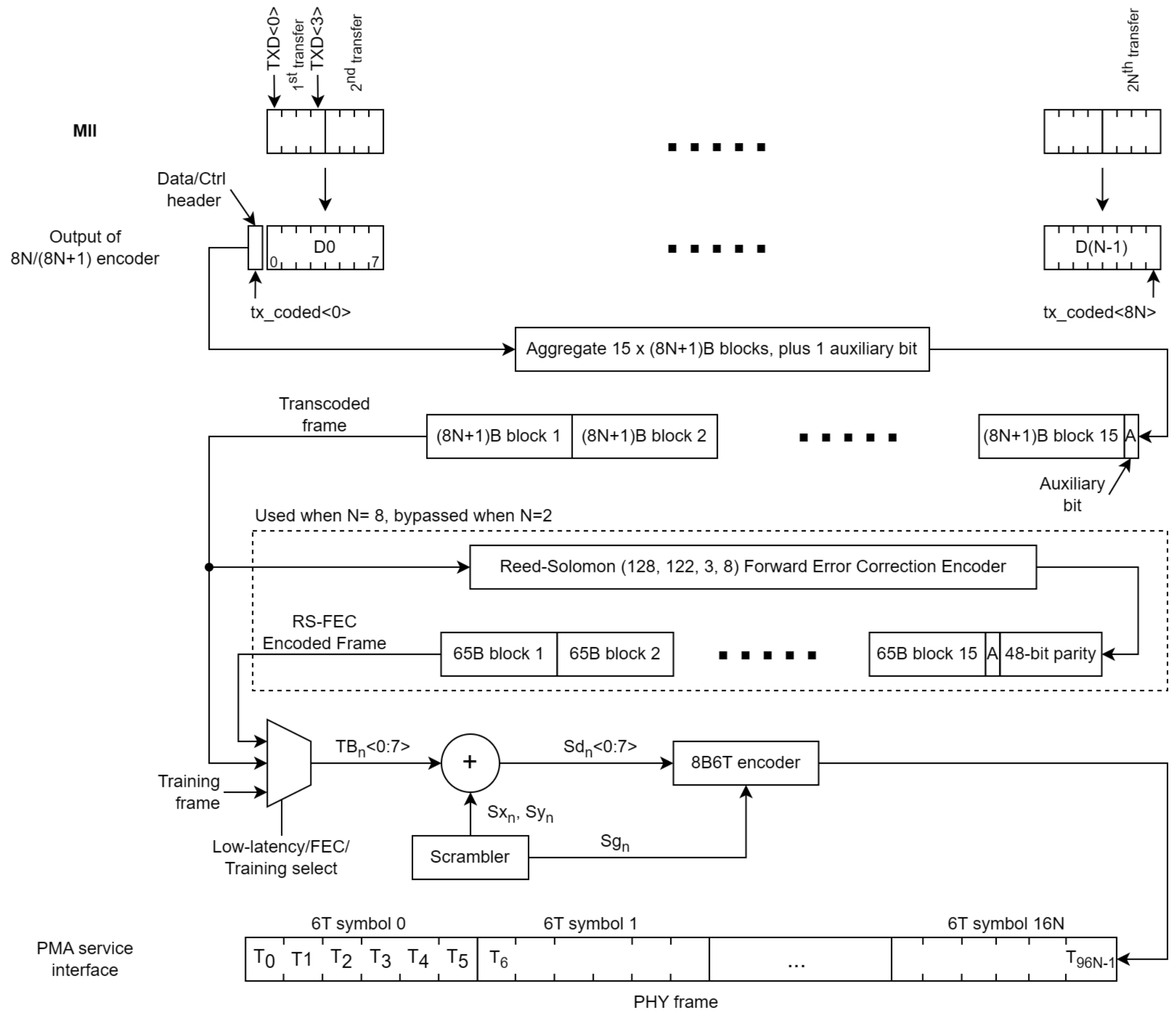
  Riesco_3dg_01a_09172024

8b6T Coding, 8N/8N+1 Block Codes and RS-FEC

- 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 MSym/s in both cases
  - 8b6T coding is described in:
    - Curran_3dg_01a_07152024
    - Murray_3dg_01a_07152024
  - 8N/(8N+1) encoding is described in:
    - Lo_3dg_01_012524
    - Lo_3dg_01a_0724
  - RS FEC is described in
    - Tingting_3dg_02_16_07_2024

This Section is the:
Proposal for Constant Latency MII to 8N/8N+1 Encoding in the 100BASE-T1L PCS

PCS TX Block Diagram and Bit Ordering



N=2: 16B/17B low latency

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

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

- In Riesco_3dg_01a_09172024 we presented a PCS for 100BASE-T1L including a simpler scheme for the constant latency MII to 8N/8N+1 encoding
  - Always encode 2 nibbles into an octet including at the start and end of the packet
  - Always decode an octet into two nibbles
- A limitation with respect to error handling compared to previous PHY technologies has been described in Riesco_3dg_01_10292024
- We propose to resolve this limitation by adding start (S) and End (T) packet delimiter control codes to the 8N/(8N+1) encoding

Control Codes and Mode Encoding/Decoding Table

3	4	5	6	7		
Mode M(n)[0:1]		**Control Code C(n)[0:2]**			**Symbol**	**Definition**
0	-	0	0	0	Q	Sequence Ordered Set Control Code
0	-	0	0	1	E	Transmit Error Propagation
0	-	0	1	0	I	Normal Inter-Frame (Idle) with loc_phy_ready = OK
0	-	0	1	1	**Su**	Start of Packet with leading Idle/LPI
0	-	1	0	0	Tp	End of Packet
0	-	1	0	1	L	Assert Low Power Idle (LPI)
0	-	1	1	0	Ix	Normal Inter-Frame (Idle) with loc_phy_ready = NOT_OK
0	-	1	1	1	Sp	Start of Packet
1	0	0	0	0	**Tu**D0	Dribble Nibble, Data = 0x0
1	0	0	0	1	TuD8	Dribble Nibble, Data = 0x8
1	0	0	1	0	TuD4	Dribble Nibble, Data = 0x4
1	0	0	1	1	TuDC	Dribble Nibble, Data = 0xC
1	0	1	0	0	TuD2	Dribble Nibble, Data = 0x2
1	0	1	0	1	TuDA	Dribble Nibble, Data = 0xA
1	0	1	1	0	TuD6	Dribble Nibble, Data = 0x6
1	0	1	1	1	TuDE	Dribble Nibble, Data = 0xE
1	1	0	0	0	TuD1	Dribble Nibble, Data = 0x1
1	1	0	0	1	TuD9	Dribble Nibble, Data = 0x9
1	1	0	1	0	TuD5	Dribble Nibble, Data = 0x5
1	1	0	1	1	TuDD	Dribble Nibble, Data = 0xD
1	1	1	0	0	TuD3	Dribble Nibble, Data = 0x3
1	1	1	0	1	TuDB	Dribble Nibble, Data = 0xB
1	1	1	1	0	TuD7	Dribble Nibble, Data = 0x7
1	1	1	1	1	TuDF	Dribble Nibble, Data = 0xF

* From Lo_3dg_01a_0724
Modified

- M(n)[0:1] = 00 – No more control codes
- M(n)[0:1] = 01 – More control codes
- M(n)[0] = 1 - 16 x Tu control codes
  - More control codes implied

Sequence Ordered Sets

- Optional support for sequence ordered sets
- SEQen bit added to the InfoField to allow support for sequence ordered sets to be negotiated with the link partner
  - If either link partner has SEQen = 0 then sequence ordered sets are disabled
- If sequence ordered sets are disabled
  - If the PHY receives a sequence ordered set on the MII the PHY Tx will encode it as Idles
  - If the PHY Rx receives a sequence ordered set control code (Q) at the decoder, this is not expected or supported and is treated like any other error
    - Following the error handling rules in such cases a False Carrier will be signalled at the MII
- If sequence ordered sets are enabled, they are transparently conveyed between the local and remote MII

Normal Inter-Frame (Idle)

- Keep two versions of Normal Inter-Frame
  - **I** when loc_phy_ready = OK or **Ix** when loc_phy_ready = NOT_OK
  - As in Clause 97
- Normal Inter-Frame encoding indicates whether the PHY is ready to receive data or not (loc_phy_ready = OK/NOT_OK)
  - Decoded as rem_phy_ready = OK/NOT_OK in the link partner
  - Synchronizes link_status = OK between link partners
  - Cannot rely on sequence ordered sets for this synchronization as existing MACs do not support them

Encoding Rules for Error Handling

- Packets are delimited with Start of Packet and End of Packet symbols
  - **Sp** symbol used for packets starting on even cycles - aligned
  - **Su** (was Cs) for packet starting on odd cycles - unaligned
    - The start of packet is always encoded using Sp or Su
    - If transmit error propagation is encoded at the MII during the start of packet, an error is encoded in the following octet
  - **Tp** symbol used for packets ending on even cycles - aligned
  - 16 x **Tu** (was CD) symbols for packets ending on odd cycles – unaligned
    - The end of packet is always encoded using Tp or Tu

Decoding Rules for Error Handling

- PCS receive functions or Multi-G RS functions are incorporated into the PHY decoder
  - RX_DV will be asserted in response to the reception of a start of packet control code if the previous nibble was a normal inter-frame
  - RX_DV will be de-asserted when a control code other than an error is received
    - When RX_DV is de-asserted because of a control code other than end of packet, RX_ER will be asserted before RX_DV is de-asserted
  - If a symbol other than Idle, LPI, start of packet or sequence ordered set control code (if supported), is received following an idle nibble or a sequence ordered set (if supported), then a false carrier indication is encoded onto the MII RX
    - False carrier is held until an Idle or a sequence ordered set control code (if supported) is received
    - This is the same as 100BASE-X, 1000BASE-X, 1000BASE-T, 100BASE-T1, 10BASE-T1L and 10BASE-T1S (Clauses 24, 36, 40, 96, 146 and 147)

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.

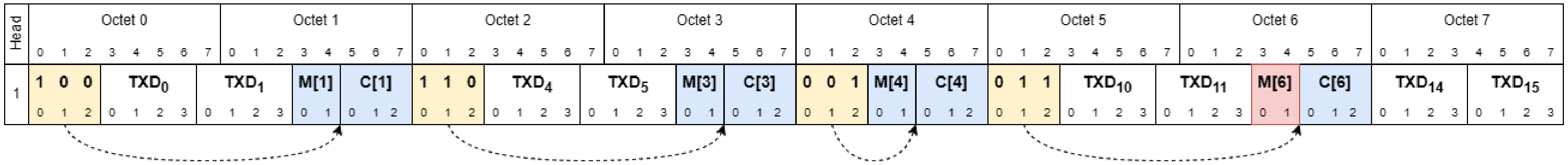
TC[n]	: 0 if octet n is encoded as a data packet octet (the octet n contains two MII data nibbles, TXD[2n][0:3] and TXD[2n+1][0:3]); 1 otherwise.
TC[-1]	: 1 by definition
TD[n][0:7]	: MII octet n (TD[n][0:3] = TXD[2n][0:3], TD[n][4:7] = TXD[2n+1][0:3]) if TC[n] = 0
B[0:8N]	: 8N+1 block. Bit 0 transmitted 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 2 is MSB
C(n)[0:2]	: MII control code n, corresponding to MII data nibbles 2n, 2n+1 as per control codes and mode encoding table
M(n)[0:1]	: MII mode n, corresponding to MII data nibbles 2n, 2n+1 M(n)[0] = 1 if encoded symbol is CD; else 0 M(n)[1] = TXD[2n][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]   else if TC[n-1] = 1
                 TD[n-1][5:7]   else
B[8n+4:8n+5]   = TD[n][3:4]     if OR(n) = 0
                 M(n)[0:1]      else if TC[n] = 1
                 TD[n][0:1]     else
B[8n+6:8n+8]   = TD[n][5:7]     if OR(n) = 0
                 C(n)[0:2]      else if TC[n] = 1
                 TD[n][2:4]     else
```

- *No functional changes with respect to July*
- *Fixed inconsistencies in syntax*
- *Simplified the conditions for the encoding of the B[0:8N] bits*

8N/(8N+1) Encoding Example

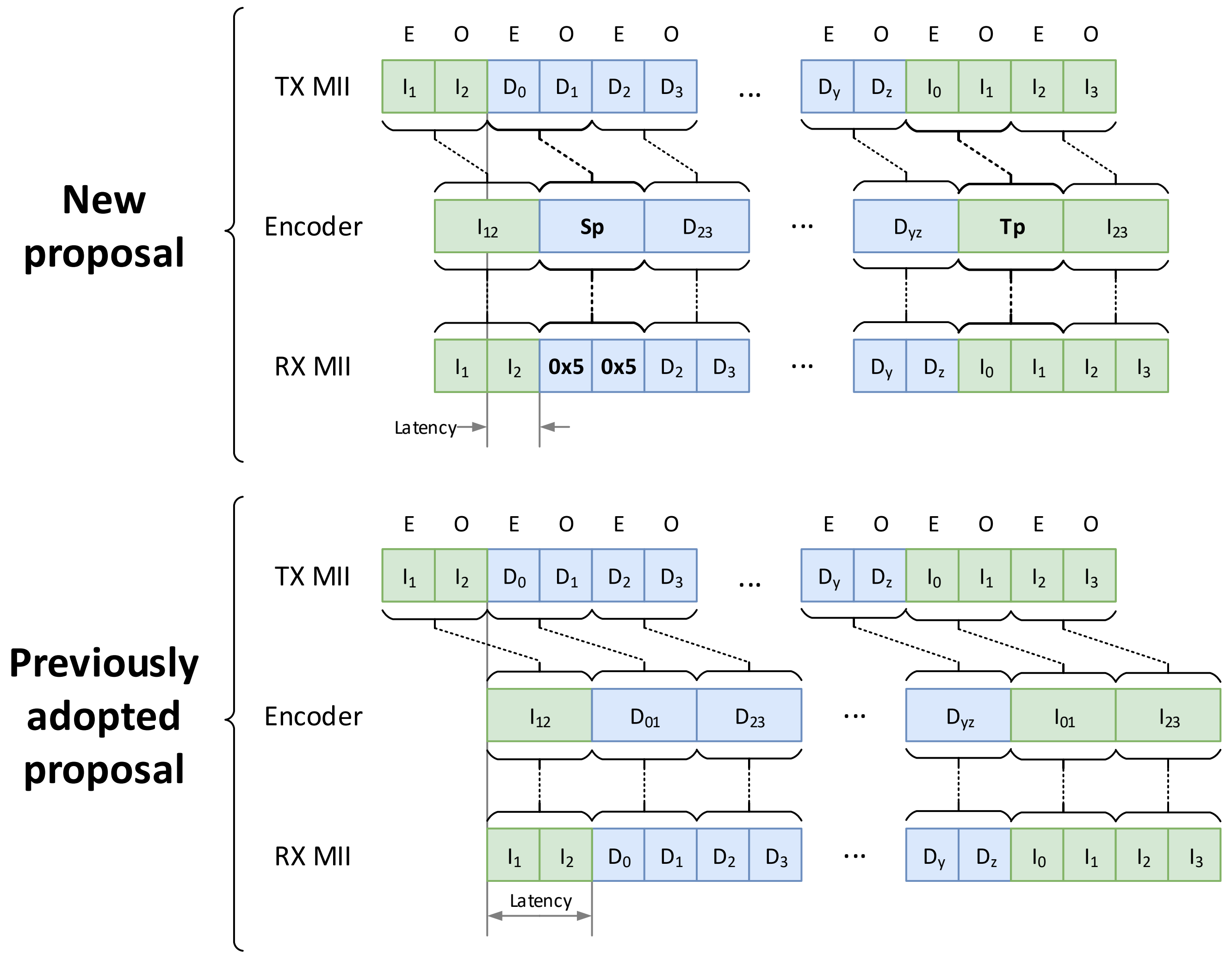
- Example encoding for a N = 8 block containing the following MII octets $D_0/C_1/D_2/C_3/C_4/D_5/C_6/D_7$
  - Octets are shown in the transmit order, and bits within an octet and its sub-fields are also shown in the transmit order, i.e. LSB first



Start on Even Cycle, Packet Even Number of Nibbles

- Encoding: combine even + odd nibbles into octet presented to encoder
  - First 2 preamble data nibbles in octet, encoded to Sp (aligned start of packet)
  - First 2 idle nibbles in octet after last data octet, encoded to Tp (aligned end of packet)
- Decoding: split octet to 2 nibbles
  - Sp decoded to two preamble (0x5) data nibbles
  - Tp decoded to two idle nibbles

➢ Only difference wrt previously adopted proposal is latency and aligned start and end of packet control symbols

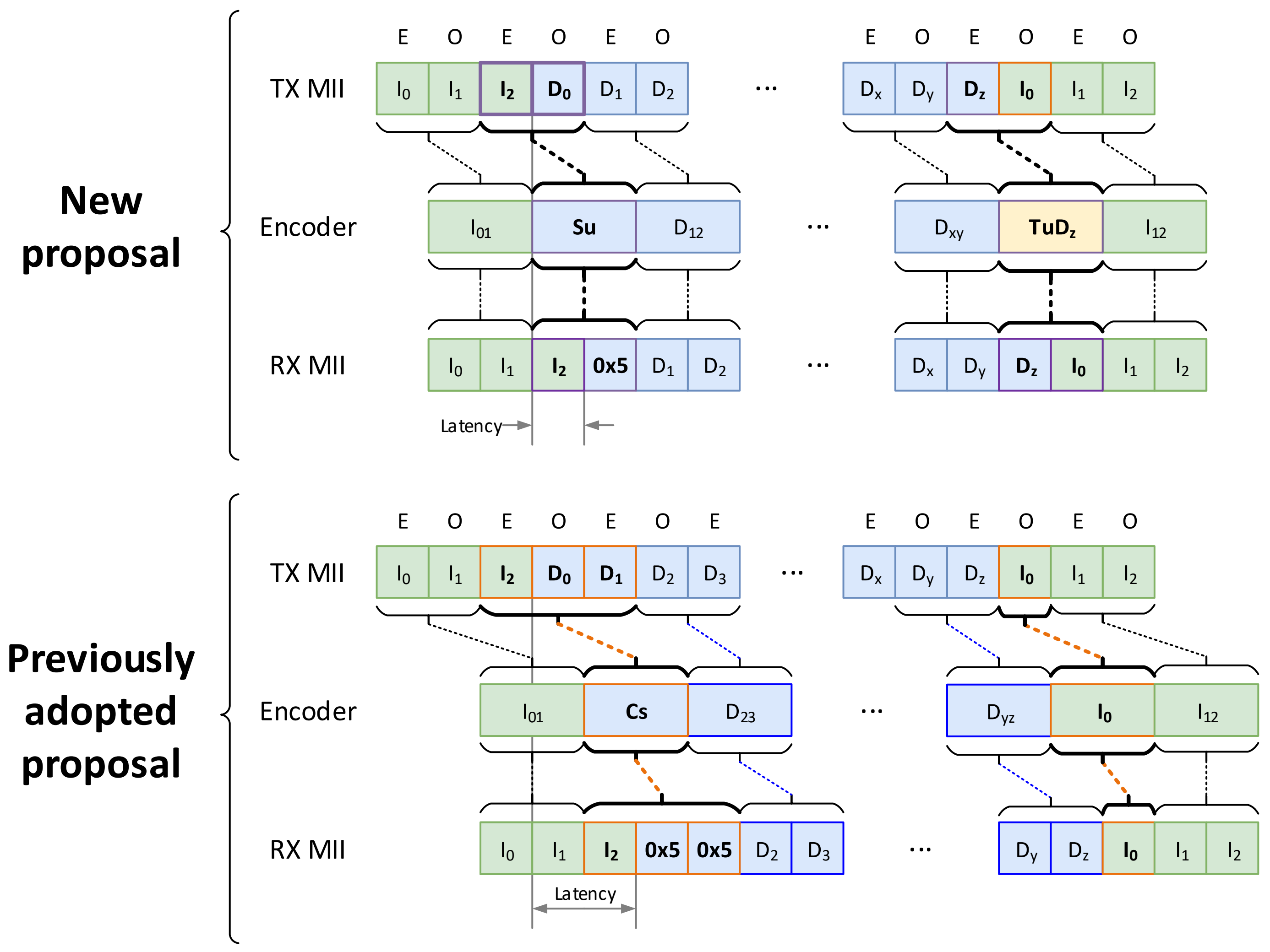


New proposal
E O E O E O
TX MII
I_1 I_2 D_0 D_1 D_2 D_3 … D_y D_z I_0 I_1 I_2 I_3
Encoder
I_{12} Sp D_{23} … D_{yz} Tp I_{23}
RX MII
I_1 I_2 0x5 0x5 D_2 D_3 … D_y D_z I_0 I_1 I_2 I_3
Latency
Previously adopted proposal
E O E O E O
TX MII
I_1 I_2 D_0 D_1 D_2 D_3 … D_y D_z I_0 I_1 I_2 I_3
Encoder
I_{12} D_{01} D_{23} … D_{yz} I_{01} I_{23}
RX MII
I_1 I_2 D_0 D_1 D_2 D_3 … D_y D_z I_0 I_1 I_2 I_3
Latency

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 Su (unaligned start of packet)
  - Data nibble (D_z) followed by idle nibble in octet, encoded to TuD_z (unaligned end of packet)
- Decoding: Split octet to 2 nibbles
  - Su decoded to idle nibble followed by 0x5 data nibble
  - TuD_z decoded to D_z data nibble followed by idle nibble

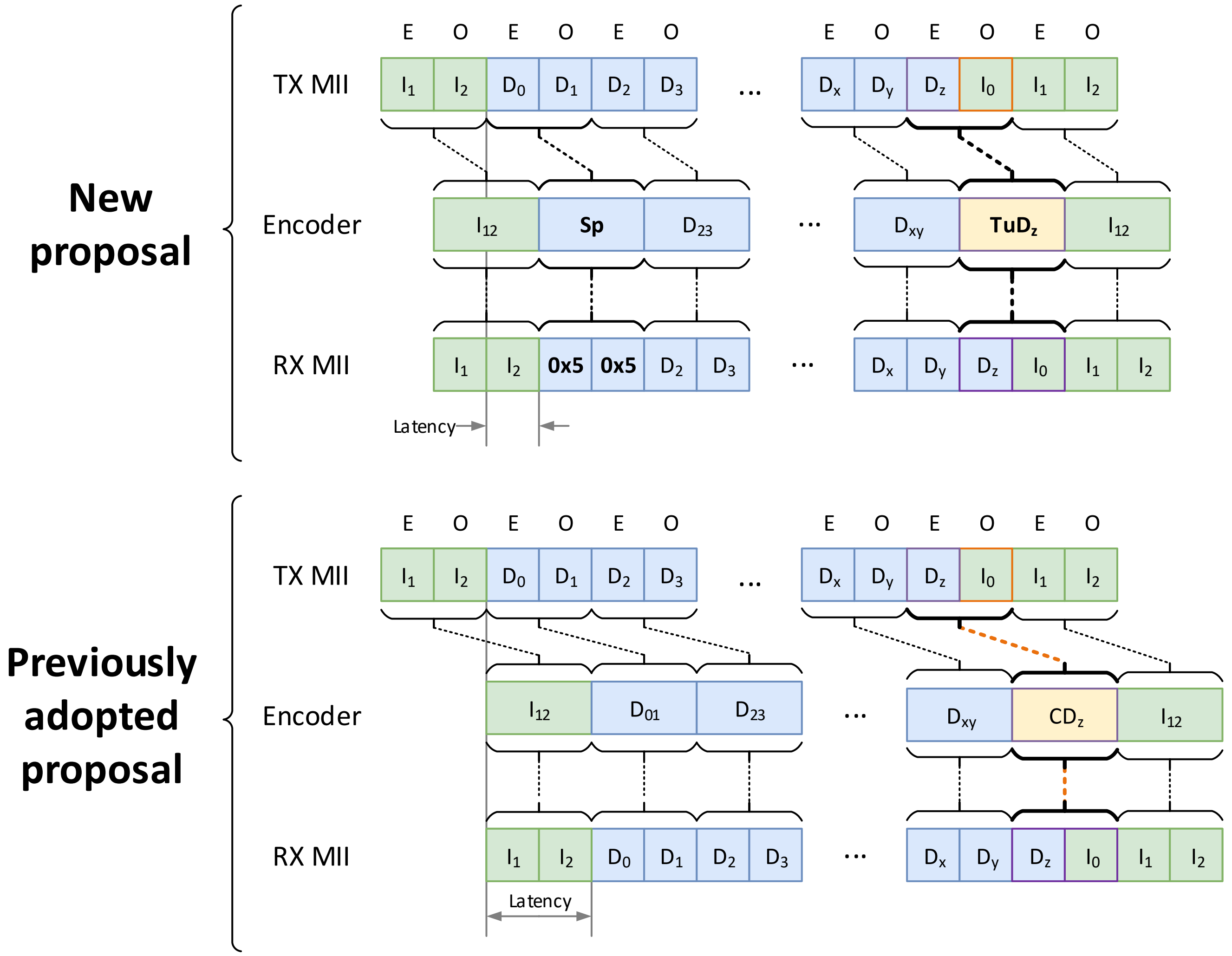
➢ Only difference wrt previously adopted proposal is latency, unaligned start and end of packet control symbols and we do not need to group/ungroup 3 nibbles or expand/compress one idle


New proposal
E O E O E O
E O E O E O
TX MII
I0 I1 I2 D0 D1 D2 … Dx Dy Dz I0 I1 I2
Encoder
I01 Su D12 … Dxy TuDz I12
RX MII
I0 I1 I2 0x5 D1 D2 … Dx Dy Dz I0 I1 I2
Latency
Previously adopted proposal
E O E O E O E
E O E O E O
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 I2 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
  - First 2 preamble data nibbles in octet, encoded to Sp
  - Data nibble (D_z) followed by Idle nibble in octet, encoded to TuD_z
- Decoding: Split octet to 2 nibbles
  - Sp decoded to two 0x5 data nibbles
  - TuD_z decoded to D_z data nibble followed by Idle nibble

➢ Only difference wrt previously adopted proposal is latency and aligned start of packet control symbol



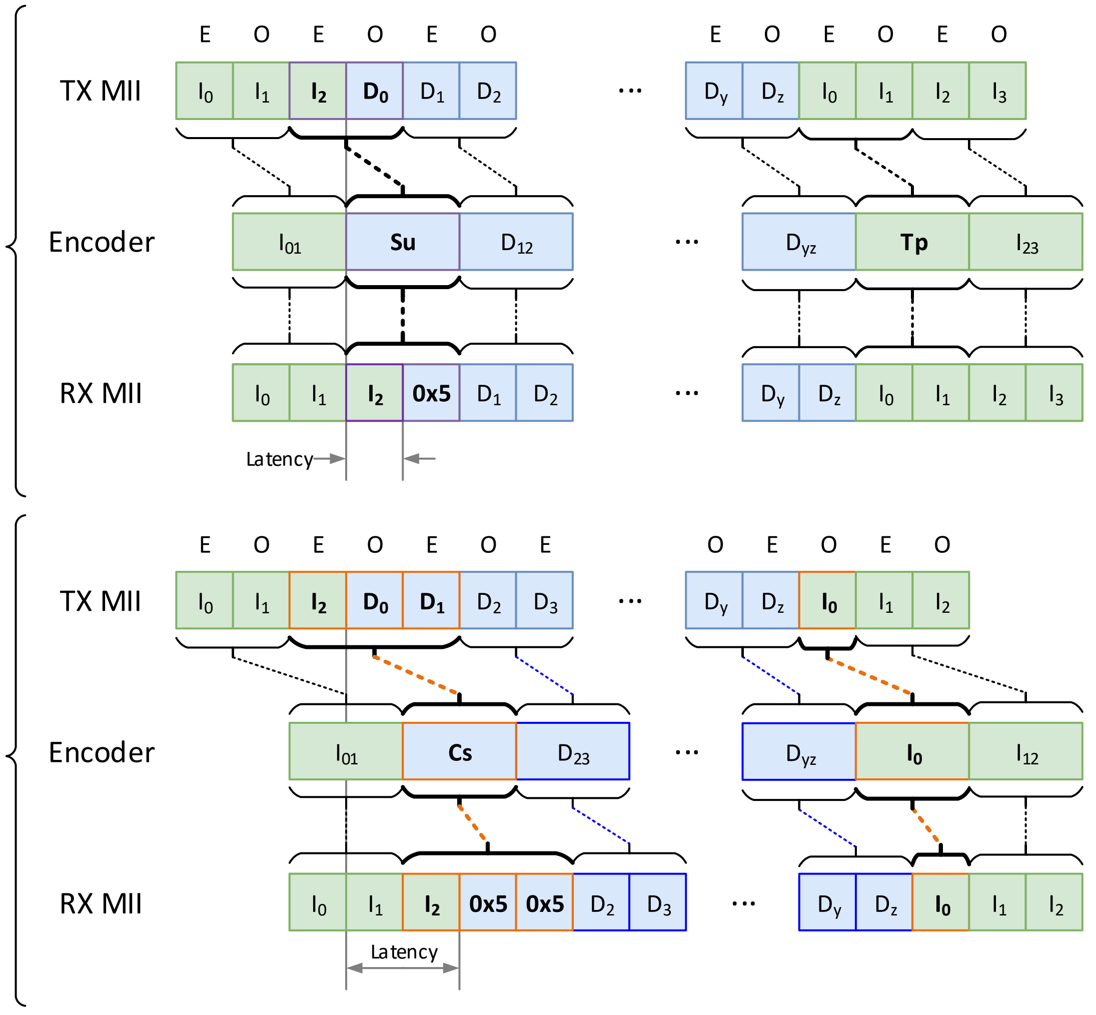
Start on Odd Cycle, Packet Odd Number of Nibbles

- Encoding: combine even + odd nibbles into octet presented to encoder
  - Idle nibble followed by first preamble data nibble in octet, encoded to Su
  - First 2 idle nibbles in octet after last data octet, encoded to Tp
- Decoding: Split octet to 2 nibbles
  - Su decoded to idle nibble followed by 0x5 data nibble
  - Tp decoded to two idle nibbles

➢ Only difference wrt previously adopted proposal is latency, aligned end of packet control symbol, and we don't need to group/ungroup 3 nibbles or expand/compress one idle

New proposal

Previously adopted proposal



E O E O E O
TX MII
I_0 I_1 I_2 D_0 D_1 D_2 … D_y D_z I_0 I_1 I_2 I_3
Encoder
I_{01} Su D_{12} … D_{yz} Tp I_{23}
RX MII
I_0 I_1 I_2 0x5 D_1 D_2 … D_y D_z I_0 I_1 I_2 I_3
Latency
E O E O E O E
O E O E O
TX MII
I_0 I_1 I_2 D_0 D_1 D_2 D_3 … D_y D_z I_0 I_1 I_2
Encoder
I_{01} Cs D_{23} … D_{yz} I_0 I_{12}
RX MII
I_0 I_1 I_2 0x5 0x5 D_2 D_3 … D_y D_z I_0 I_1 I_2
Latency

This Section is the:
Proposal for the Scrambler and PMA Training Frame used in the 100BASE-T1L PCS

Reed Solomon Forward Error Correction (RS-FEC)

- Optional support for RS-FEC
  - RS-FEC provides burst error protection at the cost of latency
- RSen bit added to the InfoField to allow support for RS-FEC to be negotiated with the link partner
  - If both link partners have RSen = 1 then RS-FEC is enabled
  - Latency sensitive applications should disable support for RS-FEC
- See previously adopted proposal Murray_3dg_01a_07152024 for more details
  - Which in turn references slide 5 of Tingting_3dg_02_16_07_2024 where the details of RS FEC are described

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 bit stream is sliced into octets $TB_n[7:0]$
  - 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
  - Scrambling 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 $TB_n[7:0]$ during transmission
    - Sg_n is used to randomize the sign of the non-negative disparity (NND) 6-tuple 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:

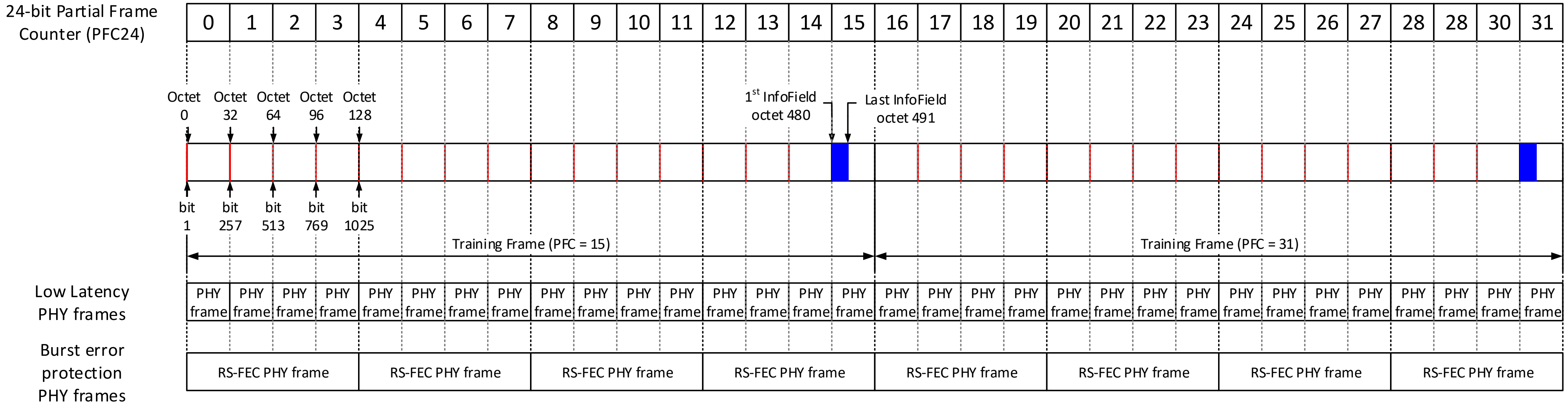
$$Sd_n[7:4] = Sx_n[3:0] \wedge TB_n[7:4]$$
$$Sd_n[3:0] = Sy_n[3:0] \wedge TB_n[3:0]$$

8b/6T Encoding

- The bits $Sd_n[7:0]$ are used to select an NND 6-tuple
- Running disparity control is applied
- See previously adopted proposal Murray_3dg_01a_07152024 for more details

PMA Training Frame

- The PMA training frame follows a similar approach as 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 frames



PMA Training Frame: Scrambling

- All the bits in the training frame are zero except for:
  - The second bit in the first 15 partial frames which are set to 1 (in red in the previous figure)
  - The InfoField (in blue in the previous figure)
- $Scr_n[0]$ is available on the first bit of each octet except during the InfoField to facilitate scrambler acquisition at the receiver
- With the proposed definition of the InfoField we ensure:
  - The second bit of each partial frame is set to 1 to facilitate partial frame alignment
  - The 3rd bit of the 16th partial frame is set to 1 to facilitate PHY frame alignment
- The first 12 octets (96 bits) in the 16th partial frame are XORed with the contents of the InfoField[0:95]
- The scrambled training octet $Sd_n[7:0]$ is 8b/6T encoded as in normal operation
  - Ternary symbols (PAM3) with bounded running disparity are also used during training

PMA Training Frame: Scrambling

- The scrambled octets during training are:

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

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

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

$$Sd_n[1] = \begin{cases} Sy_n[1] \wedge InfoField[m+1] & 480 \leq (n\,mod\,512) \leq 491 \\ Sy_n[1] \wedge 1 & else\ if\ m = 0 \\ Sy_n[1] & otherwise \end{cases}$$

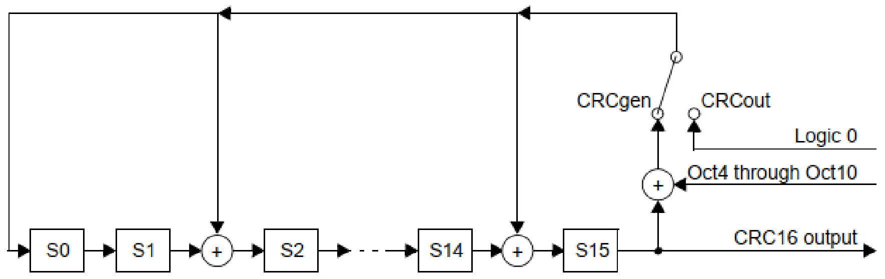
$$Sd_n[0] = \begin{cases} Sy_n[0] \wedge InfoField[m] & 480 \leq (n\,mod\,512) \leq 491 \\ Sy_n[0] = Scr_n[0] & otherwise \end{cases}$$

PMA Training Frame: PHY Control InfoField

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

Octet 1 <7:0>	Octet 2 <7:0>	Octet 3 <7:0>	Octet 4 <7:0>	Octet 5 <7:0>	Octet 6 <7:0>	Octet 7 <7:0>	Octet 8 <7:0>	Octet 9 <7:0>	Octet 10 <7:0>	Octet 11 <7:0>	Octet 12 <7:0>
Header <7:0>	Header <15:8>	Header <23:16>	PFC24 <7:0>	PFC24 <15:8>	PFC24 <23:26>	Message	PHY capability / Countdown			CRC16	

  - Octets sent in order, LSB first



CRC16 same as Clauses 97, 149 and 165

PHY Control InfoField: Training Format

- This phase is used to exchange the PHY capability information between the link partners
  - Same approach as in Clauses 97, 149 and 165

Octet 1 <7:0>	Octet 2 <7:0>	Octet 3 <7:0>	Octet 4 <7:0>	Octet 5 <7:0>	Octet 6 <7:0>	Octet 7 <7:0>	Octet 8 <7:0>	Octet 9 <7:0>	Octet 10 <7:0>	Octet 11 <7:0>	Octet 12 <7:0>
0xEE	0xA7	0x00	PFC24 <7:0>	PFC24 <15:8>	PFC24 <23:16>	Message	PHY Capability			CRC16	

PMA_state			en_slave_tx (master)	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

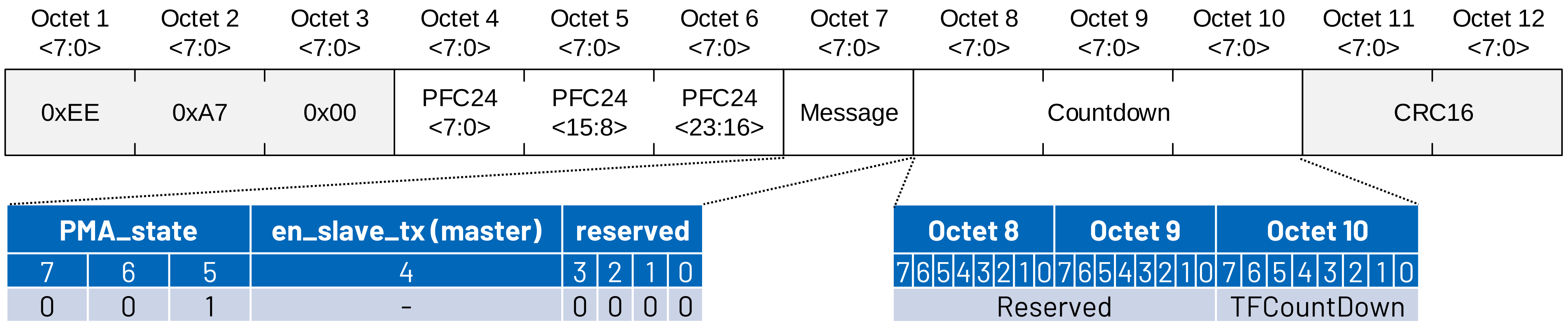
Octet 8								Octet 9								Octet 10							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Reserved																SEQen	EEECntrl[1:0]		RSen	Reserved			

- The PMA_state has been expanded to 3 bits and incorporates signalling of receiver status
- It is proposed not to send timing_lock_OK from the slave since it cannot be transferred reliably to the master before timing lock has been acquired

- SEQen advertises support for sequence ordered sets
- EEECntrl[1:0] advertises the EEE abilities
- RSen advertises support for RS-FEC

PHY Control InfoField: Countdown Format

- The countdown phase is used to synchronize the switch from training to idle
  - Same approach as in Clauses 97, 149 and 165
  - The slave must align its partial frame count with the master before it starts the countdown



Octet 1 <7:0>	Octet 2 <7:0>	Octet 3 <7:0>	Octet 4 <7:0>	Octet 5 <7:0>	Octet 6 <7:0>	Octet 7 <7:0>	Octet 8 <7:0>	Octet 9 <7:0>	Octet 10 <7:0>	Octet 11 <7:0>	Octet 12 <7:0>
0xEE	0xA7	0x00	PFC24 <7:0>	PFC24 <15:8>	PFC24 <23:16>	Message	Countdown			CRC16	

PMA_state			en_slave_tx (master)	reserved			
7	6	5	4	3	2	1	0
0	0	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
Reserved																TFCountDown							

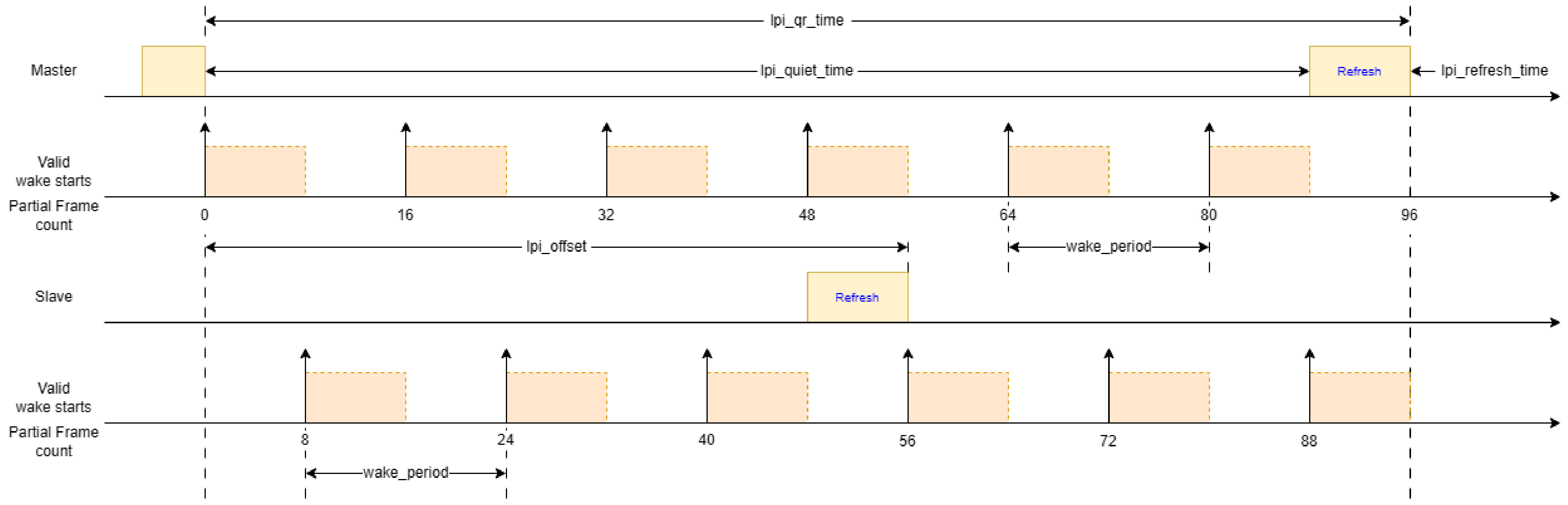
- TFCountDown[7:0] indicates the number of PMA training frames that will be transmitted after the current one before switching to idle

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

This Section is the:
Proposal for EEE in the 100BASE-T1L PCS

Optional EEE Capability

- EEE uses the same approach as in Clauses 97, 149 &165
- EEE abilities advertised in InfoField (EEECntrl[1:0] bits in the PHY Capability field)
  - LPI only entered when both link partners have both EEECntrl bits set
  - If EEECntrl[1] = 1 and EEECntr[0] = 0, the *PCS encoding transparently carries the LPI signalling MII to MII*, but the PHY does not enter the LPI quiet-refresh cycle
  - If EEECntrl[1] = 0, LPI is encoded as a normal inter frame (Idle), and decoded as Idle (or 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 data encoded 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)
    - PHY health information is sent using auxiliary bit
  - 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

Auxiliary Bit - Insufficient LPI Refresh

- Only 1 auxiliary bit is available in the proposed encoding for the low latency mode and the burst error protection mode
  - This bit is only used when LPI entry is enabled and is normally zero
  - This bit will be used to indicate that LPI refresh is insufficient to maintain reliable operation of the receiver
    - When set and the link partner is in LPI, requests the link partner to exit LPI
    - When set, requests the link partner not to enter LPI

OAM

- OAM is not supported
  - No side-channel communications

100BASE-T1L PCS Summary

- 8N/8N+1 Encoding and Decoding
  - Constant latency MII to 8N/8N+1 encoding using simple and robust encoding and decoding
  - Optional support for sequence ordered sets
  - Add start of packet and end of packet delimiters consistent with previous technologies for robust error handling
- 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
  - Similar approach as in Clauses 55, 97, 149 and 165
    - Composed of 16 partial frames with 32 octets per partial frame
    - InfoField transmitted in 16th partial frame of each training fame
    - Structured to facilitate scrambler acquisition and frame alignment
- 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 can transparently carry the LPI signalling MII to MII, even if PHY does not enter LPI
  - The auxiliary bit is used to indicate that LPI refresh is insufficient to maintain reliable operation of the receiver
- OAM
  - OAM is not supported; no vendor specific messages transferred between local and remote

Questions ?