64b/66b line code

Marek Hajduczenia, PhD

ZTE Corporation

marek.hajduczenia@zte.pt

Summary

- This slide deck provides overview of the 64b/66b line coding, its purpose, code structure, available control codes, etc.
- We also briefly look at the reasons for using 64b/66b line coding in 10G-EPON.
- Suggestions for the line coding in EPoC are also made, indicating that decision on the use and the type of code influences many other functions in EPoC.
- A decision on line code is needed to move the draft development forward.

Why is line code important for EPoC?

- EPoC will need data structure with predictable characteristics, allowing the receiving side to synchronize to the incoming data stream
  - A well designed line code can facilitate the process of acquiring synchronization, speeding it up
  - In simpler terms: thanks to a selected line code, we can easily find start and end of frames.
- EPoC might also need to use Ordered Sets to exchange information between local and remote PCS, including local and remote fault indication.
- So what options for line code do we have for EPoC?

Line Code choices (I)

- 8b/10b (25% overhead), as used in the following PHYs:
  - P2P fiber: 1000BASE-LX10 (59), 1000BASE-BX10 (59), 10GBASE-LX4 (53), 10GBASE-CX4 (54), 1000BASE-LX (38), 1000BASE-SX (38)
  - P2MP fiber: 1G-EPON (60), 10/1G-EPON (upstream) (75)
  - Backplane: 1000BASE-KX (70), 10GBASE-KX4 (71)
  - Cable assembly: 10GBASE-CX4 (84), 1000BASE-CX (39)

Line Code choices (II)

- 64b/66b (3.125% overhead), as used in the following PHYs:
  - P2P fiber: 100GBASE-CR10 (86), 100GBASE-SR10 (86), 40GBASE-LR4 (87), 100GBASE-LR4 (88), 100GBASE-ER4 (88), 40GBASE-FR (89), 10GBASE-SR (52), 10GBASE-SW (52), 10GBASE-LR (52), 10GBASE-LW (52), 10GBASE-ER (52), 10GBASE-EW (52), 10GBASE-LRM (68)
  - P2MP fiber: 10/10G-EPON (75), 10/1G-EPON (downstream) (75)
  - Backplane: 10GBASE-KR (72), 40GBASE-KR4 (84)
  - Cable assembly: 40GBASE-CR4 (85), 100GBASE-CR10 (85)

Line Code choices (III)

- 64b/65b (1.5625% overhead) :
  - P2P copper: 10GBASE-T (55)
- 4D-PAM5
  - P2P copper: 1000BASE-T (40)
- 8B6T
  - P2P copper: 100BASE-T4 (23)
- 4b/5b (25% overhead)
  - P2P copper: 100BASE-TX (25)
  - P2P fiber: 100BASE-FX (26)

Line Code choices (IV)

- New line code designed explicitly for EPoC
  - Longer development time for the spec and silicon. Long-term testing of a new line code would be done by fire on real EPoC devices in production networks.
  - A new code might be more optimized for coax transmission than current 64b/66b code and have lower overhead
- Development of a new line code adds to project timeline and uncertainty to performance of the final product
- 64b/66b has been used commonly for the vast majority of 1G+ Ethernet PHYs developed in recent years.

64b/66b line encoding process (I)

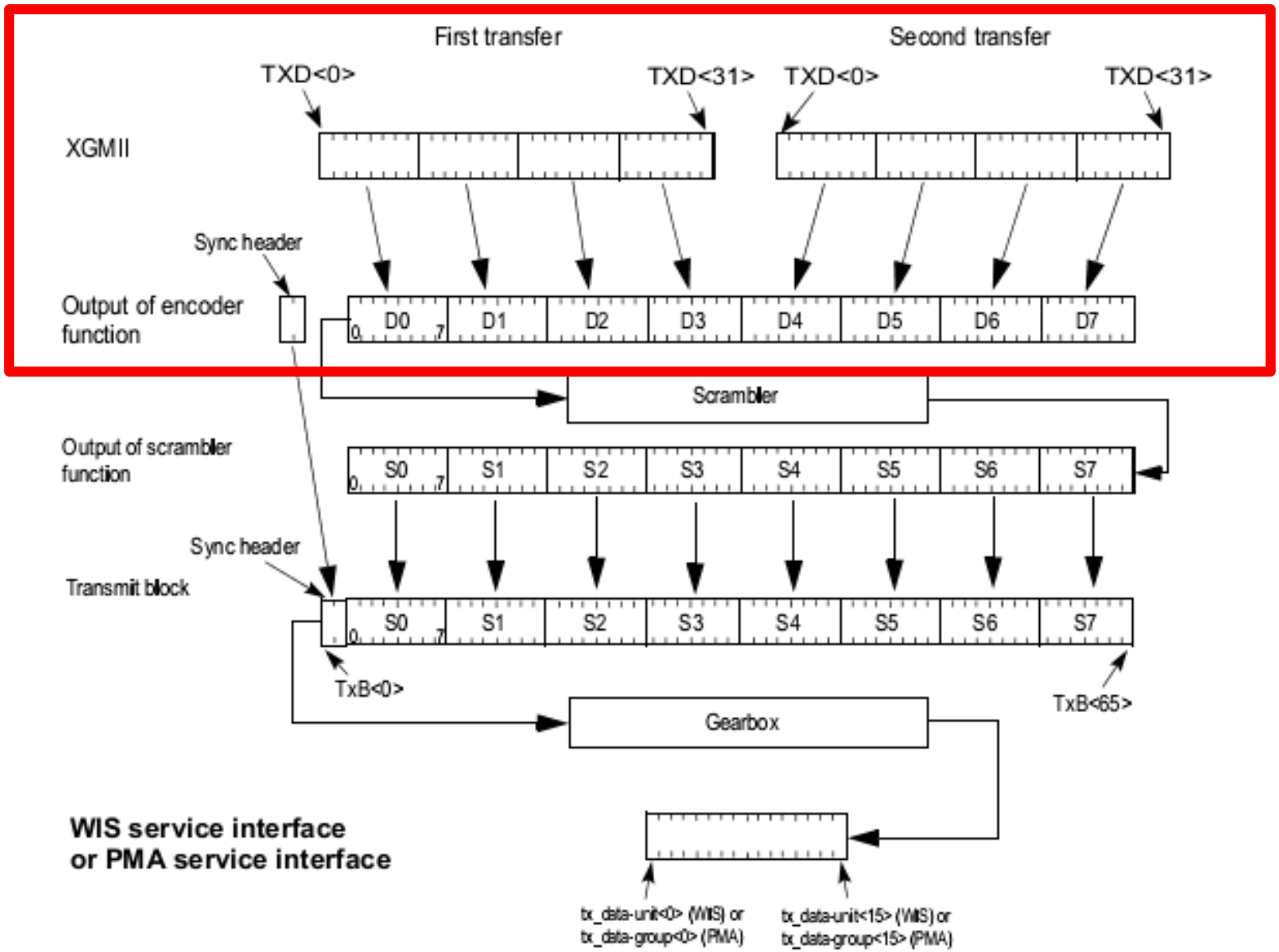


Figure 49–5—PCS Transmit bit ordering

64b/66b line encoding process (II)

- We are only interested in functions in red box marked on the previous slides (Tx direction)
- Two consecutive XGMII transfers (32 bits + 32 bits of data) are aggregated into a 64-bit data vector.
- That data vector is then used to generate a 2-bit synchronization header (Sync header for short), prepending the actual 64-bit data vector
  - Content of Sync header depends on data carried in 64-bit vector received in two consecutive XGMII transfers
- In this way, a 66-bit encoded data vector is created.
- The process in Rx direction is essentially an inverse of the process shown in slide 3.

So what's with this Sync header?

- Sync header is used primarily to facilitate alignment to data vectors on the receiving side and synchronization to the received bit stream
- Sync header is also used in various Ethernet PHYs to estimate the BER by looking for Sync header violations.
  - see 76.3.3.4 BER monitor in 10G-EPON for an example
- Sync headers, combined with the specific start, terminate and IDLE control codes and large Hamming distance between various block type fields, give 64b/66b coding very high MTTFPA* as long as the errors are randomly distributed.

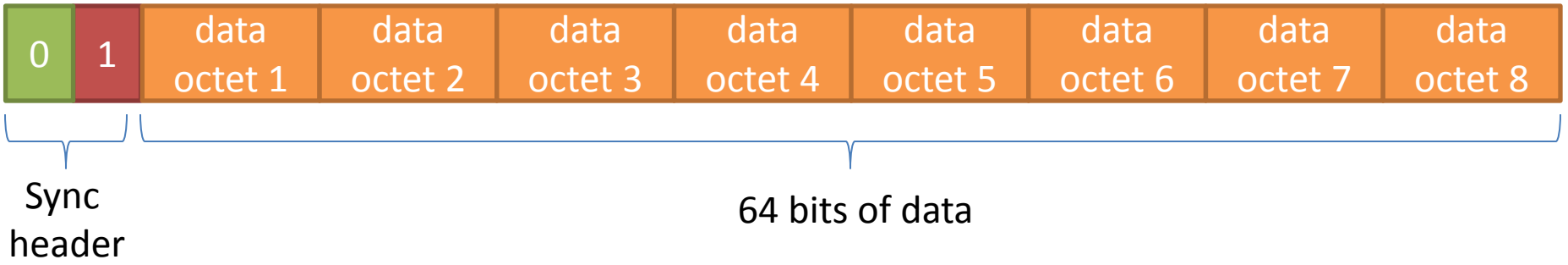
* Mean Time To False Packet Acceptance

Vector types

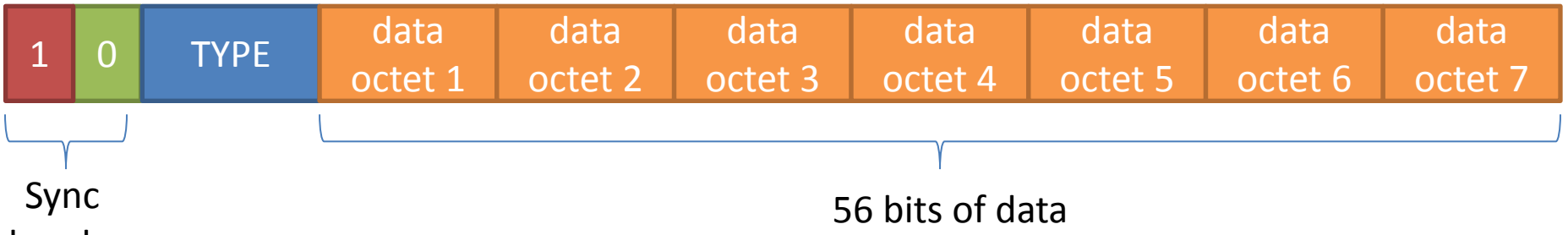
- There are two vector types possible
  - Sync header = 0b01: this 66-bit vector carries data
  - Sync header = 0b10: this 66-bit vector carries a mix of data and control characters, or control characters only
- Sync headers with values of 0b00 and 0b11 are considered invalid and treated as error on the receiving side. A 66-bit vector with invalid Sync header is replaced with an Error (/E/) character on XGMII.

More on valid vector types

- Data vector (Sync header = 0b01)



- Control vector (Sync header = 0b10)



  - TYPE field indicates the type of the control vector, and its internal structure
  - Next slide shows available types of control vectors

Control Symbols (I)

- Start /S/ indicates the start of packet
  - occurs only on positions 0 or 4 within aggregated 64-bit vector received from XGMII
  - receipt of /S/ on any other lane indicates error
- Terminate /T/ indicates the end of packet
  - occurs on any position within aggregated 64-bit vector received from XGMII
  - followed by /S/ or /I/
- Error /E/ indicates error in data stream
  - occurs on any position within aggregated 64-bit vector received from XGMII
  - used to relay Error indication across XGMII and then onto the link peer station

Control Symbols (II)

- Ordered Set /Q/
  - used to send control and status information (e.g., a remote fault and local fault status) across the link to the link peer station
  - consist of a control character and three data characters
  - always begin on the first octet of the XGMII transfer vector
  - may be deleted / inserted by PCS to control adapt between clock rates.
    Such deletion only occurs when two consecutive Ordered Sets are received and only one of the two is deleted. Only IDLE Ordered Sets may be inserted for clock compensation. Other types of Ordered Sets are not deleted for the purpose of clock compensation .

64b/66b vector types

Input Data	Sync	Block Payload							
Bit Position: / Data Block Format:	0 1	2							65
$D_0\,D_1\,D_2\,D_3/D_4\,D_5\,D_6\,D_7$	01	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
Control Block Formats:		Block Type Field							
$C_0\,C_1\,C_2\,C_3/C_4\,C_5\,C_6\,C_7$	10	0x1e	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
$C_0\,C_1\,C_2\,C_3/O_4\,D_5\,D_6\,D_7$	10	0x2d	C_0	C_1	C_2	C_3	O_4	D_5	D_6	D_7
$C_0\,C_1\,C_2\,C_3/S_4\,D_5\,D_6\,D_7$	10	0x33	C_0	C_1	C_2	C_3		D_5	D_6	D_7
$O_0\,D_1\,D_2\,D_3/S_4\,D_5\,D_6\,D_7$	10	0x66	D_1	D_2	D_3	O_0		D_5	D_6	D_7
$O_0\,D_1\,D_2\,D_3/O_4\,D_5\,D_6\,D_7$	10	0x55	D_1	D_2	D_3	O_0	O_4	D_5	D_6	D_7
$S_0\,D_1\,D_2\,D_3/D_4\,D_5\,D_6\,D_7$	10	0x78	D_1	D_2	D_3	D_4		D_5	D_6	D_7
$O_0\,D_1\,D_2\,D_3/C_4\,C_5\,C_6\,C_7$	10	0x4b	D_1	D_2	D_3	O_0	C_4	C_5	C_6	C_7
$T_0\,C_1\,C_2\,C_3/C_4\,C_5\,C_6\,C_7$	10	0x87		C_1	C_2	C_3	C_4	C_5	C_6	C_7
$D_0\,T_1\,C_2\,C_3/C_4\,C_5\,C_6\,C_7$	10	0x99	D_0		C_2	C_3	C_4	C_5	C_6	C_7
$D_0\,D_1\,T_2\,C_3/C_4\,C_5\,C_6\,C_7$	10	0xaa	D_0	D_1		C_3	C_4	C_5	C_6	C_7
$D_0\,D_1\,D_2\,T_3/C_4\,C_5\,C_6\,C_7$	10	0xb4	D_0	D_1	D_2		C_4	C_5	C_6	C_7
$D_0\,D_1\,D_2\,D_3/T_4\,C_5\,C_6\,C_7$	10	0xcc	D_0	D_1	D_2	D_3		C_5	C_6	C_7
$D_0\,D_1\,D_2\,D_3/D_4\,T_5\,C_6\,C_7$	10	0xd2	D_0	D_1	D_2	D_3	D_4		C_6	C_7
$D_0\,D_1\,D_2\,D_3/D_4\,D_5\,T_6\,C_7$	10	0xe1	D_0	D_1	D_2	D_3	D_4	D_5		C_7
$D_0\,D_1\,D_2\,D_3/D_4\,D_5\,D_6\,T_7$	10	0xff	D_0	D_1	D_2	D_3	D_4	D_5	D_6	

Figure 49–7—64B/66B block formats

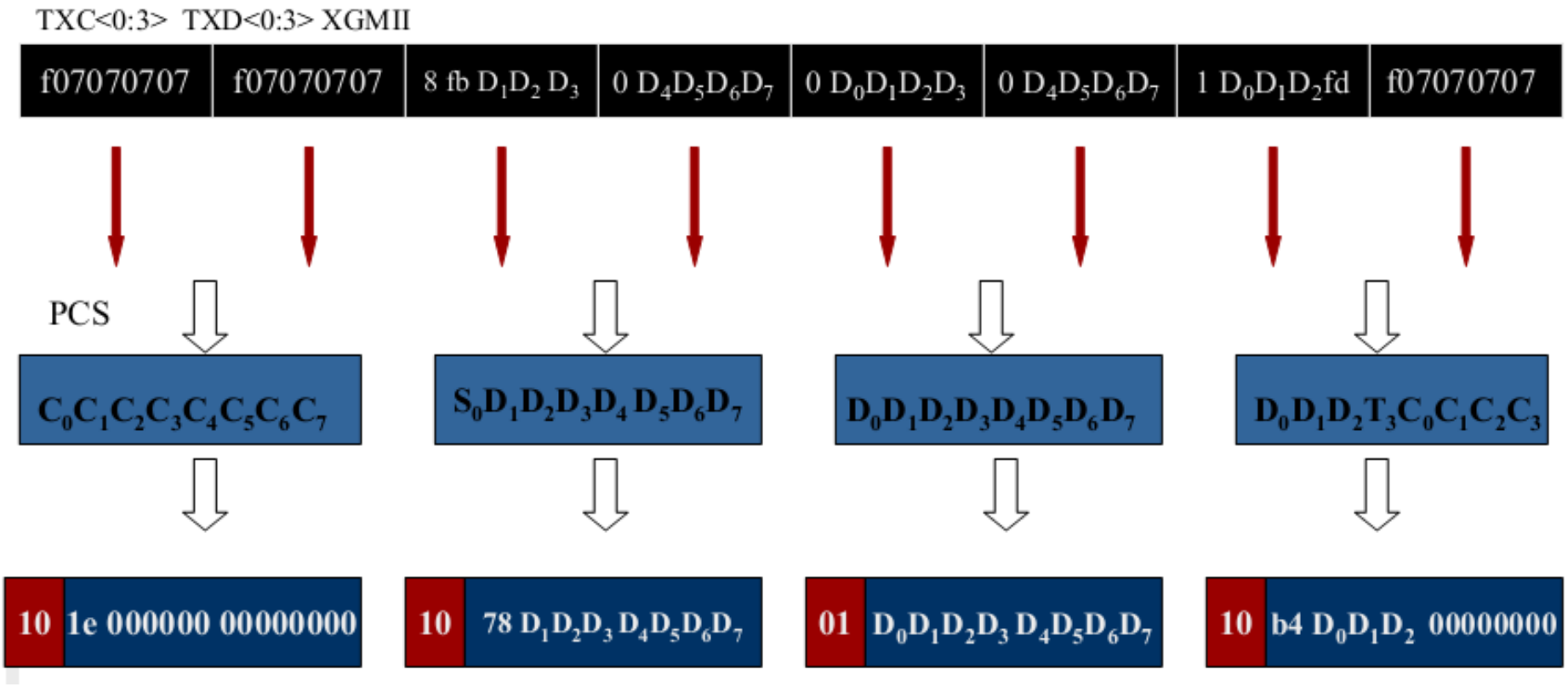
64b/66b control codes

Table 49–1—Control codes

Control character	Notation	XGMII Control Code	10GBASE-R Control Code	10GBASE-R O Code	8B/10B Code[a]
idle	/I/	0x07	0x00		K28.0 or K28.3 or K28.5
LPI	/LI/	0x06	0x06		K28.0 with D20.5 in one row or K28.3 or K28.5 with D20.5 in one row[b]
start	/S/	0xfb	Encoded by block type field		K27.7
terminate	/T/	0xfd	Encoded by block type field		K29.7
error	/E/	0xfe	0x1e		K30.7
Sequence ordered_set	/Q/	0x9c	Encoded by block type field plus O code	0x0	K28.4
reserved0	/R/[c]	0x1c	0x2d		K28.0
reserved1		0x3c	0x33		K28.1
reserved2	/A/	0x7c	0x4b		K28.3
reserved3	/K/	0xbc	0x55		K28.5
reserved4		0xdc	0x66		K28.6
reserved5		0xf7	0x78		K23.7
Signal ordered_set[d]	/Fsig/	0x5c	Encoded by block type field plus O code	0xF	K28.2

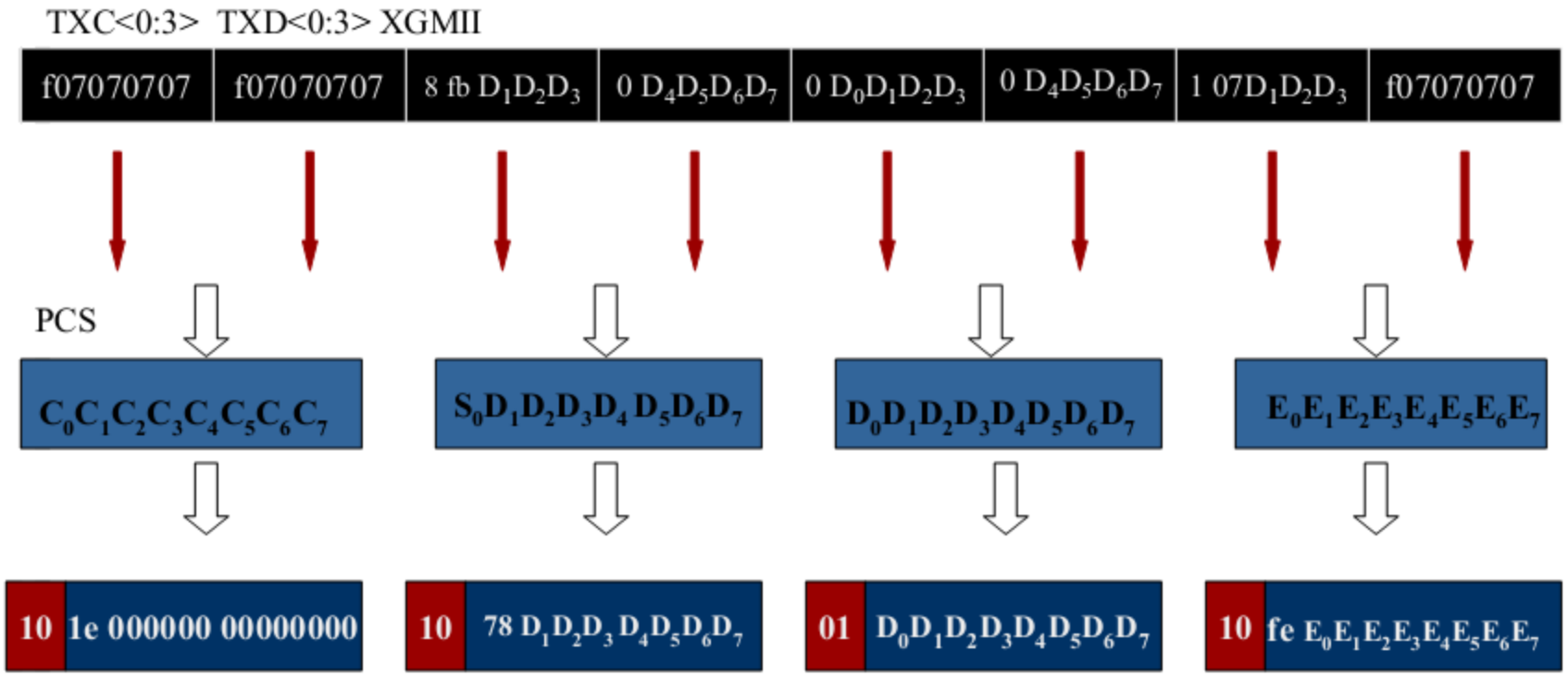
[a]For information only. The 8B/10B code is specified in Clause 36. Usage of the 8B/10B code for 10 Gb/s operation is specified in Clause 48.
[b]See 48.2.4.2.
[c]The codes for /A/, /K/, and /R/ are used on the XAUI interface to signal idle. They are not present on the XGMII when no errors have occurred, but certain bit errors cause the XGXS to send them on the XGMII.
[d]Reserved for INCITS T11 Fibre Channel use.

64b/66b encoding examples (I)



- Transmission of a valid code set is shown
- Short data frame is transmitted, with correct /S/ and /T/ symbols, followed by IDLEs

64b/66b encoding examples (II)



- Transmission of an invalid code set is shown
- /S/ symbol is transmitted correctly, and invalid XGMII symbols are detected and converted into /E/ characters

Summary & Suggestions (I)

- 10GBASE-R PCS structure, and specifically 64b/66b encoding / decoding processes are described in Clause 49
- 10G-EPON reuses these mechanisms through references. There are no additional Ordered Sets added by 10G-EPON.
- Unless changes to 64b/66b operation are required, we could reuse it through reference to Clause 49
  - Less work, less debugging, existing and tested mechanisms are already in place

Summary & Suggestions (I)

- Line code structure would then define the possible sizes of the PHY blocks
  - Multiples of 66-bit vectors would allow to avoid fragmentation of vectors across OFDM symbols
  - The line code choice also impacts FEC. Multiples of 66-bit or 65-bit vectors (to be selected*) need to be supported to avoid vector level jitter and fragmentation of vectors across FEC words.

Decision on line code impacts many choices we need for EPoC to start moving towards the draft.

** In 10G-EPON, Sync headers are truncated for FEC-coding to fit the maximum number of data vectors into a single FEC payload (see backup slide)*

Straw Poll

- EPoC PHY shall use the 64b/66b line encoding (as defined in IEEE Std 802.3-2012, Clause 49) within PCS.
- Yes:
- No:
- No opinion:

THANK YOU

FEC in 10G-EPON (encoding)

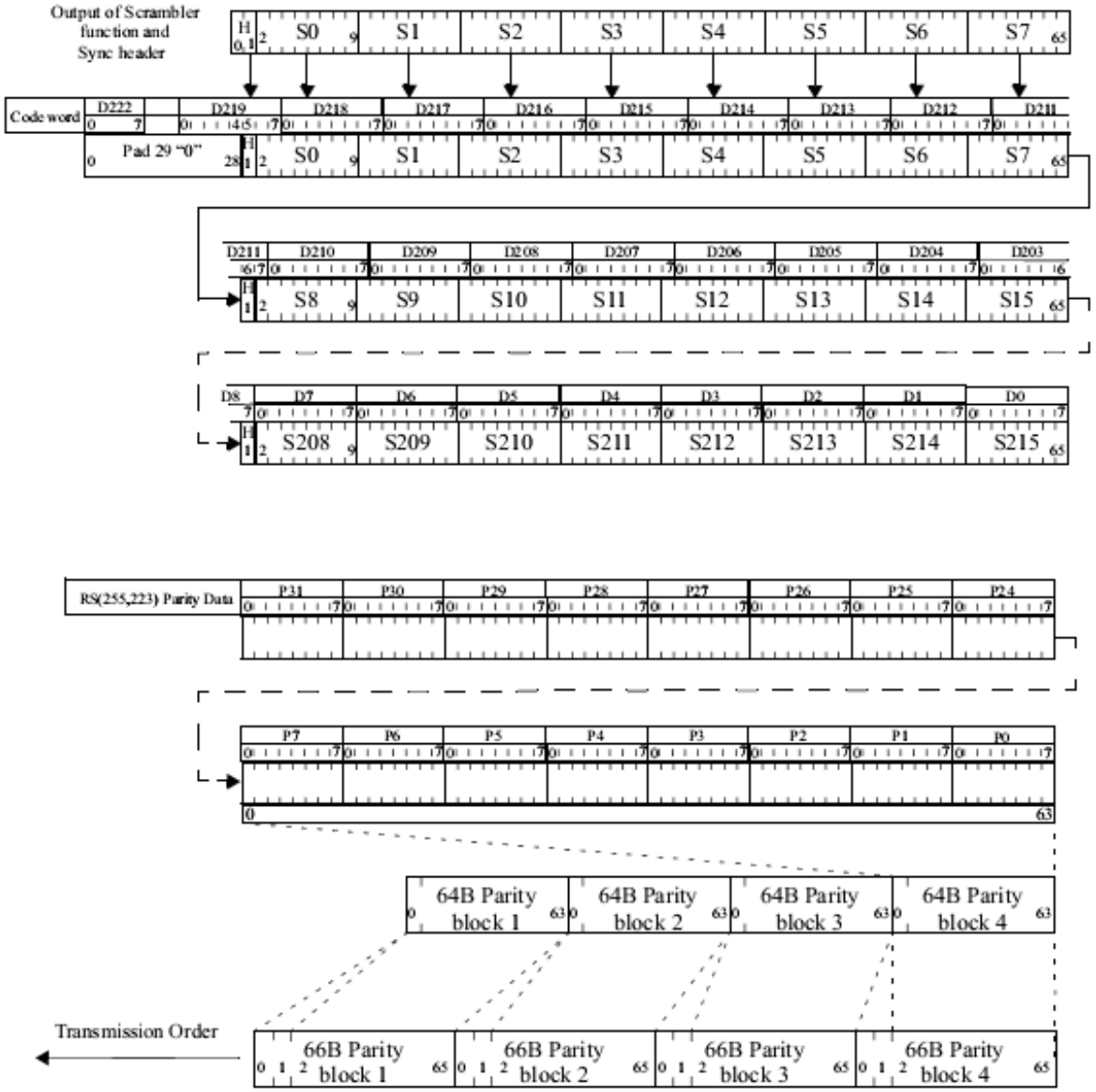


Figure 76–12—Bit ordering in FEC codeword generation