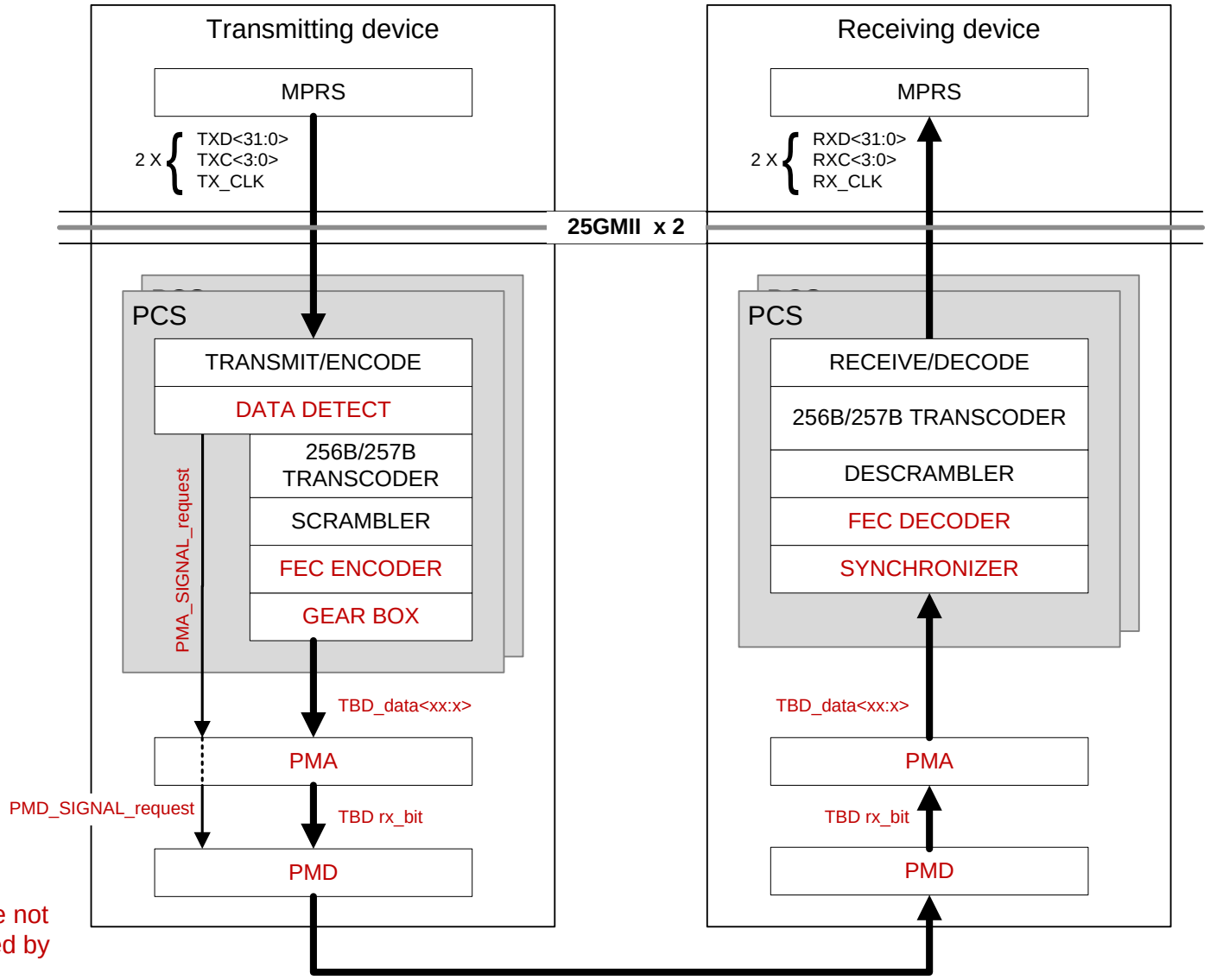
PCS
State Diagrams plus

Gao (Heaven) Bo (Huawei)
Glen Kramer (Broadcom Ltd)
Duane Remein (Huawei)

Motivation

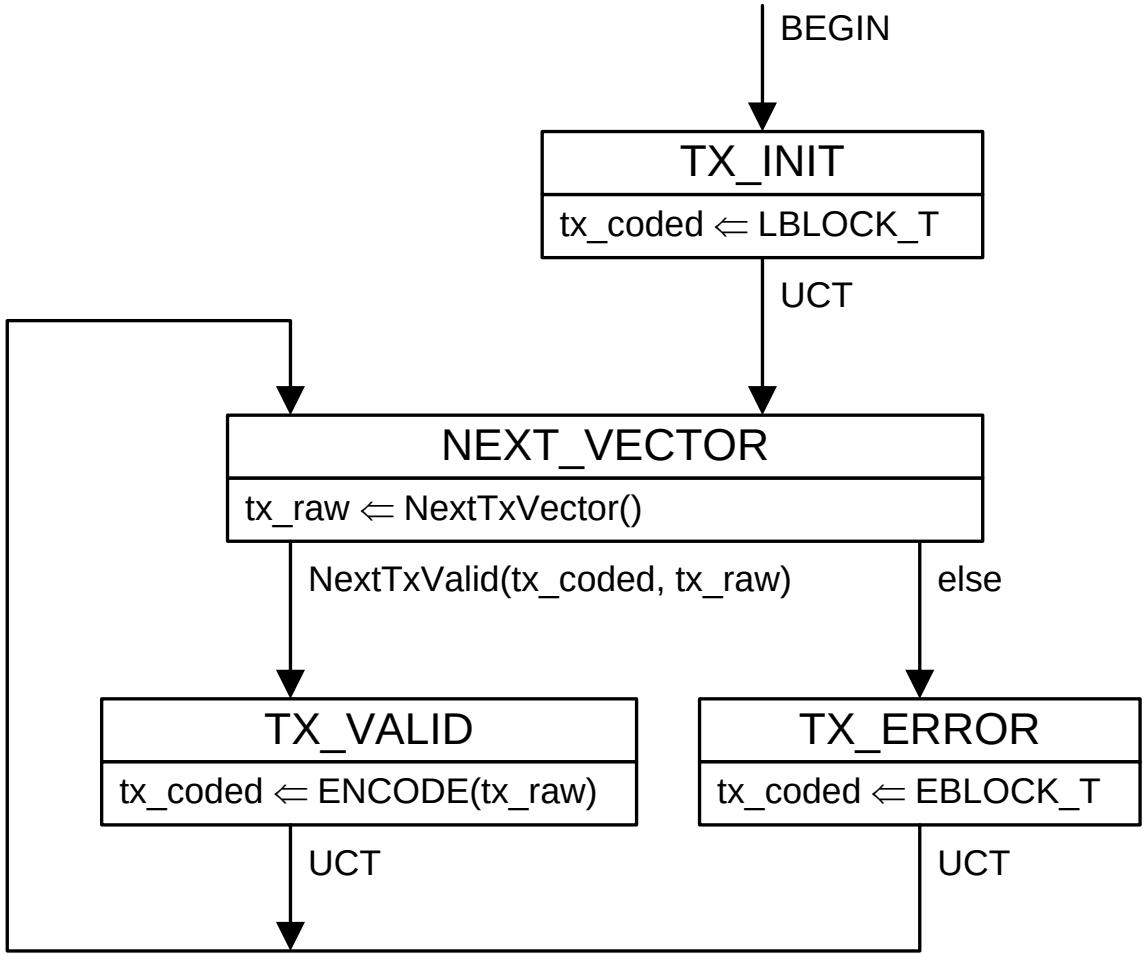
- 100G-EPON use of 25GMII is significantly different than previous uses
  - EQ splicing across multiple channels
  - Frame fragmentation
  - Inclusion of PARITY_PLACEHLDR and INTER_ENV_IDLE
- Previous uses referenced Cl 49 Transmit and Receive State Diagrams
  - Figure 49-16 & 49-17 (see back-up)
- New SDs are needed for 100G-EPON
  - Proposal based on new functions:
    - NextTxValid(prev, next)
    - NextRxValid(prev, next)
  - Renamed to Transmit/Encode SD and Receive/Decode SD to better reflect location in PCS

Location of State Diagram

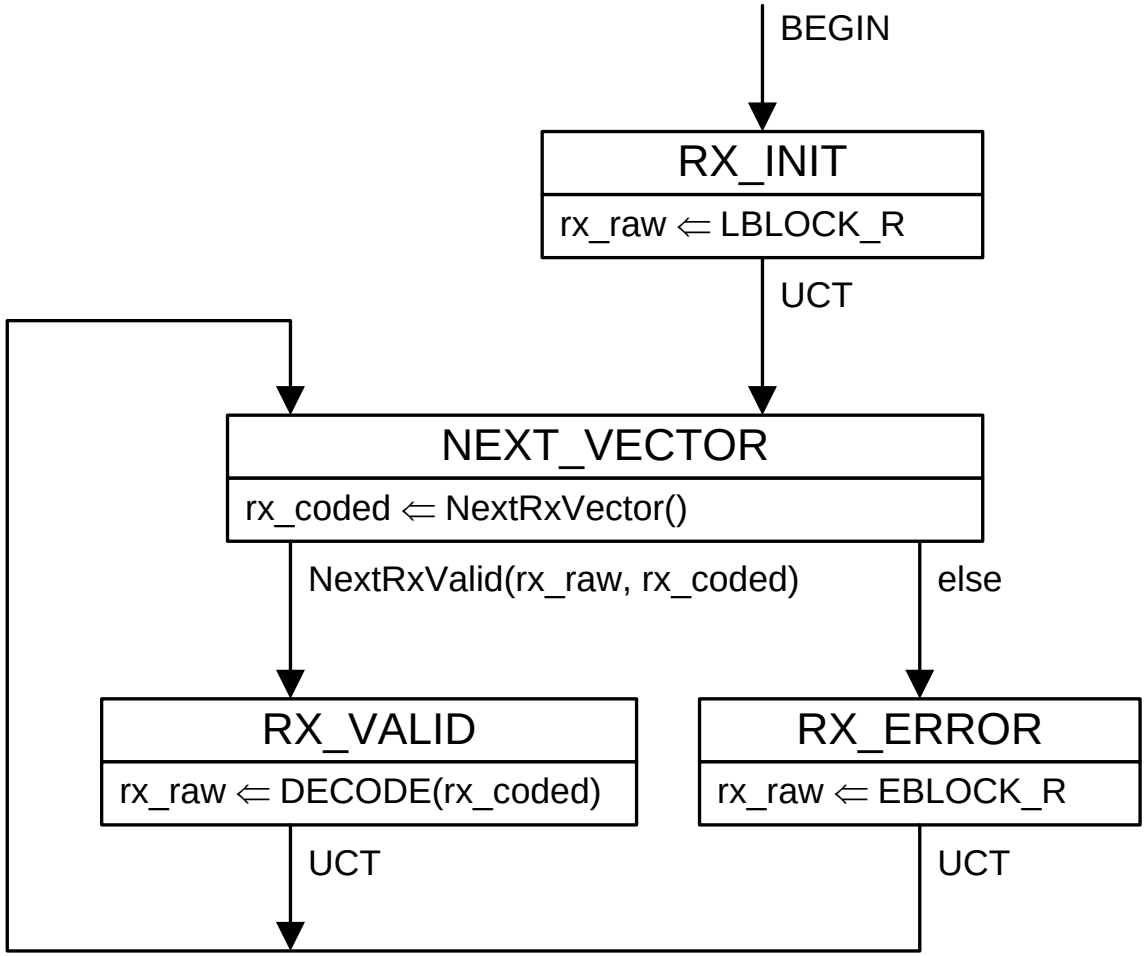


Blocks in red have not yet been discussed by the TF

Transmit/Encode SD



Receive/Decode SD



Definitions

EBLOCK_T - see 49.2.13.2.1

ENCODE(tx_raw) - see 49.2.13.2.3

LBLOCK_T - see 49.2.13.2.1

NextTxVector() – function which returns the next 72-bit vector from the 25GMII

tx_coded – see 49.2.13.2.2

tx_raw – see 49.2.13.2.2

INTER_ENV_IDLE (in 143.4.3.2)

TYPE: 72-bit vector

Value: 0xFF 08 08 08 08 08 08 08 08

The value of an EQ which represents idle space between envelopes

DECODE(rx_coded) - see Cl 49.2.13.2.3

EBLOCK_R - see 49.2.13.2.1

LBLOCK_R - see 49.2.13.2.1

NextRxVector() – function which returns the next 66-bit vector from the Descrambler

rx_coded – see 49.2.13.2.2

rx_raw – see 49.2.13.2.2

PARITY_PLACEHOLDER (in 143.4.3.2)

TYPE: 72-bit vector

Value: 0xFF 09 09 09 09 09 09 09 09

The value of an EQ which represents FEC Parity bits within a transmission

New Functions

- NextTxValid(prev_tx_coded, next_tx_raw)
  - This function returns a Boolean indicating whether the next_tx_raw vector is valid given the classification of the previously transmitted prev_tx_coded vector. The function returns the values according to the table on the next slides.
- NextRxValid(prev_rx_raw, next_rx_coded)
  - This function returns a Boolean indicating whether the next_rx_coded vector is valid given the classification of the previously received prev_rx_raw vector. The function returns the values according to the table on the next slides.

New Functions

- Table defining NextTxValid and NextRxValid

		next_tx_raw/next_rx_coded vector classification						
		IEI	S	D	T	I	P	other
prev_tx_coded/ prev_rx_raw vector classification	L	true	false	false	false	false	false	false
	IEI	true	true	false	false	false	true	false
	S	true	true	true	true	true	true	false
	D	true	true	true	true	true	true	false
	T	true	true	true	false	true	true	false
	I	true	true	true	false	true	true	false
	P	true	true	true	true	true	true	false
	other	true	true	true	true	true	true	false

Vector classifications

		Criteria for tx_raw/rx_raw vector	Criteria for tx_coded/rx_coded vector
Classification	**L**	rx_raw<71:0> = LBLOCK_R (see 49.2.13.2.1). This classification does not apply to tx_raw<71:0>.	tx_coded<65:0> = LBLOCK_T (see 49.2.13.2.1). This classification does not apply to rx_coded<65:0>.
	IEI	Vector composed of INTER_ENV_IDLE (see 143.4.3.2)	Vector composed of Inter-envelope idle (vector<1:0> = 10, vector<9:2> = 0x1E, and all control codes = 0x08).
	S	Vector beginning with a Start control code symbol (vector<7:0> = 0x80, vector<15:8> = 0xFB)	Vector comprised of a Start control code symbol (vector<1:0> = 10 and vector<9:2> = 0x78)
	D	Vector of all data bytes (vector<7:0> = 0x00)	Vector of all data bytes (vector<1:0> = 01)
	T	Vector which includes a Terminate control code symbol (vector<7:0> $\in$ {0xFF, 0x7F, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x01}, 1st control code octet = 0xFD, and all other control characters are valid)	Vector which includes a Terminate control code symbol (vector<1:0> = 10 and vector<9:2> $\in$ {0x87, 0x99, 0xAA, 0xB4, 0xCC, 0xD2, 0xE1, 0xFF}, and all control characters are valid)
	I	Vector composed of all Idle control code symbols (vector<7:0> = 0xFF and all other octets= 0x07)	Vector composed of all Idle control code symbols vector<1:0> = 10 and vector<65:2> = 0x00..00)
	P	Vector composed of PARITY_PLACEHLDR (see 143.4.3.2)	Vector composed of all Parity placeholder (vector<1:0> = 10, vector<9:2> = 0x1E, and all control codes = 0x09)
	E	rx_raw<71:0> = EBLOCK_R (see 49.2.13.2.1). This classification does not apply to tx_raw<71:0>.	tx_coded<65:0> = EBLOCK_T (see 49.2.13.2.1). This classification does not apply to rx_coded<65:0>.

25GBASE-PR Control codes

- This table replaces Table 49-1

Control character	Notation	25GMII control code	25GBASE-PR control code
idle	/I/	0x07	0x00
Inter-envelope idle	/IEI/	0x08	0x08
Parity placeholder	/P/	0x09	0x09
start	/S/	0xFB	Encoded by block type field
terminate	/T/	0xFD	Encoded by block type field
error	/E/	0xFE	0x1E

Control ch				
idle	/I/	0x07		0x00
Inter-envelope idle	/IEI/	0x08		0x08
Parity placeholder	/P/	0x09		0x09
start	/S/	0xFB		Encoded by block type field
terminate	/T/	0xFD		Encoded by block type field
error	/E/	0xFE		0x1E
Sequence ordered set	/Q/	0x9C	0x0	Encoded by block type 0x4B plus O code, control codes are set to 0x00

The proposed table's look and feel is similar to Table 82-1 for 40/100GBASE-R, which also replaced Table 49-1. (see back-up)

64B/66B block formats

- Use the control block types shown in Table 49-7, with the exception of the following Block Type Field values: 0x2d, 0x33, 0x66, 0x55, and 0x4b

64B/66B block formats

- Excluding Block Type Field values 0x2d, 0x33, 0x66, 0x55, and 0x4b result in the following table

Input Data	**Sync**	**Block Payload**							
Bit Position:	0 1	2							65
D0 D1 D2 D3/D4 D5 D6 D7	01	D0	D1	D2	D3	D4	D5	D6	D7
Control Block Formats:		**Block Type Field**							
C0 C1 C2 C3/C4 C5 C6 C7	10	0x1E	C0	C1	C2	C3	C4	C5	C6	C7
S0 D1 D2 D3/D4 D5 D6 D7	10	0x78	D1	D2	D3	D4	D5	D6	D7
T0 C1 C2 C3/C4 C5 C6 C7	10	0x87		C1	C2	C3	C4	C5	C6	C7
D0 T1 C2 C3/C4 C5 C6 C7	10	0x99	D0		C2	C3	C4	C5	C6	C7
D0 D1 T2 C3/C4 C5 C6 C7	10	0xAA	D0	D1		C3	C4	C5	C6	C7
D0 D1 D2 T3/C4 C5 C6 C7	10	0xB4	D0	D1	D2		C4	C5	C6	C7
D0 D1 D2 D3/T4 C5 C6 C7	10	0xCC	D0	D1	D2	D3		C5	C6	C7
D0 D1 D2 D3/D4 T5 C6 C7	10	0xD2	D0	D1	D2	D3	D4		C6	C7
D0 D1 D2 D3/D4 D5 T6 C7	10	0xE1	D0	D1	D2	D3	D4	D5		C7
D0 D1 D2 D3/D4 D5 D6 T7	10	0xFF	D0	D1	D2	D3	D4	D5	D6	

64B/66B block formats

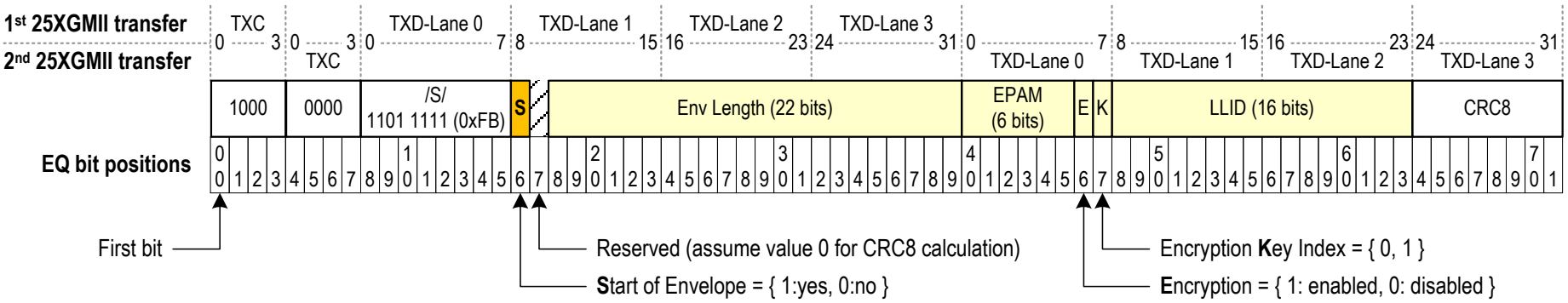
- This figure replaces Figure 49-7

Input Data	Sync	Block Payload							
Bit Position: / Data Block Format:	0 1	2							65
D0 D1 D2 D3/D4 D5 D6 D7	01	D0	D1	D2	D3	D4	D5	D6	D7
Control Block Formats:		**Block Type Field**							
C0 C1 C2 C3/C4 C5 C6 C7	10	0x1E	C0	C1	C2	C3	C4	C5	C6	C7
C0 C1 C2 C3/O4 D5 D6 D7	10	0x2D	C0	C1	C2	C3	O4	D5	D6	D7
O0 D1 D2 D3/O4 D5 D6 D7	10	0x55	D1	D2	D3	O0	O4	D5	D6	D7
S0 D1 D2 D3/D4 D5 D6 D7	10	0x78	D1	D2	D3	D4		D5	D6	D7
O0 D1 D2 D3/C4 C5 C6 C7	10	0x4B	D1	D2	D3	O0	C4	C5	C6	C7
T0 C1 C2 C3/C4 C5 C6 C7	10	0x87		C1	C2	C3	C4	C5	C6	C7
D0 T1 C2 C3/C4 C5 C6 C7	10	0x99	D0		C2	C3	C4	C5	C6	C7
D0 D1 T2 C3/C4 C5 C6 C7	10	0xAA	D0	D1		C3	C4	C5	C6	C7
D0 D1 D2 T3/C4 C5 C6 C7	10	0xB4	D0	D1	D2		C4	C5	C6	C7
D0 D1 D2 D3/T4 C5 C6 C7	10	0xCC	D0	D1	D2	D3		C5	C6	C7
D0 D1 D2 D3/D4 T5 C6 C7	10	0xD2	D0	D1	D2	D3	D4		C6	C7
D0 D1 D2 D3/D4 D5 T6 C7	10	0xE1	D0	D1	D2	D3	D4	D5		C7
D0 D1 D2 D3/D4 D5 D6 T7	10	0xFF	D0	D1	D2	D3	D4	D5	D6	

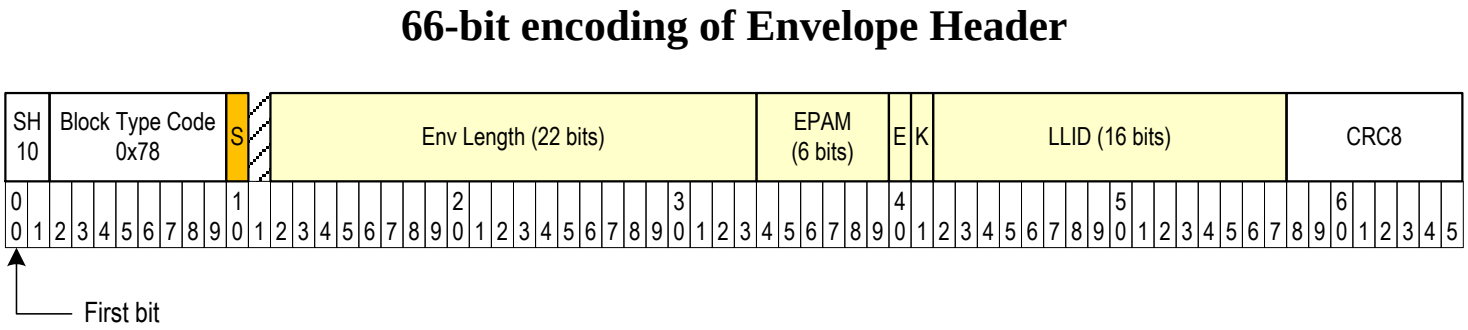
The proposed figures look and feel is similar to Figure 82-5 for 40/100GBASE-R, which also replaced Figure 49-7. (see back-up)

Envelope Header Encoding

Envelope Header (tx_raw/rx_raw, 72-bit vector)

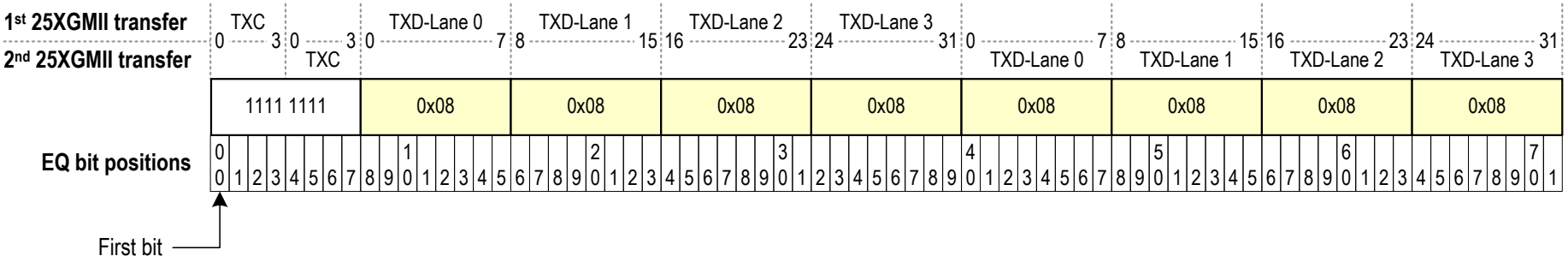


Encoded Envelope Header (tx_coded/rx_coded, 66-bit vector)

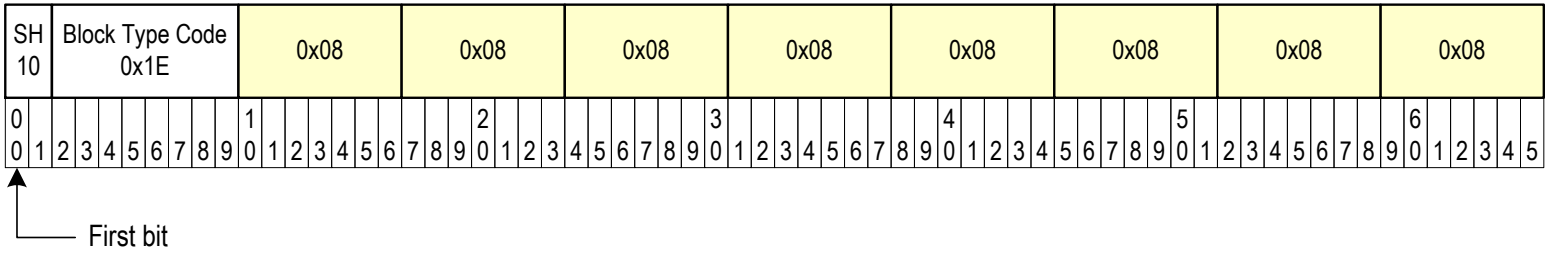


INT_ENV_IDLE Encoding

- **INTER_ENV_IDLE (tx_raw/rx_raw, 72-bit vector)**

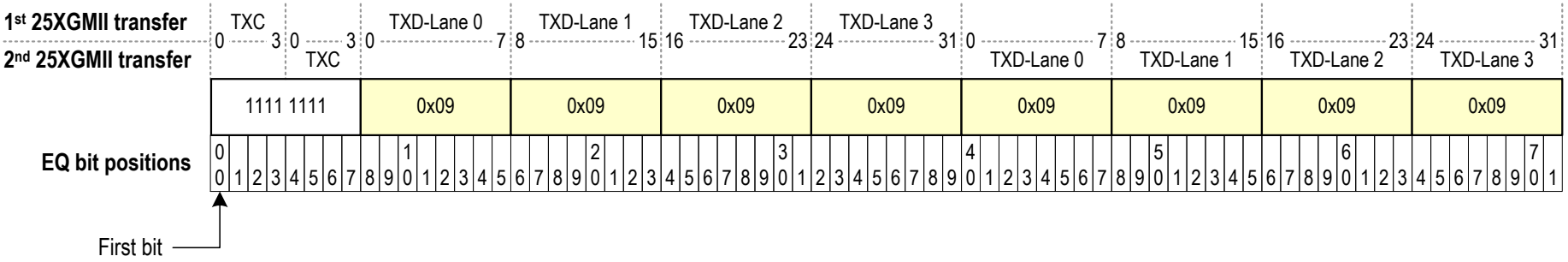


- **Encoded INTER_ENV_IDLE (tx_coded/rx_coded, 66-bit vector)**

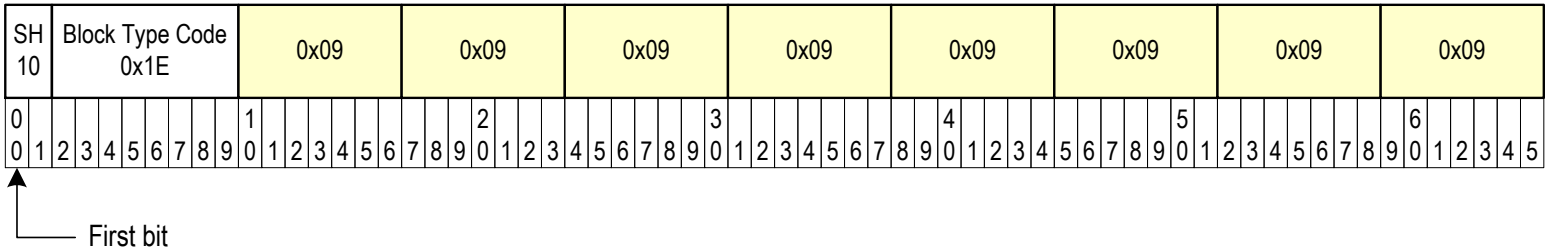


PARITY_PLACEHOLDER Encoding

- **PARITY_PLACEHOLDER (tx_raw/rx_raw, 72-bit vector)**

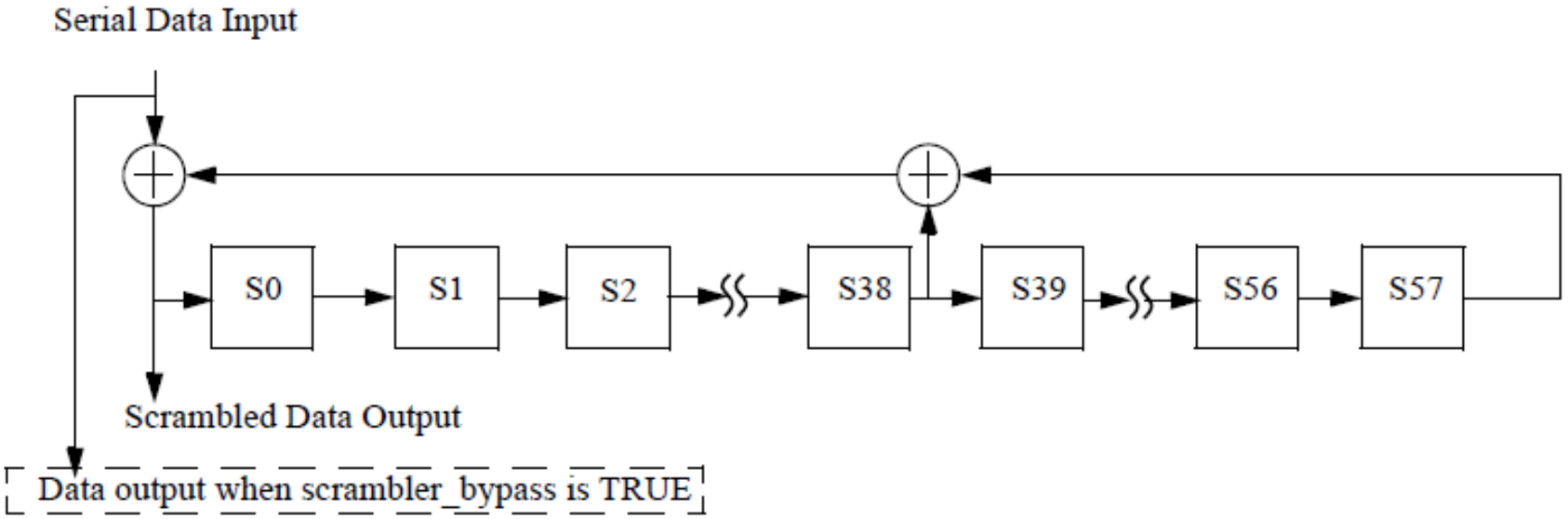


- **Encoded PARITY_PLACEHOLDER (tx_coded/rx_coded, 66-bit vector)**



Scrambler/Descrambler

- The self–synchronous scrambler defined in 49.2.6 is used in various PCS layers (Cl 49, 52, 74, 76 & 82)
- Is there any reason not to adopt this scrambler?



NOTE—Scrambler_bypass is only required to support EEE capability.

Figure 49–8—Scrambler

The Descrambler from 49-2.10 and Figure 49-10 included in back-up

Thank You

Motion

Move to adopt the figures, tables, and definitions on slides 3-11 of remein_3ca_1b_0118.pdf and the Scramble/Descrambler definitions per 49.2.6/49.2.10

Moved:

Second:

For:

Against:

Abstain:

Motion Technical (≥ 75)

Motion Passed/Failed

Table 49-1 Control codes

Table 49–1—Control codes

Control character	Notation	XGMII Control	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 rowb
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.

Table 82-1 Control codes

Table 82–1—Control codes

Control character	Notation	XLGMII/ CGMII control code	40/100GBASE-R O code	40GBASE-R and 100GBASE-R control code
idle	/I/	0x07		0x00
LPI	/LI/	0x06		0x06
start	/S/	0xFB		Encoded by block type field
terminate	/T/	0xFD		Encoded by block type field
error	/E/	0xFE		0x1E
Sequence ordered set	/Q/	0x9C	0x0	Encoded by block type 0x4B plus O code, control codes are set to 0x00
Signal ordered set[a]	/Fsig/	0x5C	0xF	Encoded by block type 0x4B plus O code, control codes are set to 0x00

[a]Reserved for INCITS T11 Fibre Channel use.

Figure 49-7 64B/66B block formats

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

Figure 82-5 64B/66B block formats

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
$S_0\,D_1\,D_2\,D_3\,D_4\,D_5\,D_6\,D_7$	10	0x78	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$O_0\,D_1\,D_2\,D_3\,Z_4\,Z_5\,Z_6\,Z_7$	10	0x4B	D_1	D_2	D_3	O_0	0x000_0000		
$T_0\,C_1\,C_2\,C_3\,C_4\,C_5\,C_6\,C_7$	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 82–5—64B/66B block formats

Descrambler

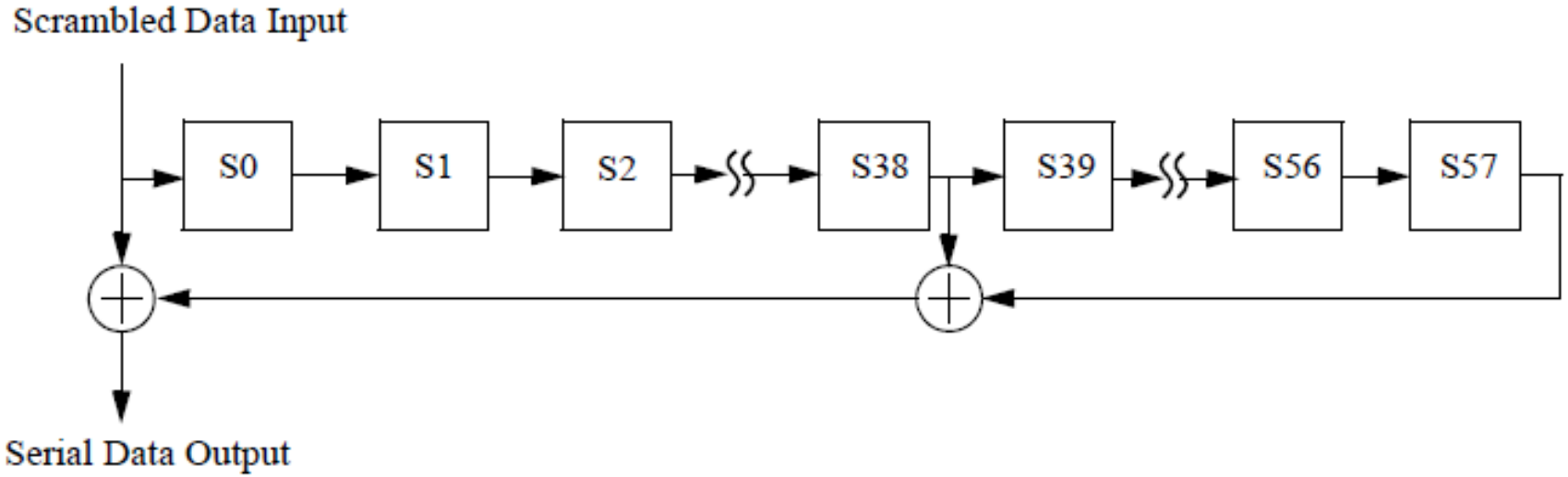


Figure 49–10—Descrambler

Figure 49-16

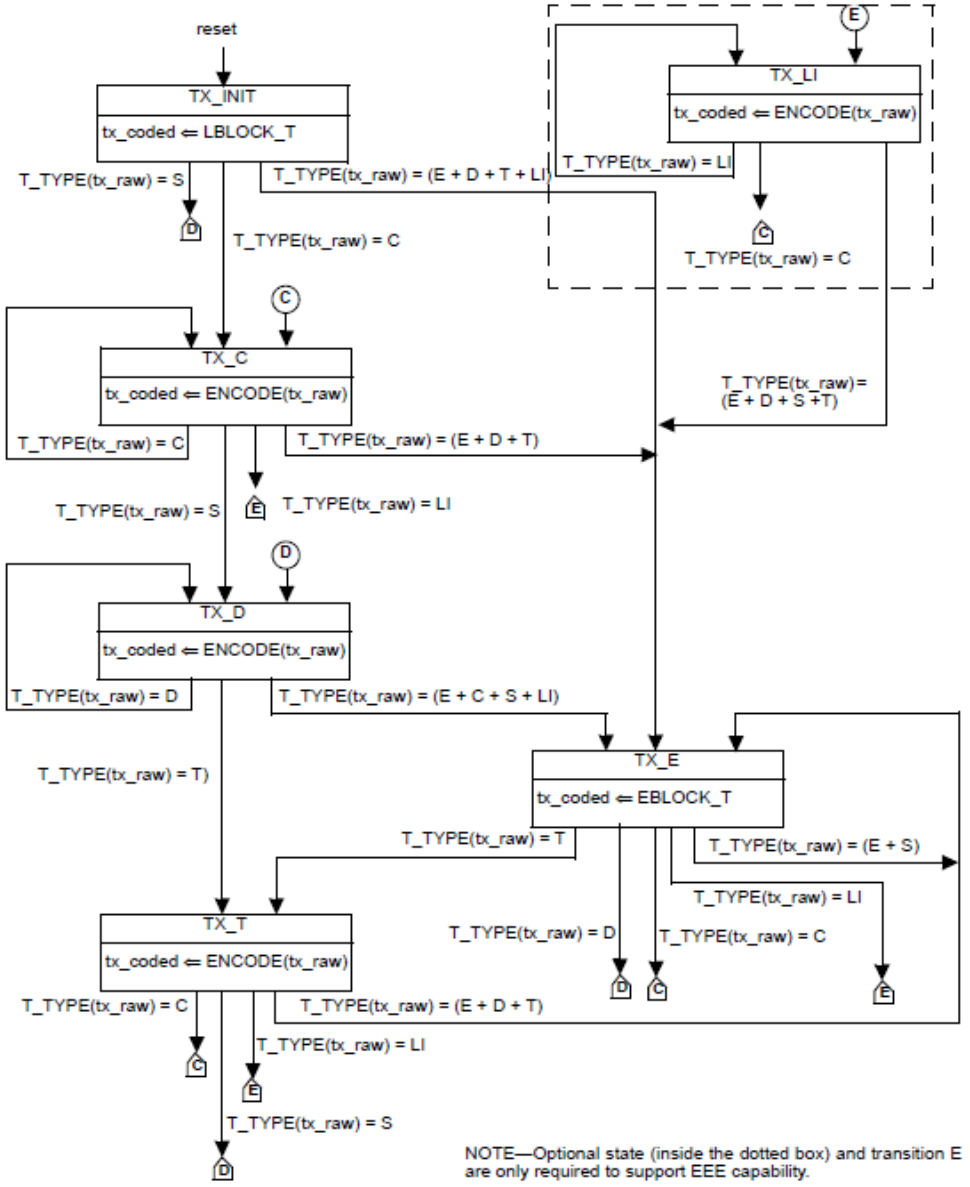
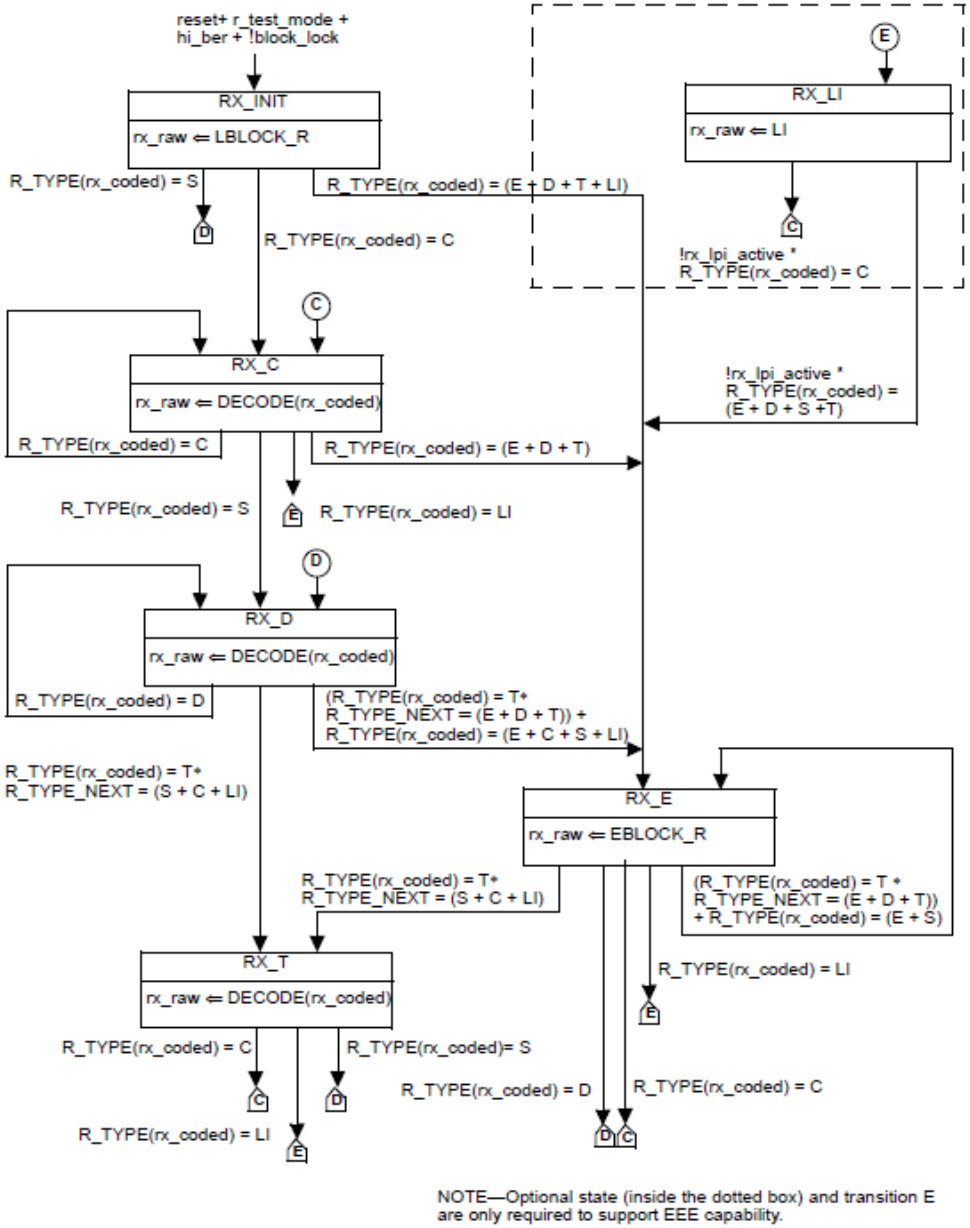


Figure 49–16—Transmit state diagram

Figure 49-17



NOTE—Optional state (inside the dotted box) and transition E are only required to support EEE capability.

Figure 49–17—Receive state diagram