Introduction to 10 Gigabit 64b/66b (Clause 49)

Sowmya S. Luckoor

Date: October 22, 2001

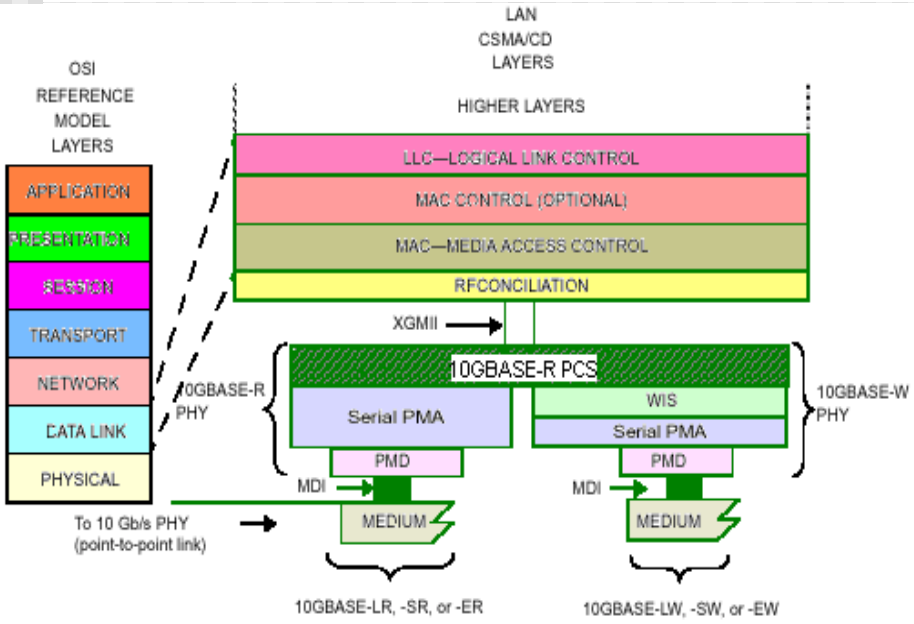
Main Topics of Discussion

- Purpose of 64b/66b
- Brief outline of 64b/66b structure
- 64b/66b PCS Process
- Data and Control Characters
- Scrambler Principles and Derivations
- Gearbox Principles
- State Diagrams of 64b/66b

What is the purpose of 64b/66b?

- Cost Effective
- Supports New Applications and Data
- Alternate to 8b/10b structure
- Complexity of hardware reduced
  - 10 => 10.3 vs. 10 => 12.5

Sublayer Diagram



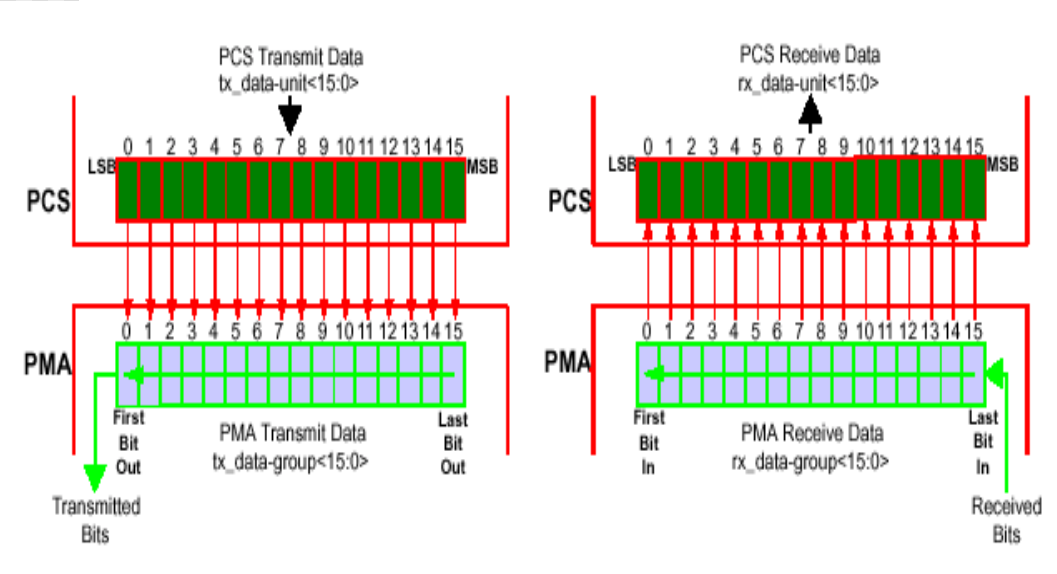
MDI=MEDIUM DEPENDENT INTERFACE
PCS=PHYSICAL CODING SUBLAYER
PHY=PHYSICAL LAYER DEVICE
PMA=PHYSICAL MEDIUM ATTACHMENT
PMD=PHYSICAL MEDIUM DEPENDENT
WIS=WAN INTERFACE SUBLAYER
XGMII=10 GIGABIT MEDIA INDEPENDENT INTERFACE

PMD TYPES:
Medium:
E = PMD FOR FIBER—1550 nm WAVELENGTH
L = PMD FOR FIBER—1310 nm WAVELENGTH
S = PMD FOR FIBER—850 nm WAVELENGTH
Encoding:
R = 64B/66B ENCODED WITHOUT WIS
W = 64B/66B ENCODED WITH WIS

NOTE—The PMD sublayers are mutually independent.

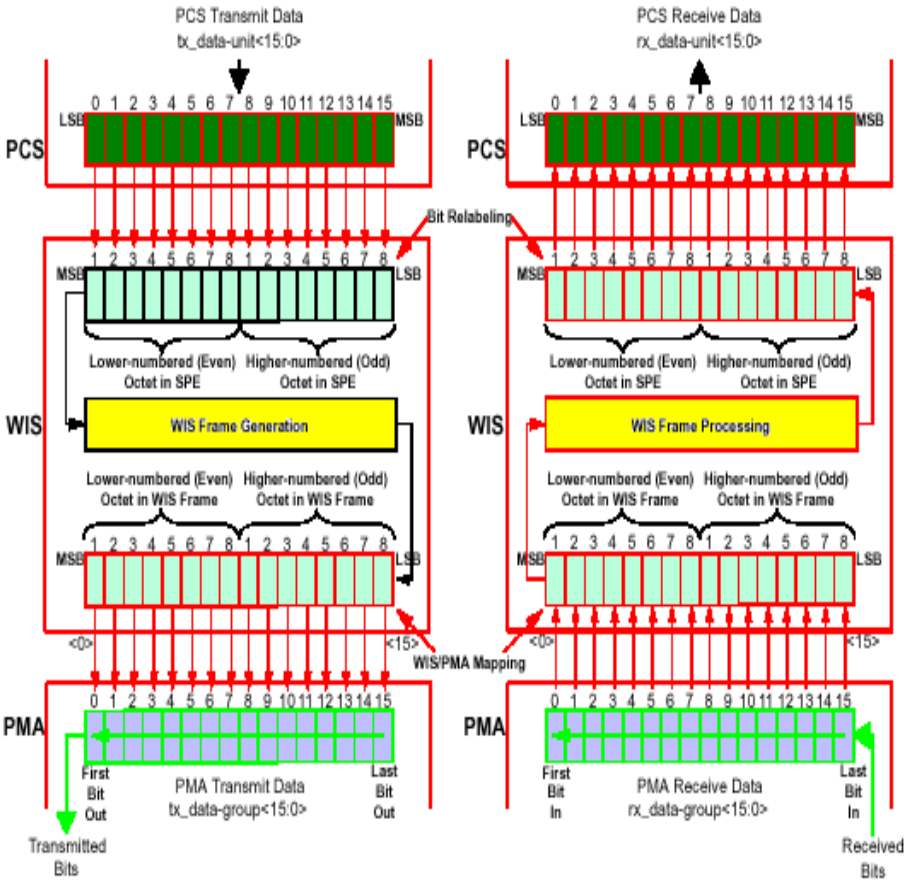
- 10GBASE-R PCS provides services to XGMII:
  - Encodes/Decodes 8 XGMII data octets to/from 66 bit blocks
  - Transfers encoded data to/from PMA in 16 bit transfers.
  - If connected to WAN PMD, inserts/deletes idles due to rate difference between MAC and PMD
  - Determines when link available, therefore informing management entity via MDIO when PHY is ready to be used.

Bit Ordering for 10GBASE-LAN



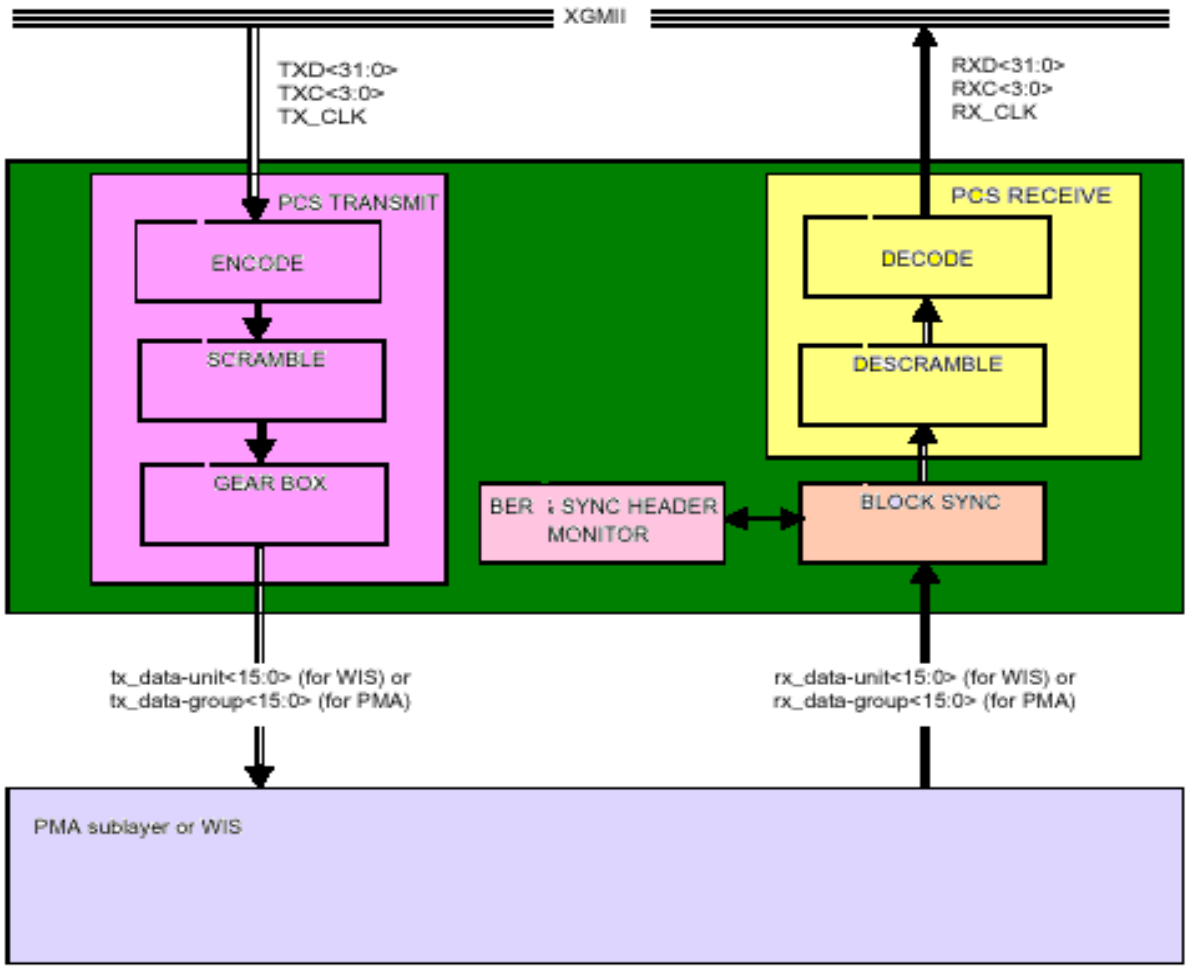
- PMA Functions:
  - Mapping of 16-bit data stream to from the PMA Service Interface via PCS, WIS or underlying PMD
  - 16-bit data serialized/deserialized for transmission to PMD
  - Recovery of clock from received data stream
  - Data Loopback at PMA Service Interface
  - Nominal Rate for PMA Service Interface = 644.53 Mtransfer/s =10 GB

Bit Ordering for 10GBASE-W



- Follows SONET/SDH Standard of labeling bits from MSB to LSB compared to conventional Ethernet standards
- Nominal rate of WIS Service Interface = 599.04 Mtransfers/s <10GB

Functional Block Diagram for 64b/66b



Functions within the PCS

- PCS Transmit (Normal Mode and Jitter Test Mode)
- Block (Frame) Synchronization
- PCS Receive
- BER Monitor Processes (assuming 10GBASE-(LR, SR, ER) or 10GBASE-(LW, SW, EW)
- Maps packets between XGMII format and PMA service interface format

PCS Transmit Process

- Transmit channel in normal mode:
  - Blocks generated continuously based upon TXD<31:0> and TXC<3:0> signals on XGMII
  - 66 bit blocks are packed by gearbox into 16 bit data units and sent to PMA or WIS via PMA_UNITDATA.REQUEST or WIS_UNITDATA.REQUEST, respectively.
  - If WIS present, adapts rates between XGMII and WIS by deleting IDLE characters when necessary.
- Transmit channel in jitter test mode:
  - Jitter test pattern packed into transmit data units, which are sent to PMA Service Interface.

PCS Receive Process

- Continuously accepts blocks once synchronization is complete
- Monitors blocks to generate RXD<31:0> and RXC<3:0> on XGMII
- When WIS is present, PCS receive process adapts between WIS and XGMII data rates by inserting IDLE characters

PCS Synchronization Process

- Continuously monitors PMA or WIS SIGNAL_DETECT. When SIGNAL_DETECT indicates OK, PCS SYNC process accepts data units and attempts to attain sync.
- Frame synchronization based on 2-bit sync header
- SYNC_STATUS flag set to indicate that PCS has received synchronization.

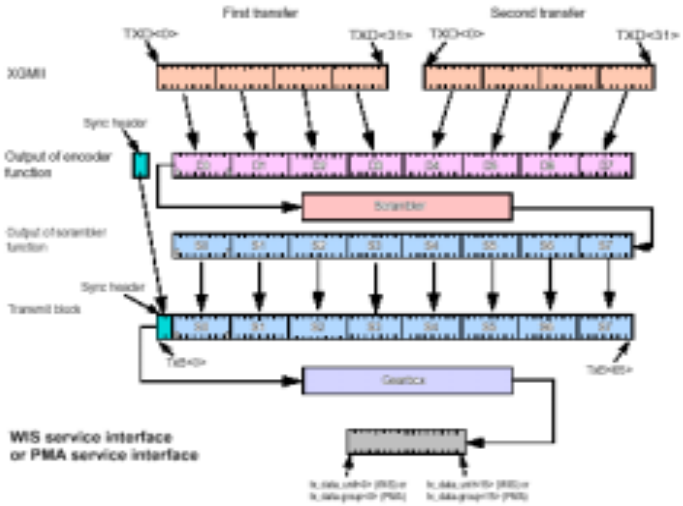
Bit Error Rate(BER) Monitor Process

- When synchronization is complete, signal quality monitoring is done by the BER Monitor Process.
- Asserting hi_ber occurs if errors are detected, therefore stopping acceptance of incoming blocks.
- When sync_status is asserted and hi_ber is deasserted, the PCS Received Process can accept frames.

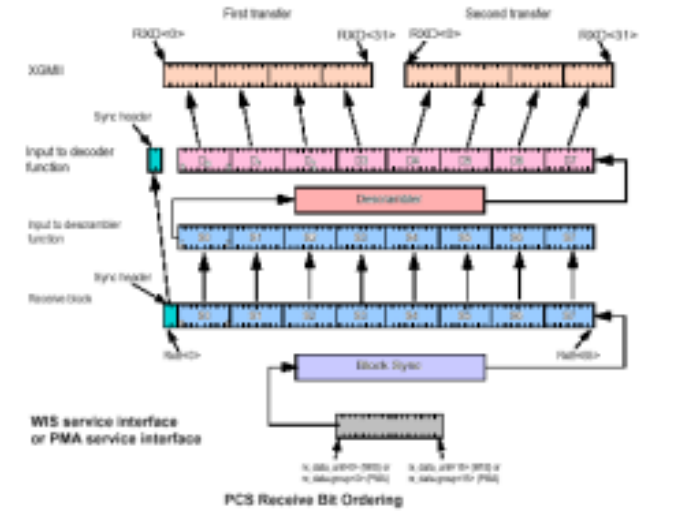
64b/66b Transmission Code

- Improves transmission characteristics of information transferred across link to support control and data characters.
- Transmission encoding ensures clock recovery is possible at receiver as well as detection of invalid codes when transmitting/receiving.
- Bit errors that form valid blocks that are not detected by transmission encoding, are detected by the frame's CRC.
- Block alignment is achieved by the synchronization headers.

Notation Conventions



PCS Transmit Bit Ordering

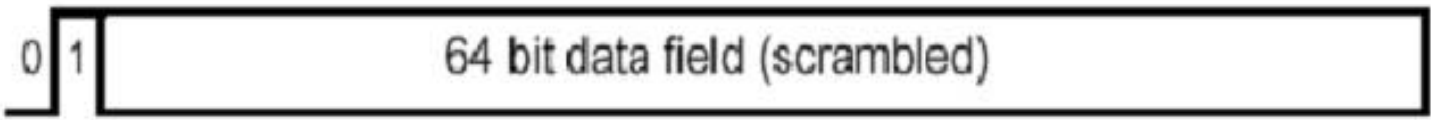


PCS Receive Bit Ordering

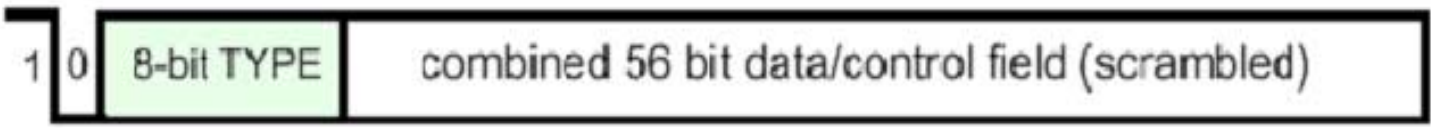
- 64b/66b encodes 8 data octets and or control codes into a block
- Blocks containing control codes also have type fields
- Data Octets D0-D7
- Control Octets C0-C7
  - Start S0 or S4
  - Terminate T0-T7
  - Ordered_Set O0 or O4

Code Process

Data Codewords have "01" sync preamble



Mixed Data/Control frames are identified with a "10" sync preamble. Both the coded 56-bit payload and BLOCK TYPE field are scrambled



00, 11 preambles are considered code errors and cause the packets to be invalidated by forcing an error (E) symbol on the XGMII output

Control Characters

- Start /S/ indicates start of packet
  - Occurs only on block 0 or 4
  - Receipt on any other lane indicates error
- Terminate /T/ indicates end of packet
  - Occurs on any octet
  - Needs to be followed by Idle or Start
- Ordered Set /Q/
  - Send control and status information such as remote fault and local fault status
  - Consist of a control character and three data characters
  - Always begin on the first octet of the XGMII
- Error /E/ indicates an error

Code Groups

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:		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	

64B/66B Block Formats

- Valid code groups are given on the left most side of the table.
- If a code group not present in the table occurs, error will be detected

Control Code Table

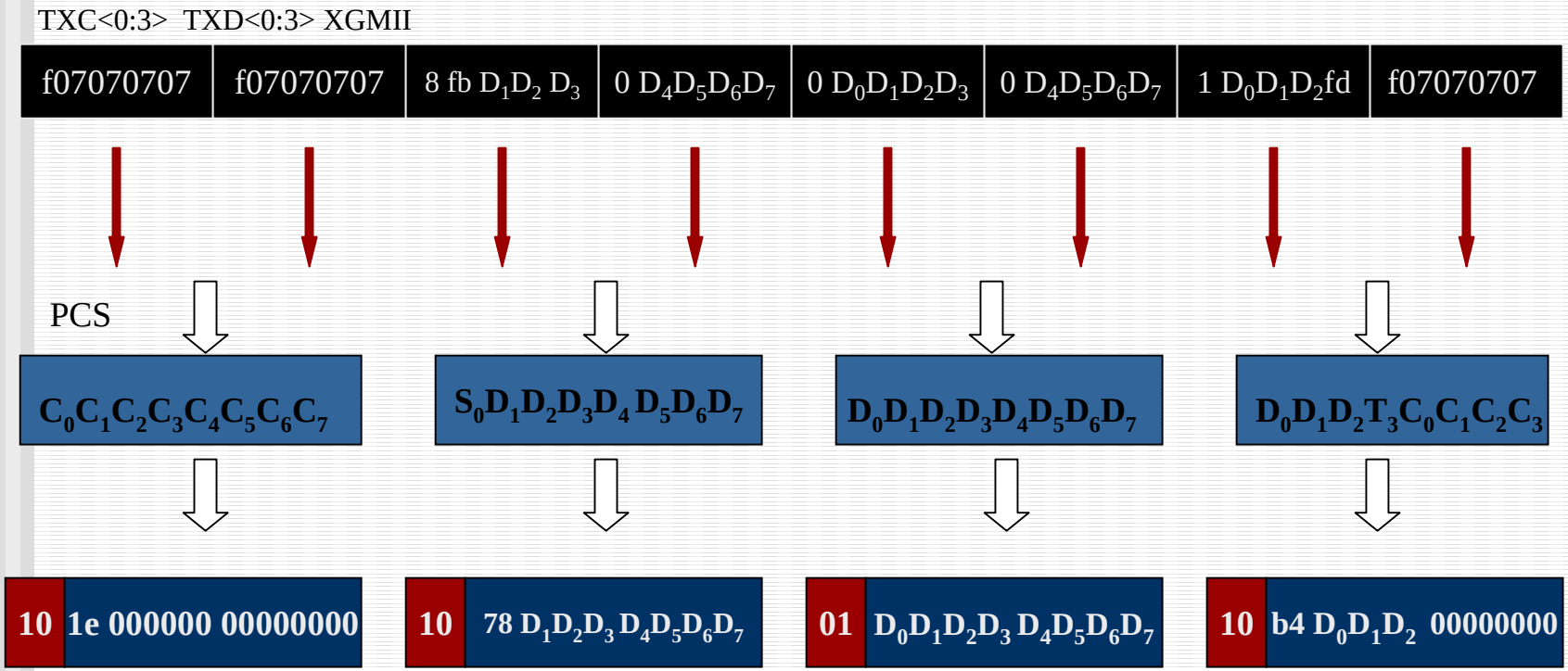
Control Codes

Control Character	Notation	XGMII Control Code	10GBASE-R Control Code	10GBASE-R O Code	8B/10B Code*
idle	/I/	0x07	0x00		
start	/S/	0xfb	encoded by type field		K27.7
terminate	/T/	0xfd	encoded by type field		K29.7
error	/E/	0xfe	0x1e		K30.7
Sequence ordered_set	/Q/	0x9c	encoded by type field plus O code	0x0	K28.4
reserved0	/R/†	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
reserved6	-	0x5c	-	0xF	K28.2

*For information only. 8B/10B code is specified in clause 48.
†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..

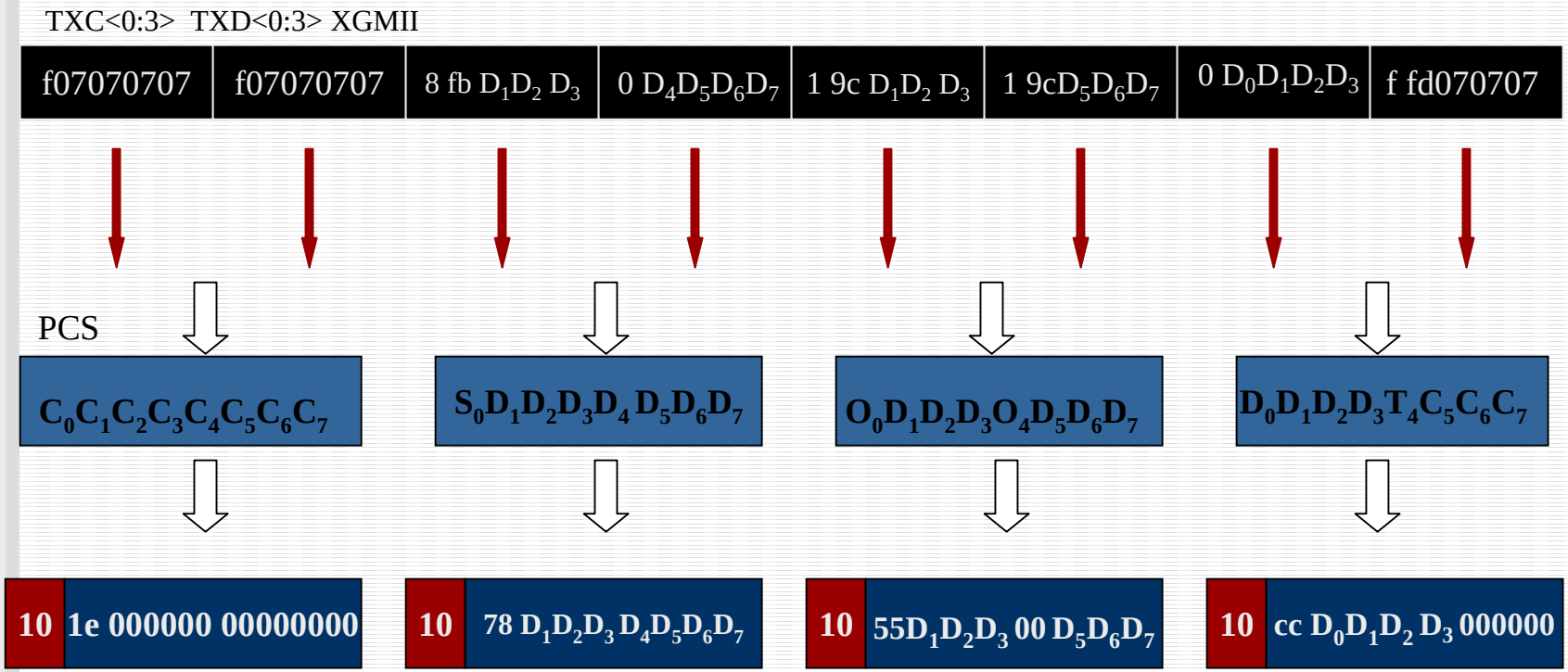
Examples of Code for 64b/66b transmit

Representation of Valid Code



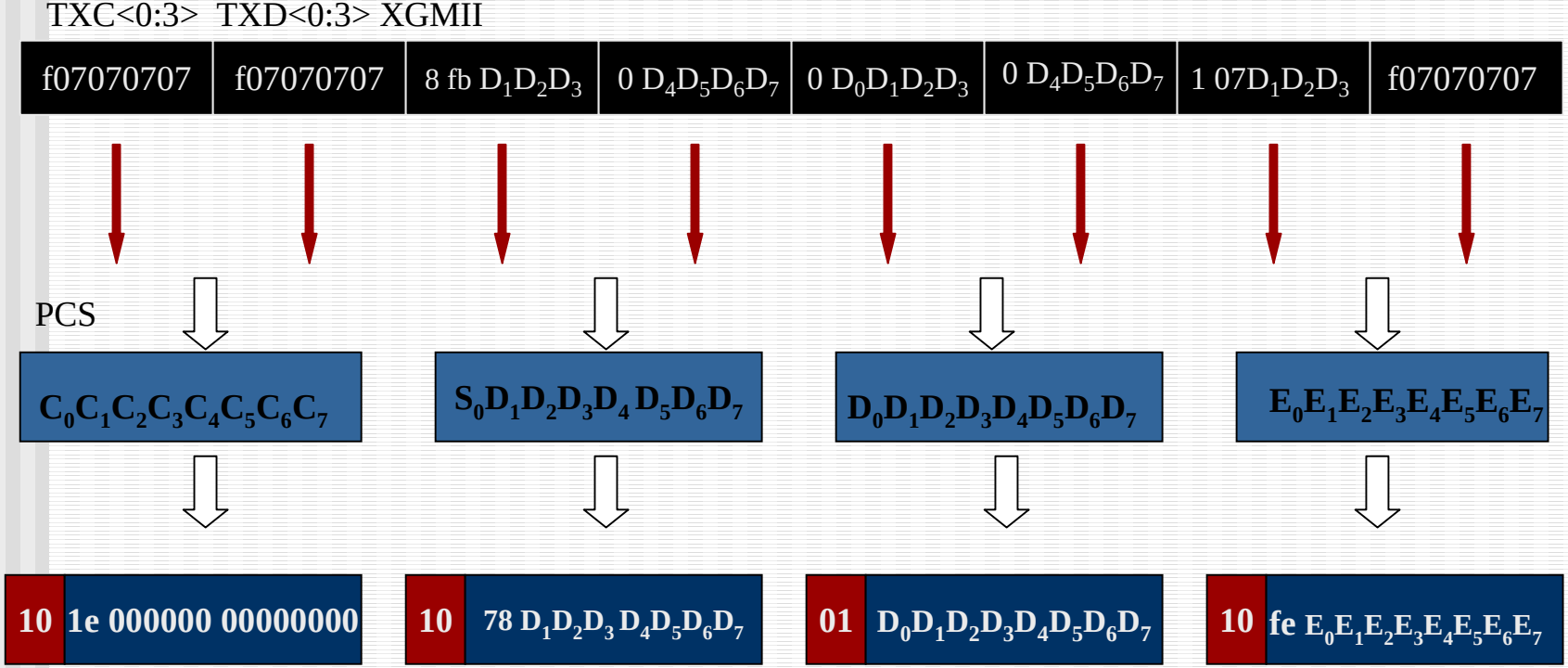
Examples of Code for 64b/66b transmit (2)

More Representation of Valid Code



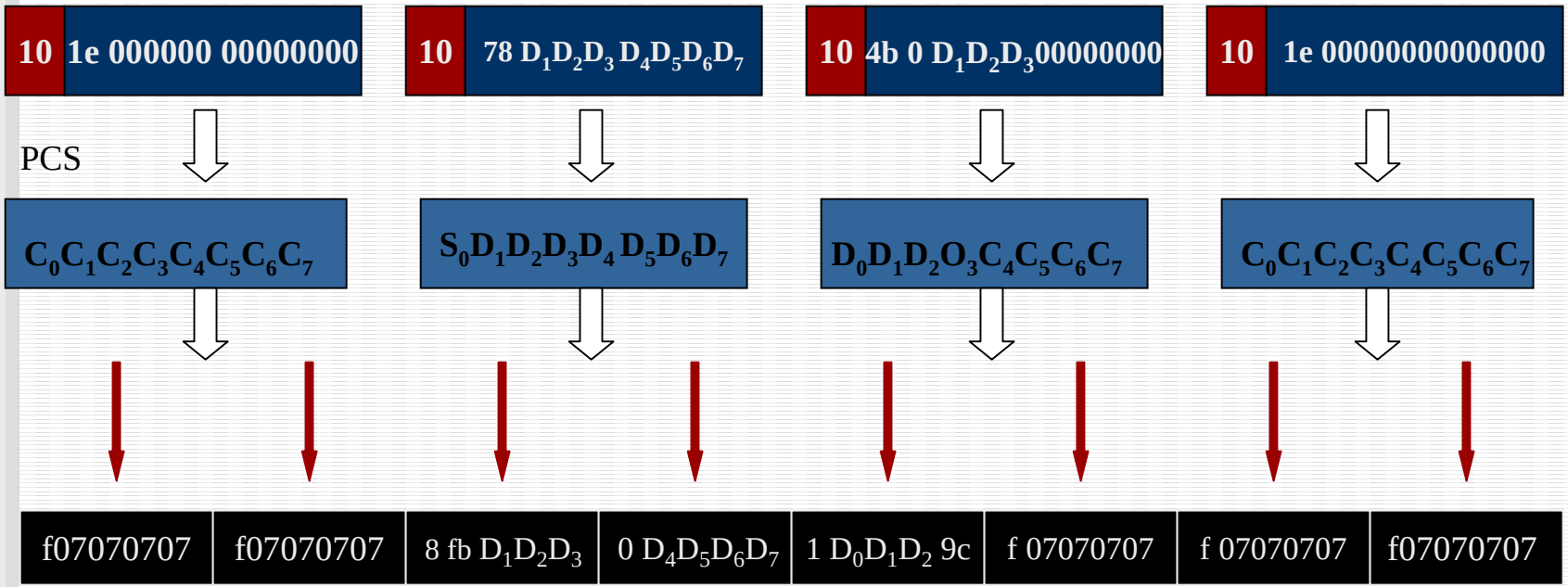
Examples of Code for 64b/66b transmit (3)

Representation of Invalid Code



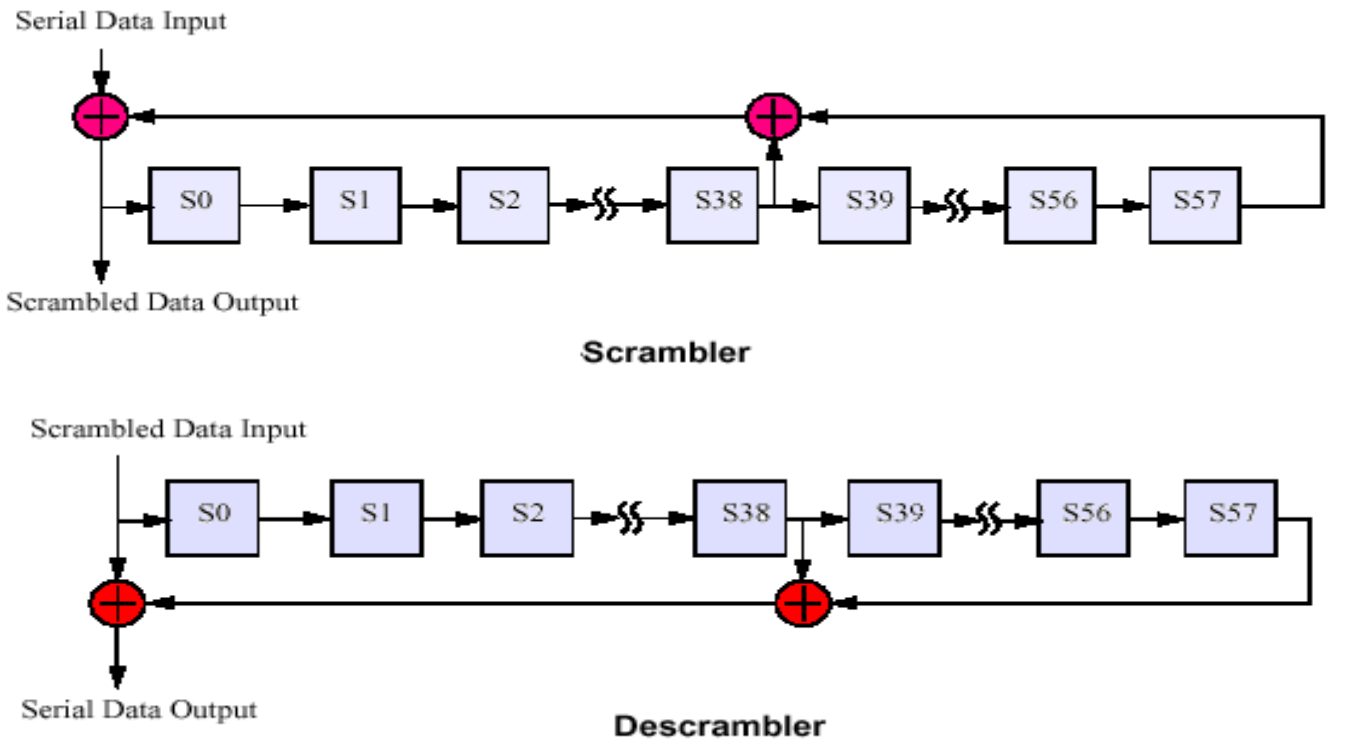
Example of Code for 64b/66b receive

Representation of Valid Code



RXC<0:3> RXD<0:3> XGMII

Scrambler / Descrambler



- Self-synchronizing scrambler scrambles 64 bit payload of block.
- Polynomial for the scrambler : $G(x) = 1 + x^{39} + x^{58}$
- No initial value needed for the scrambler since it runs continuously on all payload bits.
- Sync header is added after payload is scrambled.
- The descrambler uses the same polynomial and reverses the effect of the scrambler.

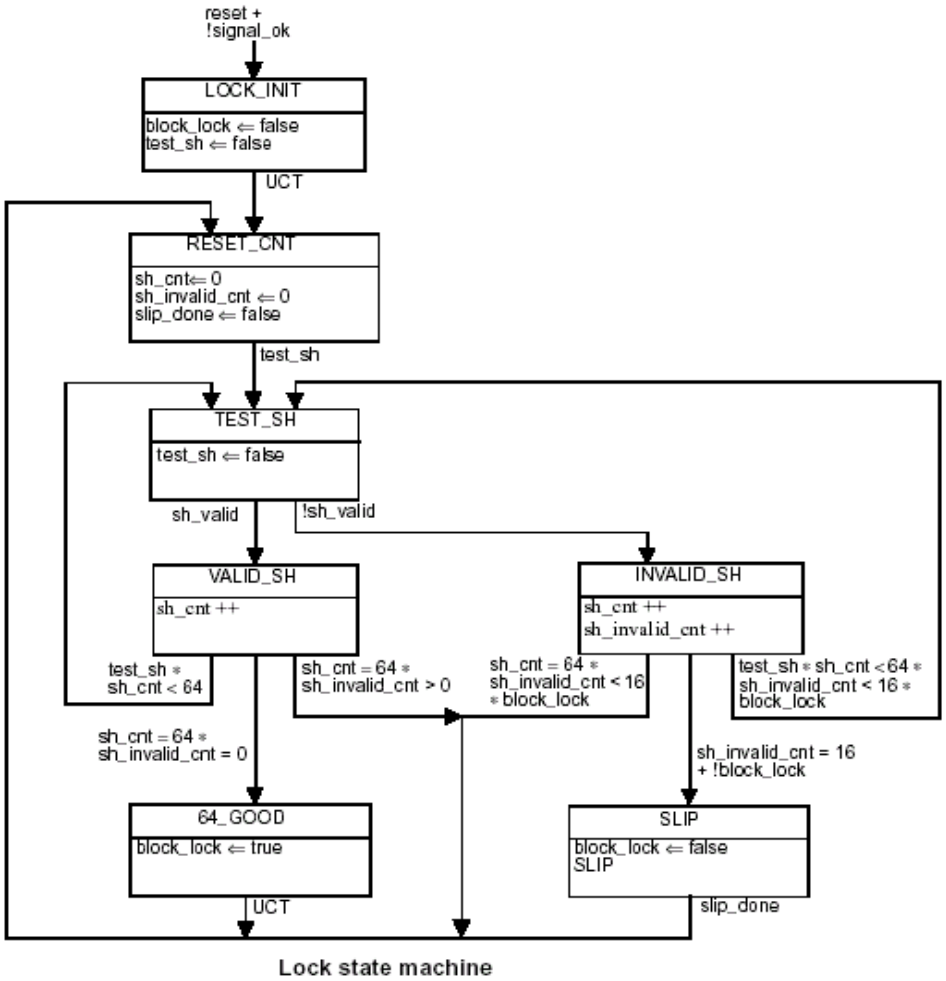
Gearbox

- Adapts between 66-bit width of the frames and 16-bit width of PMA or WIS
- Sends 16 bits of transmit data at a time
- No boundaries on 16 bit words
  - Sync headers can be in any two bit positions for example:
    - 01001101…
- When in receive process, needs to be able to slip the 66 bit output such that the sync bits are properly aligned to achieve synchronization
- Reduces pin count chip to chip
  - Matches the OIFs SFI-4 interface for OC192c

64b/66b State Diagrams

- Four state machines:
  - Lock State Machine
  - BER Monitor State Machine
  - Transmit State Machine
  - Receive State Machine

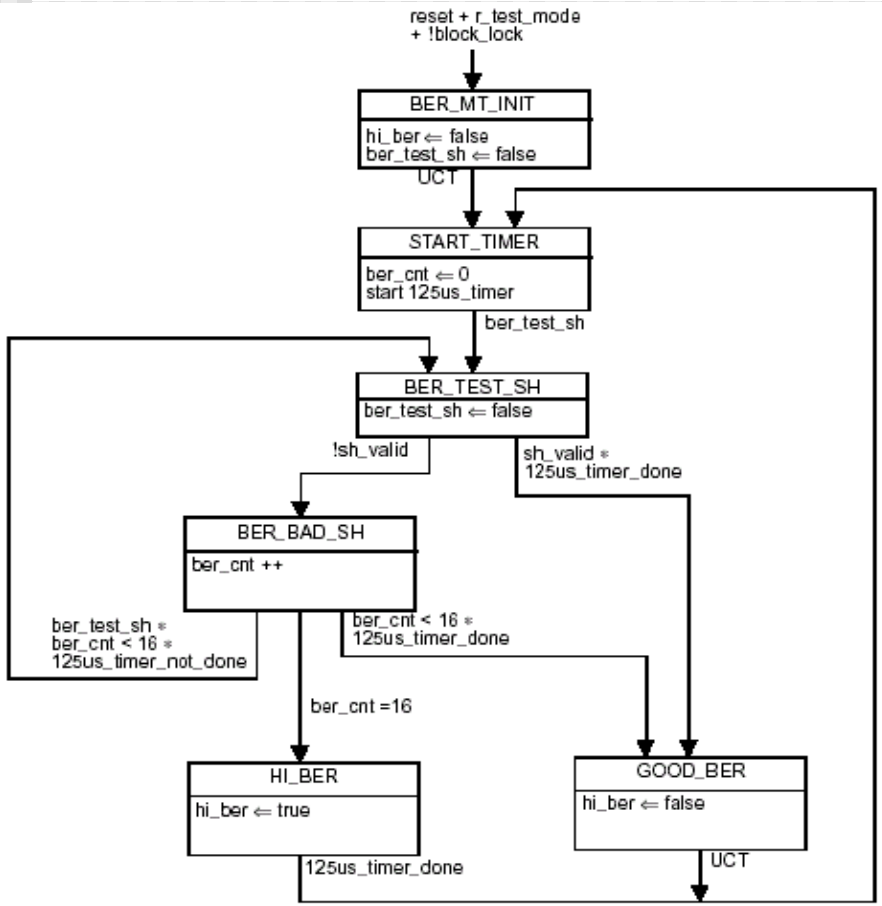
Lock State Machine



Lock state machine

- Sets block_lock to false when:
  - Sync header invalid count is 16
  - When any 16 66-bit blocks have invalid sync headers in a 64 block window. The block_lock is set to false and a new 66 bit alignment is tried.
- Sets block_lock to true when:
  - No invalid sync headers occur over a window of 64 blocks

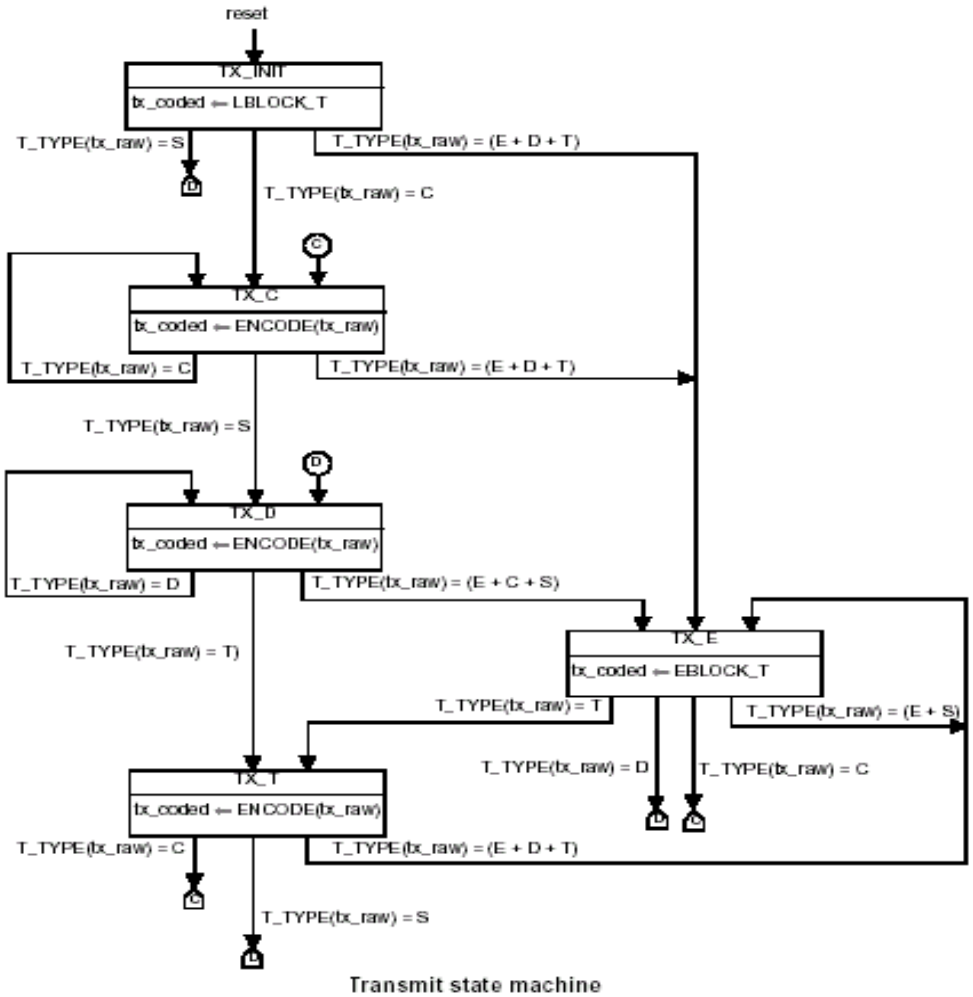
BER Monitor State Machine



BER monitor state machine

- Hi_BER set to true when:
  - Block_lock is false and BER count is equal to 16.
- Hi_BER set to false when:
  - Sync header is valid, timer is done and the BER count is under 16.

Transmit State Machine



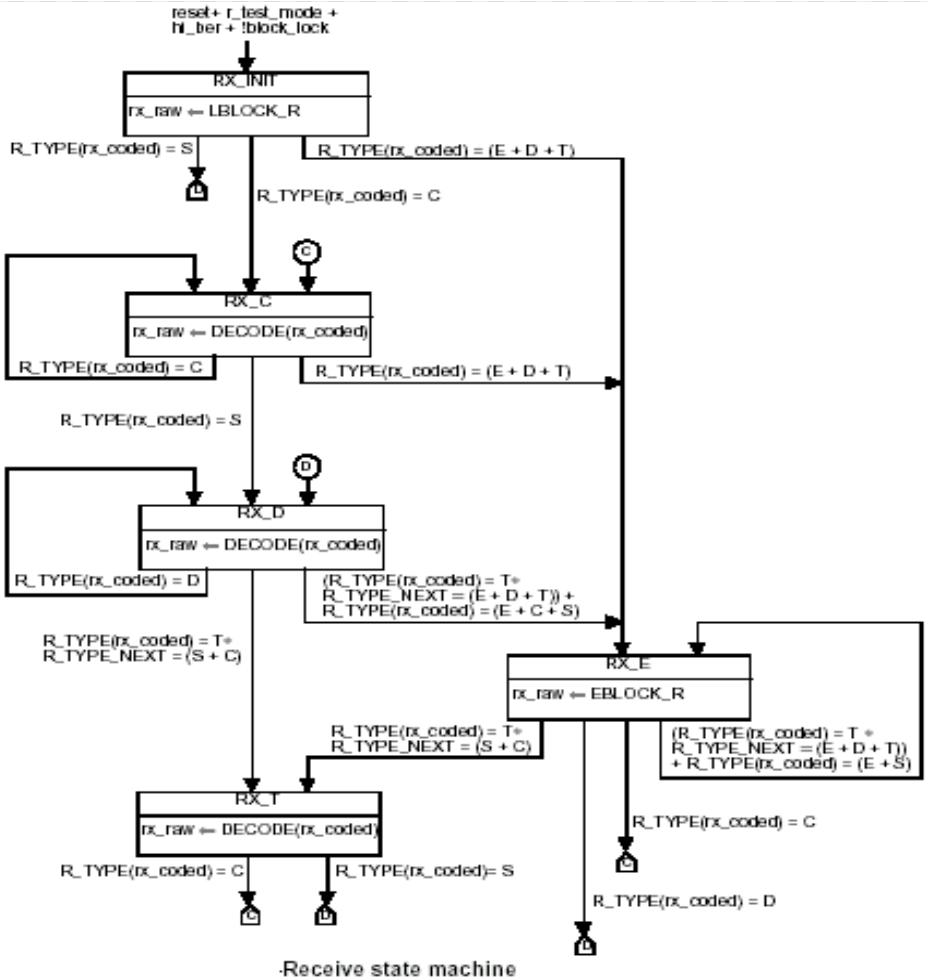
Transmit state machine

- C = Idle
- D = Data
- E = Error
- S = Start
- T = Terminate

ENCODE

- The 64b/66b encodes or processes eight data octets or control characters into one block.

Receive State Machine



Receive state machine

- C = Idle
- D = Data
- E = Error
- S = Start
- T = Terminate

DECODE

- The DECODE function decodes the block that contains eight data octets or control characters.

Credit

- Diagrams from IEEE P802.3ae/D3.2