64B/65B PCS encoding for GEPOF

David Ortiz
Rubén Pérez-Aranda
(rubenpda@kdpof.com)

IEEE 802.3bv Task Force - Plenary, March 2015

Agenda

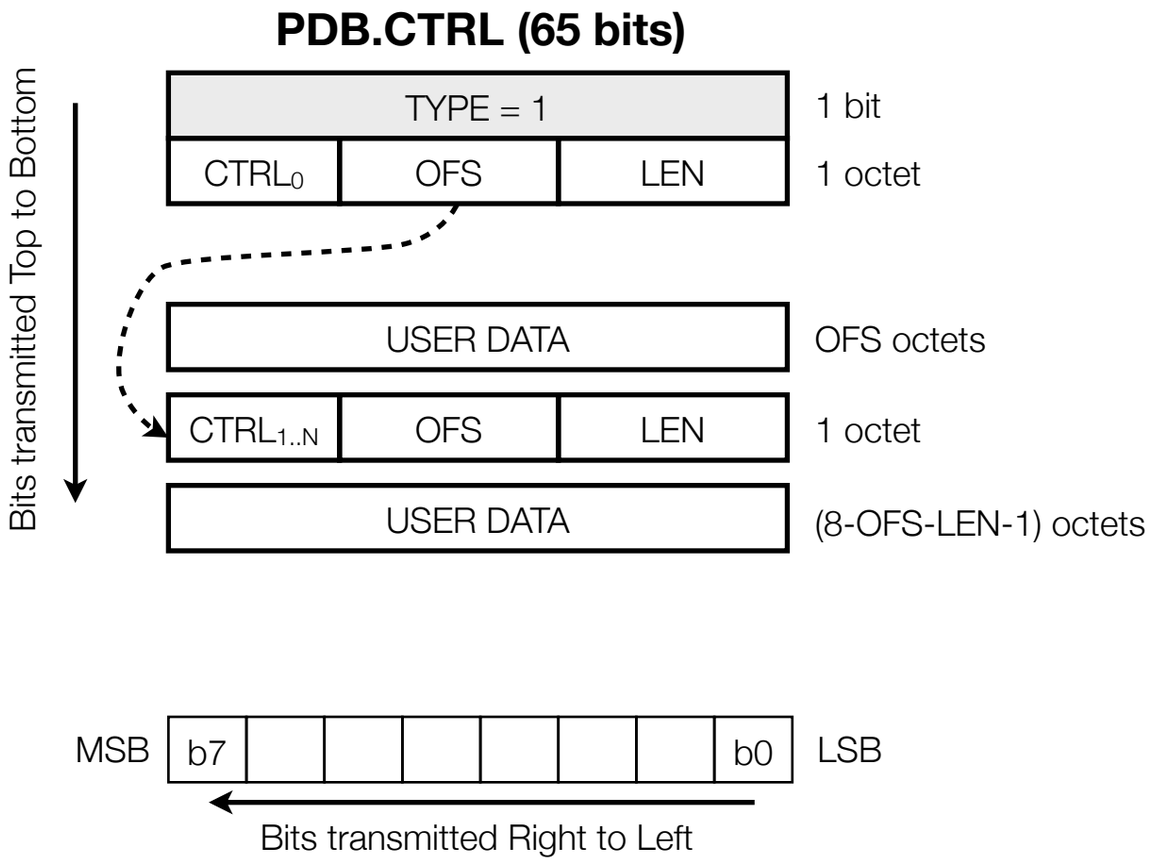
- Background and objectives
- PCS encoding - description
- PCS encoding - formal definition
- PCS encoding - error handling
- PCS encoding - latency and jitter analysis

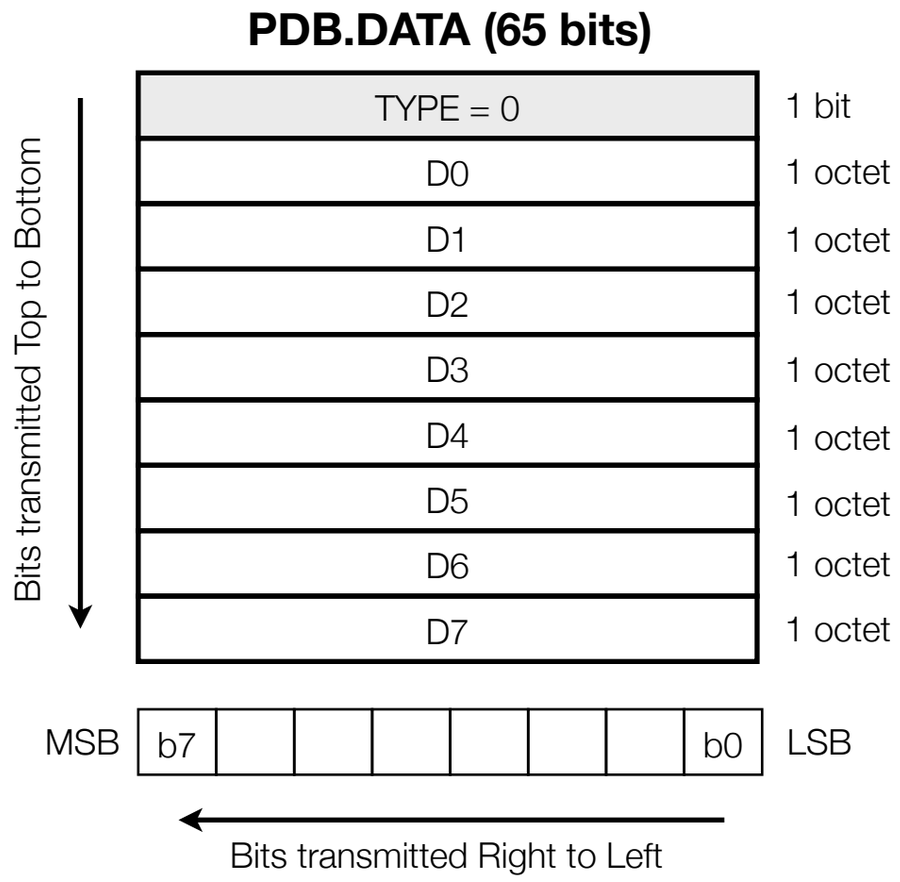
Background & Objectives

- In 802.3bv TF interim meeting of January 2015, the FEC and modulation schemes proposed in [1] were adopted for the baseline together with the transmission structure defined in [2]
- Both, the FEC encoding and the transmission scheme are asynchronous respect to GMII Ethernet packets encoding (see 802.3 clause 3 for packet definition)
- The main objective of this presentation is to define an encapsulation method of any data entering the GMII interface into FEC code-words defined in [1] to be transmitted along the communication channel by using the scheme defined in [2], therefore allowing the end-to-end transmission of GMII control words and packets delimitation
- Only full duplex operation is considered in the PCS encoding definition, as adopted in the p802.3bv objectives. See annex 4A for simplified full duplex MAC
- The PCS encoding block defined in this presentation is going to be between the GMII TX interface and the Binary Scrambler as described in slide 12 of [2]
- Important note: the PCS encoding defined in this presentation is different to that defined in 802.3 Clause 55. The one defined in Clause 55 is oriented to encapsulate XGMII interfaces, which is 4 bytes aligned.

PCS encoding - description

- Ethernet frames from GMII TX are segmented and encapsulated into chunks of 8 octets (64 bits)
- One control bit is prepended to each 64 bits unit, composing the basic transmission unit that we are going to call Physical Data Block (PDB) of 65 bits each
- Type bit is used to indicate if the PDB contains only GMII normal data transmission (PDB.DATA) or contains information from at least 1 GMII control byte (PDB.CTRL)
- “Normal” data transmission: the whole Ethernet packet content from Preamble to FCS between GMII control bytes (see 802.3 Clause 3 and Annex 4A)





PCS encoding - description

- $CTRL_X$ fields carry the information of the control bytes present in the GMII.
- The mapping of the encoded 2-bit CTRL field is given in the following table:

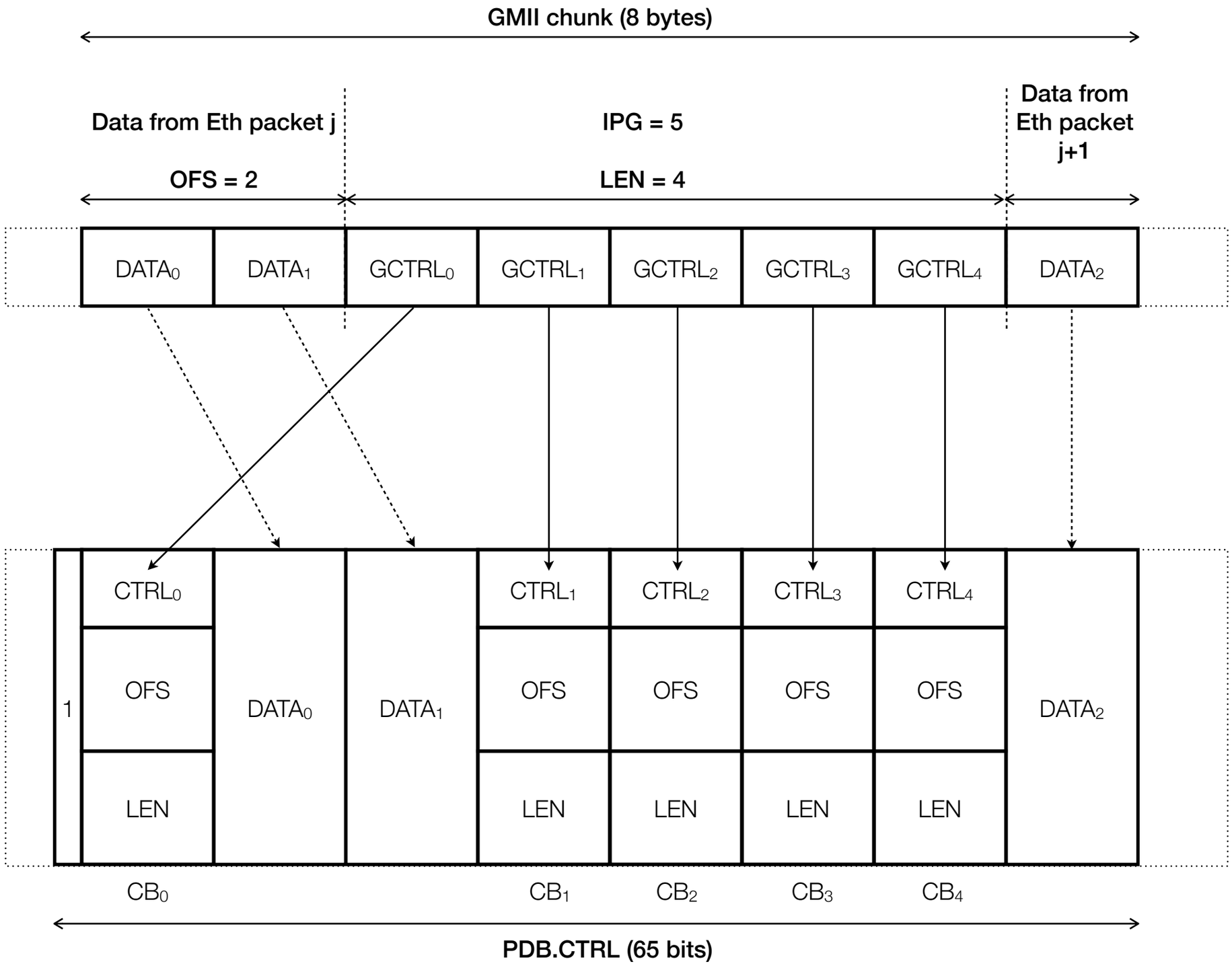
CTRL[1:0]	GMII TX	GMII RX
00	Transmit Error Propagation	Data Reception Error
01	Idle (Normal Inter-frame)	Idle (Normal Inter-frame)
10	Assert Low Power Idle (LPI)	Assert Low Power Idle (LPI)
11	RESERVED	RESERVED

PCS encoding - description

- 1 PDB is generated for each chunk of 8 GMII bytes
- If the GMII chunk only contains data from a user data packet, then a PDB.DATA is generated.
- If the GMII chunk contains at least 1 GMII control byte, a PDB.CTRL is inserted. The generation procedure is the following:
  - Every GMII byte containing control information (GCTRL in the figure) is replaced by a Control Byte (CB in the figure) with the following contents:
    - Content of the GMII Control information encoded in two bits (CTRL).
    - Offset from the beginning of the GMII chunk to the first GMII control byte within the chunk encoded in 3 bits (OFS).
    - Total number of GMII control bytes encoded in 3 bits (LEN). LEN starts at 0 to indicate 1 GCTRL was present in the chunk.
  - Then the first CB byte is always shifted to the beginning of the PDB.CTRL.
- Since the minimum length of an Ethernet packet is higher than 7 bytes, all the GMII control bytes in a chunk must be contiguous, so the proposed encapsulation covers all the possible scenarios with a minimum IPG of 1 byte and a minimum Ethernet packet length of 7 bytes (see Clause 4 Annex 4A).

PCS encoding - example

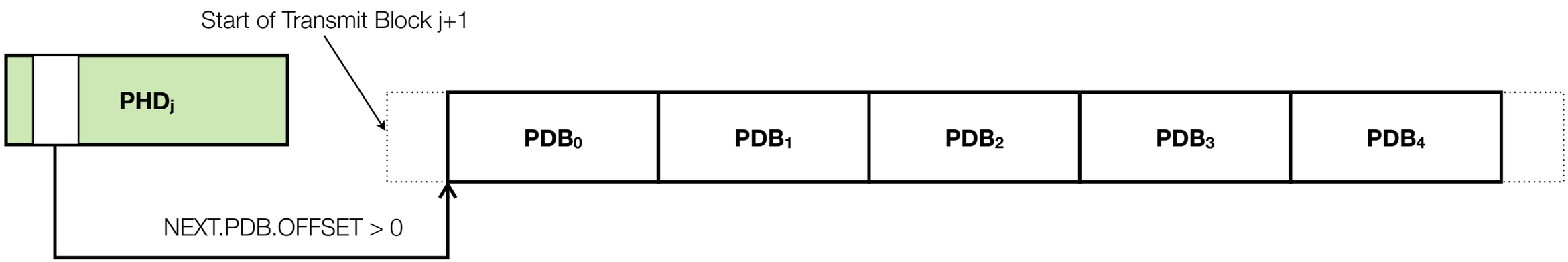
GMII TX Interface example



Encapsulated PDB's example

PCS encoding - description

- The number of information bits in a transmission block (see [2]) is not multiple of the PDB length, so in general the PDB are not aligned to the transmission block structure:
  - For fast synchronization and checking, the Physical Header Data (PHD) of transmission block j gives the offset to the start of the first PDB in transmission block j+1 (PHD.TX.NEXT.PDB.OFFSET)
  - PHD.TX.NEXT.PDB.OFFSET[k+1] = (PHD.TX.NEXT.PDB.OFFSET[k] + block_length) mod 65
  - block_length (information bits)= 28·8·(1668 + 1482), calculated according to [1] and [2]
  - See [3] for PHD content definition



PCS encoding - description

- The offset and length fields that indicate the position and number of control bytes in the original GMII chunk are replicated several times in the PDB. Even if not essential, this redundancy helps rejecting badly formed PDBs
- As can be seeing in [2], the proposed PCS encoding provides a data rate which is exactly $65 \cdot 10^9/64$ bps. No rate matching is needed, so Ethernet packets can be directly encapsulated using the proposed PCS encoding method
- As it was described in [1], the transmission structure is composed by different parts (pilots, header, and payload), being the structure periodic in time
  - When a packet is received from the GMII TX and it is encapsulated, it could be delayed due to the transmission of pilot or headers parts, introducing a deterministic jitter by the nature of the transmission structure
  - To avoid this, flow control is implemented in FEC encoder, in such a way that the bit-rate demanded in the encoder input is constant and equal to the average one
  - Target constant bit-rate should be $1000 \cdot 65/64$ Mbps = 1015.625 Mbps

PCS encoding - formal definition

- Matlab code for PCS encoding is provided as formal definition

```
%-------------------------
% Variables definition
%-------------------------

GMII.TX_ER    % GMII TX_ER signal, 1xL row vector
GMII.TX_EN    % GMII TX_EN signal, 1xL row vector
GMII.TXD      % GMII TXD bus, 1xL row vector
PDB.TYPE      % PDB type field, 1x(L/8) row vector
PDB.PAYLOAD   % PDB payload field, 8x(L/8) matrix


%-------------------------
% PCS encoding procedure
%-------------------------

for i = 0:8:length(GMII.TXD)-1,
  TC = (GMII.TX_EN(i+1:i+8)*2 + GMII.TX_ER(i+1:i+8)) ~= 2;

  if any(TC)
    % There are at least one control byte in the 8-bytes GMII chunk
    % The PDB is a PDB.CTRL
    PDB.TYPE(i/8 + 1) = 1;

    if any(~TC(min(find(TC)):max(find(TC))))
      % This is not a valid GMII chunk because there are no contiguous
      % control bytes
      % Transmit error encoding
      PDB.PAYLOAD(:, i/8 + 1) = 7*ones(8,1);

    else
      % Valid GMII chunk
      % OFS field computation
      OFS = min(find(TC)) - 1;

      % LEN field computation
      LEN = sum(TC) - 1;

      % Build the PDB.CTRL payload
      for j = 1:8,
        if TC(j)
          if ~GMII.TX_EN(i+j) & ~GMII.TX_ER(i+j)
            % Normal inter-frame
            CTRL = 1;
          elseif ~GMII.TX_EN(i+j) & GMII.TX_ER(i+j) & (GMII.TXD(i+j) == 1)
            % Assert LPI
            CTRL = 2;
          elseif GMII.TX_EN(i+j) & GMII.TX_ER(i+j)
            % Transmit error propagation
            CTRL = 0;
          else
            % Otherwise, default case (normal inter-frame)
            CTRL = 1
          end

          B(j) = LEN + OFS*2^3 + CTRL*2^6;
        else
          B(j) = GMII.TXD(i+j);
        end
      end

      % Shift the first control byte
      PDB.PAYLOAD(1,     i/8 + 1) = B(OFS + 1);
      PDB.PAYLOAD(2:end, i/8 + 1) = [B(1:OFS) B(OFS+2:end)].';
    end

  else
    % Pure normal data transmission bytes in the GMII chunk
    % The PDB is a PDB.DATA
    PDB.TYPE(i/8 + 1)       = 0;
    PDB.PAYLOAD(:, i/8 + 1) = GMII.TXD(i+1:i+8).';
  end
end
```

PCS encoding - error handling

- The present PCS encoding proposal assumes error detection capability in the MLCC decoder to satisfy the required MTTFPA (see [4] and [5])
- If BCH decoder operating at first decoding level cannot correct errors (correction capability overpassed) for a given code-word, then it marks all the affected PCS 64B/65B blocks as errors (i.e. GMII RX_ER signal)
- BCH error detection capability allows a trivial error handling by decapsulator

PCS encoding - latency & jitter analysis I

- If all the clocks in the chain (GMII TX, Transmitter+Receiver, and GMII RX) are generated from the same source:
  - No rate adaptation is needed as the bitrates of all the interfaces are exactly matched.
  - Encapsulation latency is minimal and is limited to the time needed to receive 8 GMII bytes.
  - Decapsulation latency is also minimal as the receiver can start generating an Ethernet packet right after 1 byte of the PDB has been received.
    - This minimal latency is needed to compensate for the possible shift of 1 byte that happens if the last bytes of a data packet are encapsulated in a PDB.CTRL, as can be seen in the previous slices.
    - This delay also allows processing of the first GMII control word immediately following the data packet which could carry the transmit error indication, for error forwarding before the end of the packet is transmitted by the decapsulator to GMII RX
  - Encapsulation jitter due to the encapsulation protocol is 0. At the encapsulation side the first byte of an data packet is always inserted in the PDB with a fixed offset with respect to the reception of the byte, regardless of the type of PDB. At the decapsulator side, the latency is also constant.
  - In this scenario the total extra latency of the system is only limited by the FIFOs needed on the cross-domain crossings due to the different frequencies involved (i.e. 125 MHz at GMII and 325 MHz at the PHY), and accounts for a GMII clock cycles.
  - In this scenario the jitter is negligible and only limited by the needed synchronization of the different clocks
    - PHY TX & RX clocks run at the same frequency: 325MHz and the jitter is of few ps.
    - The extra jitter will be limited by the quality of the different clocks and the mechanisms used to generate them.

PCS encoding - latency & jitter analysis II

- In a more practical scenario, the PHY TX clock will be generated from a local free-running clock not related to the GMII TX clock, and GMII RX clock will be generated independently from the clock recovered by the receiver
  - Encapsulator and Decapsulator need to implement a small buffer of few bytes length to accommodate for rate mismatches over the maximum length Ethernet packet due to frequency deviations of the different clocks (i.e. to void FIFO underflow). This small buffer increases the total latency in a few GMII clocks
  - Encapsulator and Decapsulator also need to perform a minor rate adaptation to compensate for clock deviations. This rate adaptation is performed by adding or removing GMII (shrinkage) control bytes during the IPG
  - In this situation the main GMII to GMII jitter contributor is given by the cross-domain crossings, and is estimated typically limited to < 4 GMII clock cycles

References

- [1] *Rubén Pérez-Aranda, et al., "High spectrally efficient coded 16-PAM scheme for GEPOF based on MLCC and BCH", IEEE 802.3bv TF, Interim Meeting, January 2015*
- [2] *Rubén Pérez-Aranda, "Transmission scheme for GEPOF", IEEE 802.3bv TF, Interim Meeting, January 2015*
- [3] *Rubén Pérez-Aranda, et al., "Physical Header Data content for PCS encoding, PHY control and OAM implementation in GEPOF", IEEE 802.3bv TF, Plenary Meeting, March 2015*
- [4] *Rubén Pérez-Aranda, et al., "Study of Undetected Error Probability of BCH codes for MTTFPA analysis", IEEE 802.3bv TF, Interim Meeting, January 2015*
- [5] *Rubén Pérez-Aranda, et al., "Performance analysis of coded 16-PAM scheme for GEPOF. BER, MTBE, MTTFPA, PER", IEEE 802.3bv TF, Interim Meeting, January 2015*

Questions?