PCS receive function

The PCS receiver accepts equalized symbols provided by the PMA receiver. The PCS receiver knows to which part of the received Transmit Block the symbols belong to, based on the symbol time alignment information provided by the clock recovery function of PMA receiver (see 114.3.2 and 114.3.5.3). The PCS receiver carries out the PHD decoding, the MLCC decoding of payload sub-blocks, and the 64B/65B decoding.

The PHD decoding comprises detection and de-mapping of the received PAM2 symbols, BCH decoding for error correction, binary data descrambling and CRC16 checking for each received PHD. If the CRC16 computation indicates that the received PHD is correct, the contents of the different PHD fields are available to the PMA state diagrams and to the other PCS receive function that need this information.

The PAM16 symbols belonging to the payload data sub-blocks are first symbol-descrambled, and the resulting symbols are MLCC decoded, with error correction and error detection. If during MLCC decoding it is detected that a codeword contains errors that could not be corrected, the resulting bits belonging to that codeword are marked as corrupt. The bitstream is then binary descrambled.

Using the PHD.NEXT.PDB.OFFSET field of the decoded PHD carried in the previous Transmit Block, the PCS receiver knows the alignment of the first PDB of a Transmit Block and can extract all the PDBs from the descrambled bitstream. The PDB are then finally processed by the 64B/65B decoder to extract the GMII receive data stream, using also the information that indicates which parts of the bitstream belong to codewords that could not be corrected.

The 64B/65B decoder implementation shall produce output consistent with the following MATLAB (see 1.3) code when state variable rx_gmii_enable is TRUE.

```
% Variables definition
GMII.RX_ER      % GMII RX_ER signal, 1xL row vector
GMII.RX_EN      % GMII RX_EN signal, 1xL row vector
GMII.RXD        % GMII RXD bus, 1xL row vector
PDB.TYPE        % PDB type field, 1x(L/8) row vector
PDB.PAYLOAD     % PDB payload field, 8x(L/8) matrix
PDB.PAYLOAD_ERR % PDB payload error flag, 1xL row vector. It indicates
                % if any of the bits within the corresponding byte of
                % the PDB payload belongs to a codeword that could not be
                % corrected.
PDB.TYPE_ERR    % PDB type field error flag, 1x(L/8) row vector. It
                % indicates if the type bit of the PDB belongs to a
                % codeword that could not be corrected.


for i = 1:length(PDB.TYPE)
   if (PDB.TYPE_ERR(i) | (PDB.TYPE(i) & PDB.PAYLOAD_ERR(1,i)))
      % Complete PDB is in error. Decode forward error propagation.
      GMII.RX_EN(8*(i-1)+1:8*(i-1)+8) = ones(1, 8);
      GMII.RX_ER(8*(i-1)+1:8*(i-1)+8) = ones(1, 8);
      GMII.RXD  (1:8, i)              = zeros(8, 1);

   elseif (PDB.TYPE(i))
      % Decode PDB.CTRL payload
      OFS  = bitand(PDB.PAYLOAD(1,i),   7*2^3)/2^3;
      LEN  = bitand(PDB.PAYLOAD(1,i),   7);
      CTRL = bitand(PDB.PAYLOAD(:,i).', 3*2^6)/2^6;

      % Assign initial data.
      if (OFS > 0)
         GMII.RX_EN(8*(i-1)+1:8*(i-1)+OFS) = ones(1,OFS)  | PDB.PAYLOAD_ERR(2:OFS+1, i).';
         GMII.RX_ER(8*(i-1)+1:8*(i-1)+OFS) = zeros(1,OFS) | PDB.PAYLOAD_ERR(2:OFS+1, i).';
         GMII.RXD  (8*(i-1)+1:8*(i-1)+OFS) = PDB.PAYLOAD(2:OFS+1, i).';
      end
   % Assign control information.
   GMII.RX_EN(8*(i-1)+OFS+1:8*(i-1)+OFS+1) = (CTRL(1) == 0) | PDB.PAYLOAD_ERR(1, i).';
   GMII.RX_ER(8*(i-1)+OFS+1:8*(i-1)+OFS+1) = (CTRL(1) ~= 1) | PDB.PAYLOAD_ERR(1, i).';
   GMII.RXD  (8*(i-1)+OFS+1:8*(i-1)+OFS+1) = (CTRL(1) == 2);

   if (LEN > 0)
      GMII.RX_EN(8*(i-1)+OFS+2:8*(i-1)+OFS+LEN+1) =
        (CTRL(OFS+2:OFS+LEN+1) == 0) | PDB.PAYLOAD_ERR(OFS+2:OFS+LEN+1, i).';

      GMII.RX_ER(8*(i-1)+OFS+2:8*(i-1)+OFS+LEN+1) =
        (CTRL(OFS+2:OFS+LEN+1) ~= 1) | PDB.PAYLOAD_ERR(OFS+2:OFS+LEN+1, i).';

      GMII.RXD  (8*(i-1)+OFS+2:8*(i-1)+OFS+LEN+1) =
        (CTRL(OFS+2:OFS+LEN+1) == 2);
   end

   % Assign final data.
   if ((OFS+LEN+1) < 8)
      GMII.RX_EN(8*(i-1)+OFS+LEN+1+1:8*(i-1)+8) = ones(1, 8-(OFS+LEN+1));
      GMII.RX_ER(8*(i-1)+OFS+LEN+1+1:8*(i-1)+8) = PDB.PAYLOAD_ERR(OFS+LEN+2:8, i)';
      GMII.RXD  (8*(i-1)+OFS+LEN+1+1:8*(i-1)+8) = PDB.PAYLOAD(OFS+LEN+2:8, i)';
   end

else
   % Decode PDB.DATA payload
   GMII.RX_EN(8*(i-1)+1:8*(i-1)+8) = ones(1,8);
   GMII.RX_ER(8*(i-1)+1:8*(i-1)+8) = PDB.PAYLOAD_ERR(1:8, i).';
   GMII.RXD  (8*(i-1)+1:8*(i-1)+8) = PDB.PAYLOAD(1:8, i).';
end
```