RS code scheme to protect “un-coded” bits at 40GBASE-T

Phil Sun, Peter Wu, Zhenyu Liu and William Lo
Marvell Semiconductor

Overview

- At 802.3bq May, 2014 Norfolk meeting:
  - Motion Passed to adopt RS code to protect “uncoded “ bits:

    Motion #3: Move to adopt Lo_3bq_02_0514.pdf to protect the bits unprotected by the LDPC.

    M: William Lo S: Tom Souvignier

    Y: 14 N: 0 A: 7

    MOTION PASSES (Technical >75%)
  - RS(140, 136, 2^{11}) has been selected
  - But generator polynomial was missing
- Primitive generator polynomial needed to be defined for the RS code and detailed procedures for bit mapping and RS encoding need to be defined.

The Scheme protecting un-coded bits

- Transcoding modification and aggregation to save bits for RS check bits
  - 4X64/65->1X256/257 + 3 bits
  - 50X65bits ->12X256/257-bit+2X64/65 bit + 36bits
  - 36+8CRC-> 44 bits -> RS check bits
  - A similar scheme has been used 100GKR- 803.3bj- Clause 91.5.2.5
- No LDPC bits involved
- No changes to 128 DSQ
- No more CRC8
- Use 11 bits symbols - RS(140, 136, 2^{11})
  - 2t= 4 X11 bits symbols
  - Maximize use of 45 available parity bits (4 x 11 bits = 44 bits)

Proposal: Field GF(2^{11}) and RS(140,136) Code

- Field GF(2^{11})
  - Primitive generator polynomial: $p(x)=x^{11}+x^2+1$
  - Primitive element: $\alpha=2$ (in octal)
  - $GF(2^{11})=\{0,1, \alpha^1, \alpha^2\ldots, \alpha^{2046}\}$
- RS(140,136) code:
  - Generator polynomial: $g(x)=(x- \alpha^0)(x- \alpha^1)(x- \alpha^2)(x- \alpha^3)$
  - Correction capability: 2 11-bit symbols
  - Error detection capability: 4 11-bit symbols

Implementation Details- current 10GBASE-T

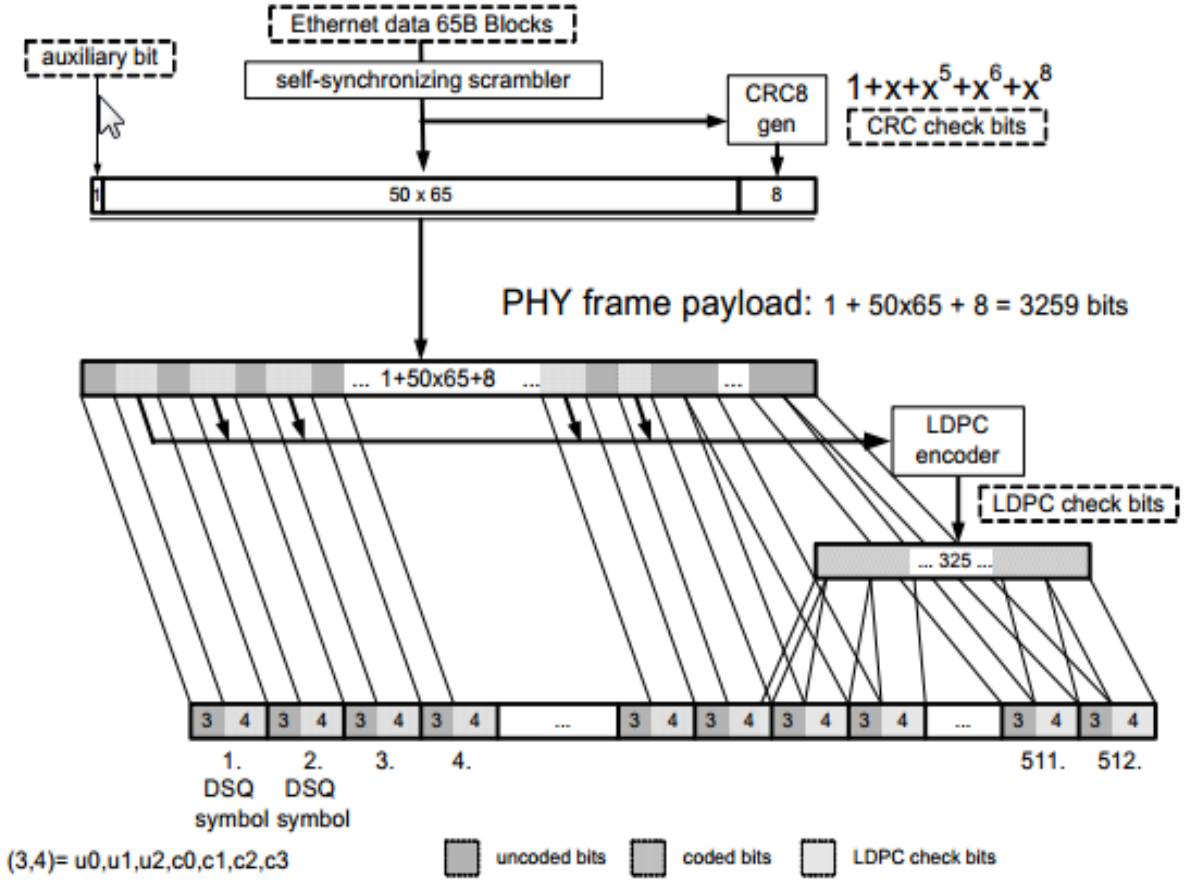
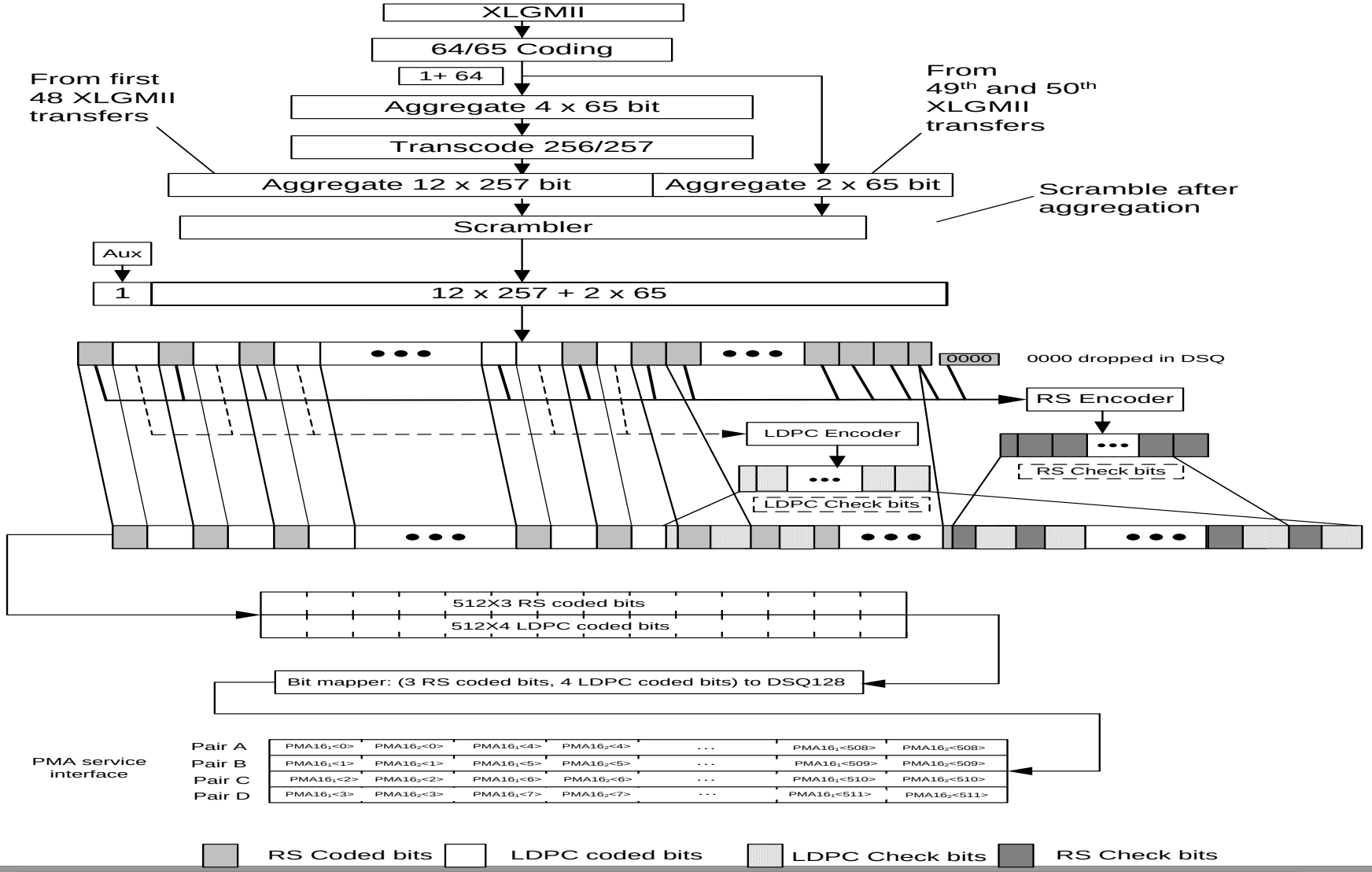


Figure 55–8—PCS detailed transmit bit ordering

- **64/65 coding**
- **Scramble**
- **Pad + 50 x 65 bit + CRC = 3259 bits**
- **1536 bits –uncoded**
- **1723 bits –coded**
- **325 bits –LDPC parity**
- **3584 bits -Total**

The Bit mapping:



Comments to D1.0

- 98.3.2.2.20 Reed Solomon encoder - pending approval
  - Show detailed text
  - Show an addition to the original text

- Some issues with the figures:
  - Figure 98-13
    - It is copied at here “ as is”
    - The issues have been resolved in other comments
  - Figure 98-9
    - Some comments regarding the labels
    - Suggest to be replaced with Figure in Page 6.

The text at D1.0 pending approval

98.3.2.2.20 Reed Solomon encoder

The group of 1536 bits are encoded using a Reed-Solomon encoder operating over the Galois Field GF(2^{11}) where the symbol size is 11 bits. The encoder process k message symbols to generate 2t parity symbols, which are then appended to the message to produce a codeword of n=k+2t symbols. For the purposes of this clause, the particular Reed-Solomon code is denoted RS(n,k). RS(140, 136) is used. The code is based on the generating polynomial given by Equation (98–3).

$$g(x) = \prod_{j=0}^{3}(x - \alpha^j) = g_4x^4 + g_3x^3 + g_2x^2 + g_1x + g_0 \qquad (98–3).$$

In Equation (98–3), α is a primitive element of the finite field defined by the polynomial $x^{11}+x^2+1$.

Equation (91–2) defines the message polynomial m(x) whose coefficients are the message symbols m_{135} to m_0.

$$m(x) = m_{135}x^{139} + m_{134}x^{138} + \ldots + m_1x^5 + m_0x^4 \qquad (98\text{-}4)$$

Continued:

Each message symbol m_i is the bit vector ($m_{i,10}$, $m_{i,9}$, … $m_{i,1}$, $m_{i,0}$), which is identified with the element $m_{i,10}\alpha^{10} + m_{i,9}\alpha^{9} + \ldots m_{i,1}\,\alpha + m_{i,0}$ of the finite field. The message symbols are composed of the bits in tx_RSmessage<1495:0> where

$m_{i,j}$= tx_RSmessage<(135-i)11 + j), i = 0 to 135, j = 0 to 10
tx_RSmessage<1495:0> is formed as follows.
tx_RSmessage<0> = Auxiliary bit
tx_RSmessage<3j+3:3j+1> = tx_scrambled<7j+2:7j> where j = 0 to 430
tx_RSmessage<1491:1294> = tx_scrambled<3213:3016>
tx_RSmessage<1495:1492> = 0000

The first symbol input to the encoder is m_{135}.
tx_scrambled<3213:0> is defined in 98.3.2.2.18
Equation (91–3) defines the parity polynomial p(x) whose coefficients are the parity symbols p_3 to p_0.

$$p(x) = p_3x^3 + p_2x^2 + p_1x + p_0 \qquad (98\text{-}5)$$

Continued:

The parity polynomial is the remainder from the division of m(x) by g(x). This may be computed using the shift register implementation illustrated in Figure 98–13. The outputs of the delay elements are initialized to zero prior to the computation of the parity for a given message. After the last message symbol, m_0, is processed by the encoder, the outputs of the delay elements are the parity symbols for that message.

The codeword polynomial c(x) is then the sum of m(x) and p(x) where the coefficient of the highest power of x, $c_{139}= m_{135}$ is transmitted first and the coefficient of the lowest power of x, $c_0= p_0$ is transmitted last. The first bit transmitted from each symbol is bit 0.

The added 0000 at C_4 [3:0] will be omitted.

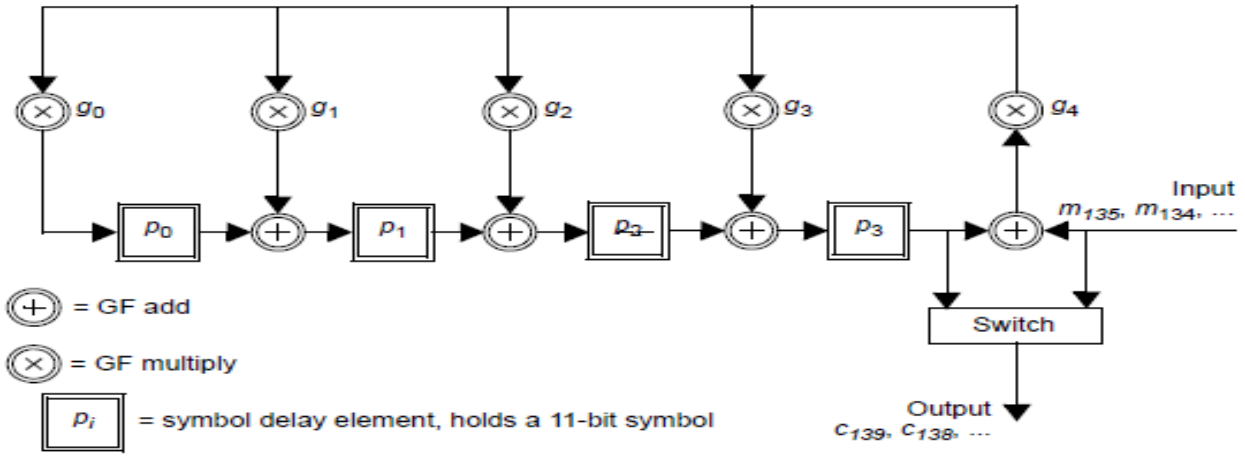


Figure 98–13—Reed-Solomon encoder functional model

Continued:

The coefficients of the generator polynomial for each code are presented in Table 98–2. Example codewords for each code are provided in Annex 98A

Table 98–2—Coefficients of the generator polynomial g_i (decimal)

i	RS(140, 136)
0	64
1	120
2	54
3	15
4	1

Q&A

THANKS YOU !