RS Options for 40GBASE-T

AQUANTIA®

Dr. Paul Langner
Dr. Hossein Sedarat

January 12, 2015

RS Overview

- Reed-Solomon is a block based error control code, built using Galois field elements
- Galois fields can be constructed from powers of prime numbers
  - Obviously 2^N is a very popular choice, which leads to a maximum block size of $2^N - 1$ symbols
  - Example: 2^8 gives a maximum block size of 255 bytes
- General format is M data symbols + 2T check symbols (a systematic code = payload is directly mapped)

RS Overview (continued)

- For RS codes, the 2T check symbols can locate and correct T errored symbols, and detect the presence of 2T errored symbols
  - Simple way to look at this limitation is that for each errored symbol, we need one symbol to locate the error within the block, and another symbol to contain the correction
- Code is described as RS2^N(Total symbols, Payload symbols)
  - For example RS256(128,120) would have 120 payload bytes in a frame with 8 check bytes capable of correcting 4 bytes in the frame
- Optimal RS block size is typically chosen such that 2^N symbols are just slightly larger than the number of symbols you have to correct
  - This usually maximizes the number of correctable symbols within the block

“Free” bits within the 40GBASE-T Frame

- 40GBASE-T is based on reusing10GBASE-T frame
- Within 10GBASE-T frame are the following possible bits:
  - Auxiliary bit
  - CRC-8 (presumably not needed as RS coverage is better than CRC
  - 50 potentially superfluous 65B frames – i.e. can convert to 129B, 257B, etc. asssuming the control code and ordered set start locations can be adequately mapped from XGMII
- Error control code has to protect 512 x 3 = 1536 uncoded bits

“Free” bits within the 40GBASE-T Frame (continued)

- Current 40GBASE-T scheme:
  - Converts 50x 65B blocks into 2x 65B + 12x 257B freeing up 36 bits
  - Combine with CRC-8 to create 44 free bits
  - Use an 11-bit symbol to create a 2 symbol correcting RS2048 code
  - Thus 1536 bits maps into 140 11-bit symbols with 4x zero bits giving us an RS2048(144,140) code
- Unfortunately, the symbol size is not optimal, as RS2048 can deal with a 2047 x 11-bit block

Optimal RS Code Choice for 40GBASE-T

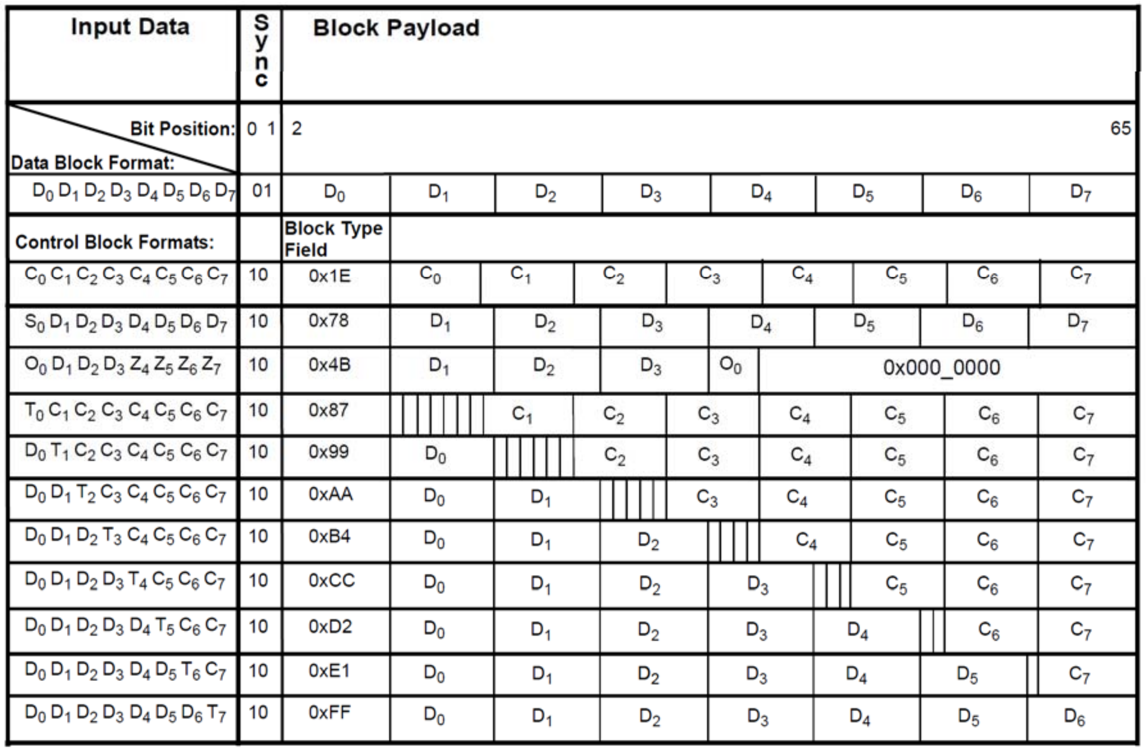
- Working in a 2^N-based field, we have the following choices for 1536 bits:
  - 2^7 = RS128 gives a maximum block size of 127 x 7 = 889 bits
  - 2^8 = RS256 gives a maximum block size of 255 x 8 = 2040 bits
- RS256 is the best choice as 1536 bits = 192 bytes
- Issue is that we need 48 bits instead of 44 bits to make GF256 effective (correct 3x 8-bit symbols)
- So we need four more bits from the frame

40GBASE-T Error Control Coding Proposal

- Convert 50x 65B blocks into 2x 65B + 6x 513B blocks freeing up 42 bits
- Combine with CRC-8 to create 50 free bits
- Use an 8-bit symbol to create a 3 symbol correcting RS256 code
- Thus 1536 bits maps into 192 8-bit symbols with 2x zero bits giving us an RS256(198,192) code

512/513B Encoding

- The 64/66B PCS of 40GBASE-R uses 11 block field types to transport all of the Start-of-Frame, End-of-Frame, and Ordered Set information received from the XLGMII interface*



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	

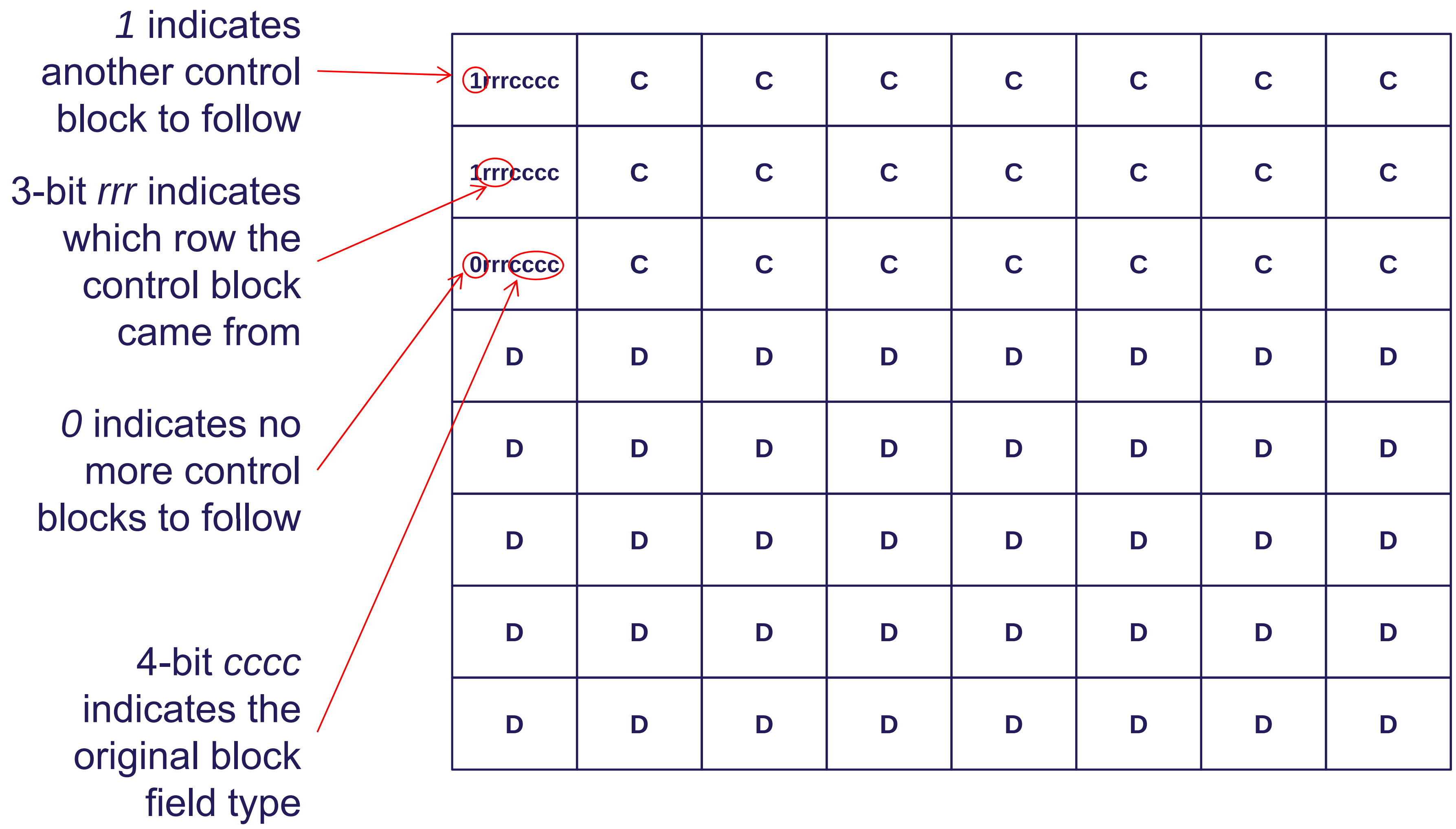
Figure 82–5—64B/66B block formats

* Versus 15 block field types in 10G

512/513B Encoding (continued)

- Since the block field type is an 8-bit value, there are 256 possible block field types, of which 11 are used
- Requires 4 bits to represent these block field types, leaving us 4 bits to encode all possible combinations of 64/66 block types in all combinations of 8 locations
- This can only be done using a clever scheme such as Trowbridge, et al. presented in 2007 (http://www.ieee802.org/3/hssg/public/july07/trowbridge_01_0707.pdf)
  - Here he uses the control/data bit for the block to indicate whether there are control frames present
  - If there are they are all placed at the top of the block with the first bit being used to indicate whether there are more control blocks to follow, and the next three bits to indicate the row

512/513B Encoding (continued)


1 indicates another control block to follow
3-bit *rrr* indicates which row the control block came from
0 indicates no more control blocks to follow
4-bit *cccc* indicates the original block field type

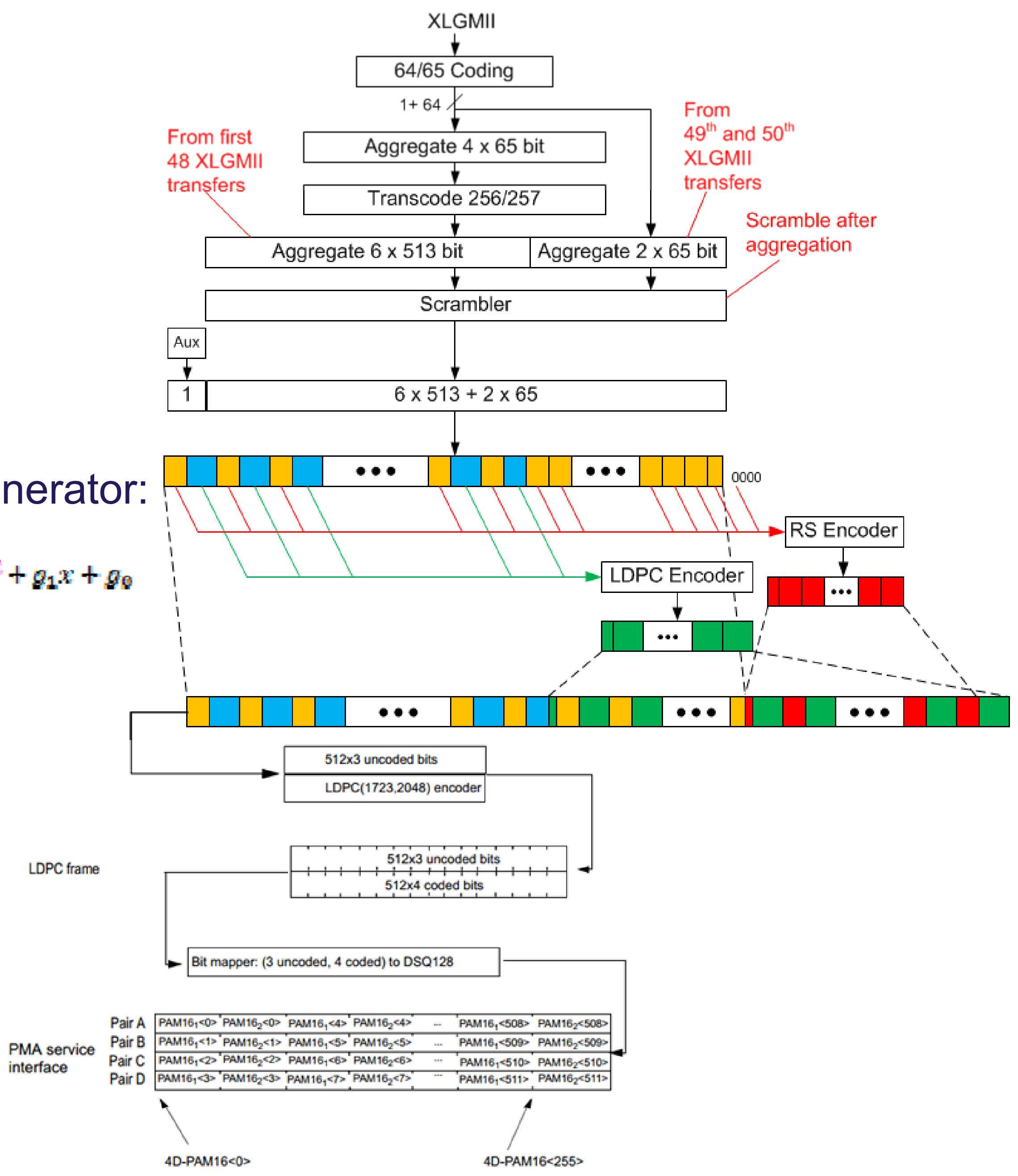
1rrrcccc	C	C	C	C	C	C	C
1rrrcccc	C	C	C	C	C	C	C
0rrrcccc	C	C	C	C	C	C	C
D	D	D	D	D	D	D	D
D	D	D	D	D	D	D	D
D	D	D	D	D	D	D	D
D	D	D	D	D	D	D	D
D	D	D	D	D	D	D	D

AQUANTIA®

Mapping

RS256(198,192) generator:

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


XLGMII
64/65 Coding
1+ 64
Aggregate 4 x 65 bit
Transcode 256/257
From first 48 XLGMII transfers
From 49th and 50th XLGMII transfers
Aggregate 6 x 513 bit
Aggregate 2 x 65 bit
Scramble after aggregation
Scrambler
Aux
1
6 x 513 + 2 x 65
0000
RS Encoder
LDPC Encoder
512x3 uncoded bits
LDPC(1723,2048) encoder
LDPC frame
512x3 uncoded bits
512x4 coded bits
Bit mapper: (3 uncoded, 4 coded) to DSQ128
PMA service interface
Pair A
Pair B
Pair C
Pair D
PAM16₁<0> PAM16₂<0> PAM16₁<4> PAM16₂<4> ... PAM16₁<508> PAM16₂<508>
PAM16₁<1> PAM16₂<1> PAM16₁<5> PAM16₂<5> ... PAM16₁<509> PAM16₂<509>
PAM16₁<2> PAM16₂<2> PAM16₁<6> PAM16₂<6> ... PAM16₁<510> PAM16₂<510>
PAM16₁<3> PAM16₂<3> PAM16₁<7> PAM16₂<7> ... PAM16₁<511> PAM16₂<511>
4D-PAM16<0>
4D-PAM16<255>

AQUANTIA

Generator

- RS256(198,192) generator:

$$g(x) = \prod_{j=0}^{5}(x - \alpha^{j}) = g_6x^6 + g_5x^5 + g_4x^4 + g_3x^3 + g_2x^2 + g_1x + g_0$$

where α is a primitive element of the finite field defined by the polynomial 0x11D = $x^8+x^4+x^3+x^2+1$

Complexity

- Using a symbol-based Euclidean decoder:
- RS2048(144,140)
  - 1048 XOR gates, 354 AND gates, 213 x 11 Flops, 295 x 11 Regfile, 1x 11-bit lookup
- RS256(198,192)
  - 1434 XOR gates, 300 AND gates, 324 x 8 Flops, 405 x 8 Regfile, 1x 8-bit lookup

- Roughly equivalent complexity

Conclusion

- With this modification we can correct 3 groups of 8 bits, versus 2 groups of 11 bits, allowing for greater and more flexible protection against uncoded bit errors