Line Code Options for 100G-EPON

www.huawei.com

Author/ Email: Duane Remein

Version: V1.0(Jan 2017)

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Contents

- **Line Code Options**
  - 256B/257B – a brief review
  - 128B/129B – a new option
- **Complexity / Power / Latency**
- **Efficiency**
- **Conclusions**

Possible Options

Option	Line Code	Defined in	Overhead
Current	64B/66B	Cl 49	3.13%
Option 1	64B/65B	Cl 55	1.56%
Option 2	128B/129B	new	0.78%
Option 3	256B/257B	Cl 91	0.39%

- **Things to consider**
  - Complexity – current is simplest, Option 3 most complex
  - Mapping – how well does the line code fit into the FEC CW
    - Avoid large padding if possible
  - Efficiency
  - Latency – probably ruled by FEC, should be considered together
  - Power
- **I assume 64B/66B & 64B/65B are well understood**
- **256B/257B covered briefly in next slides**
  - Ref 256b/257b Transcoding for 100 Gb/s Backplane and Copper Cable by Roy Cideciyan

64b/66b Coding in 100GBASE-R

- 64b/66b coding used in 100GBASE-R (IEEE 802.3ba-2010, Clause 82)
  - 1 type of data block (DB) with 2-bit header **01**
  - 11 types of control blocks (CB) with 2-bit header **1**0 where the first byte of the payload is rate-4/8 encoded (Hamming distance=4) **8-bit block type field** indicating the type of control block format

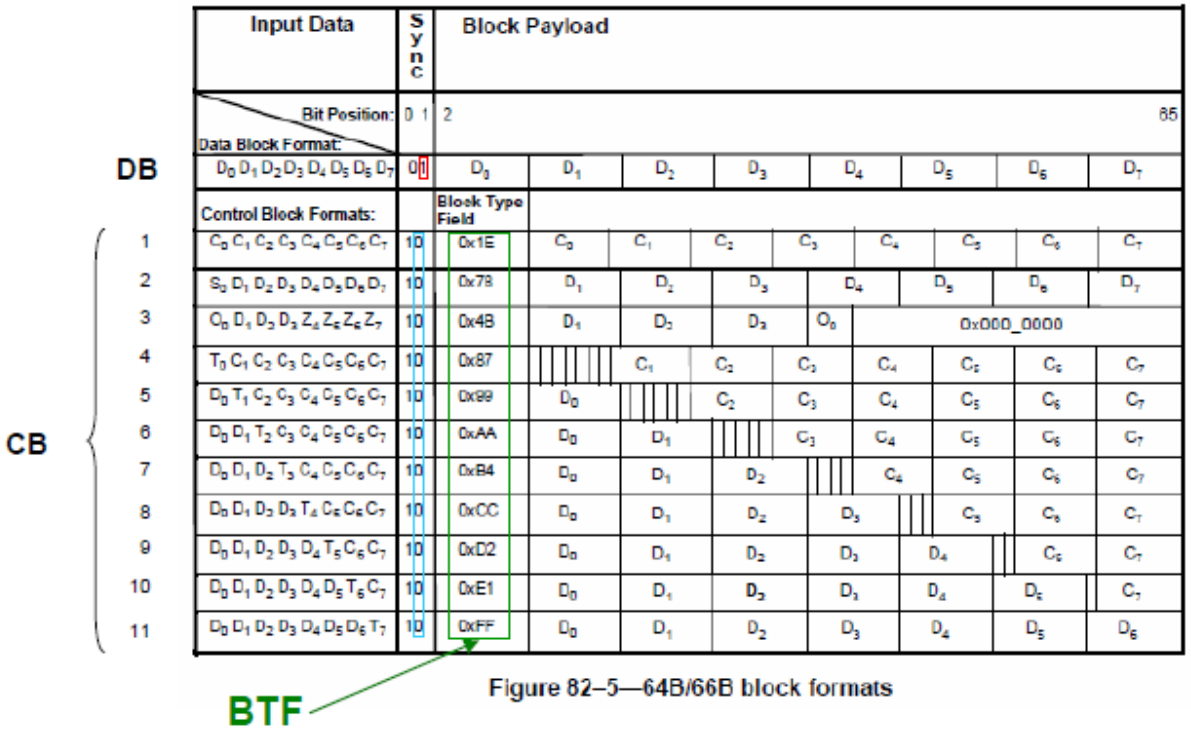


Figure 82–5—64B/66B block formats

Data Blocks and Control Blocks

- 256b/257b transcoding converts four incoming 66-bit blocks into one 257-bit block
- Four incoming 66-bit blocks that have to be transcoded may be data blocks (DB) with 2-bit header **01** or control blocks (CB) with 2-bit header **10**
- Data block #i with 64-bit block payload DBi(64) where bits are sent from left to right

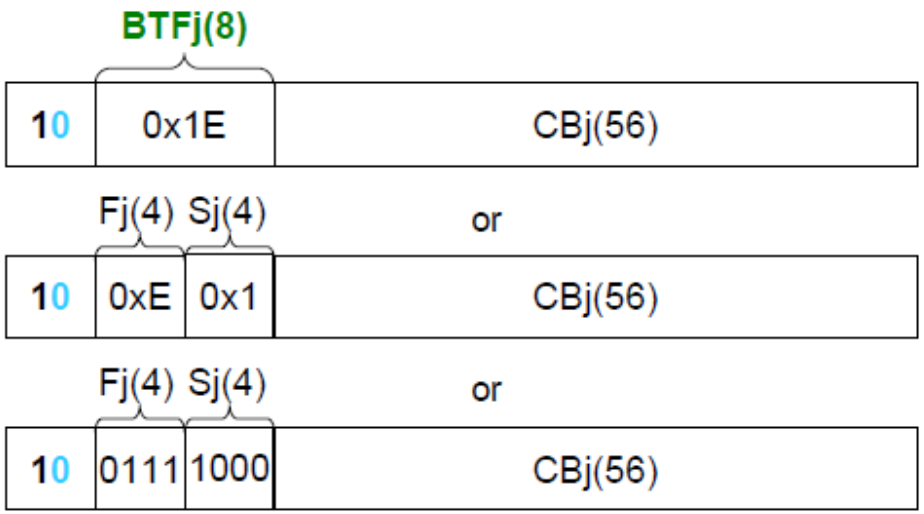
01	DBi(64)

- Control block #j with 8-bit block type field **BTFj(8)** and 56-bit control information CBj(56) where bits are sent from left to right

10	**BTFj(8)**	CBj(56)

Block Type Field

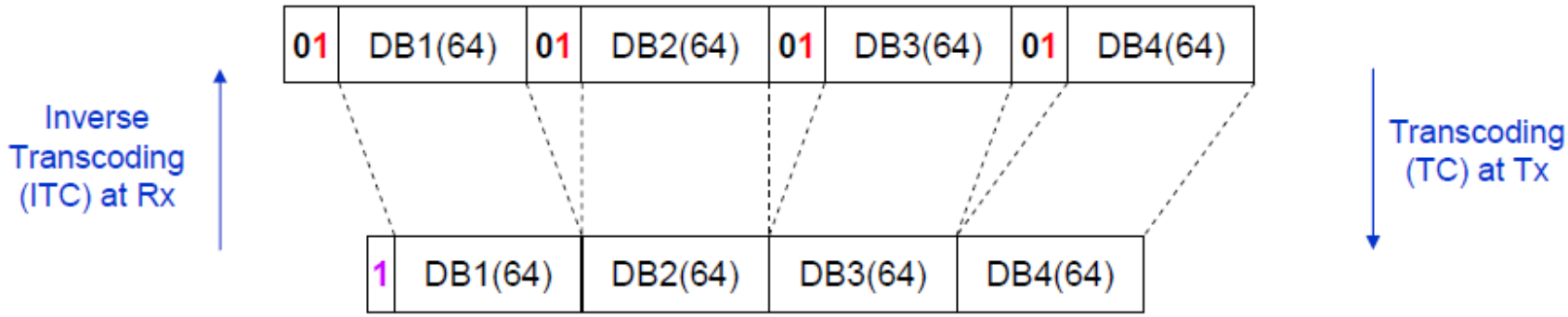
- According to clause 82.2.3.1 the LSB of the block type field (BTF) represented as a hexadecimal value is the first transmitted bit. For example, the block type field 0x1E is sent from left to right as 01111000.
- BTFj(8)=BTFj<7:0> where BTFj<0> is the first transmitted bit. We represent BTFj(8) as the concatenation of the first nibble Fj(4)=BTFj<3:0> and the second nibble Sj(4)=BTFj<7:4>. For example, for BTFj(8)=0x1E, we obtain Fj(4)=0xE sent from left to right as 0111 and Sj(4)=0x1 sent from left to right as 1000.



256b/257b Transcoding for DB-Only Payload

- *Step 1*: *Delete* **01** header bits from all DBs
- *Step 2*: *Insert* header bit 1 indicating 256-bit payload of transcoded block contains only block payloads of DBs
- DBk(64), k=1, 2, 3 and 4, indicates 64-bit block payload associated with DB #k
- Transcoded blocks are sent from left to right (leftmost bit first, rightmost bit last). Header bit of transcoded block 1 is sent first.

Only Case: DB #1, DB #2, DB #3 and DB #4

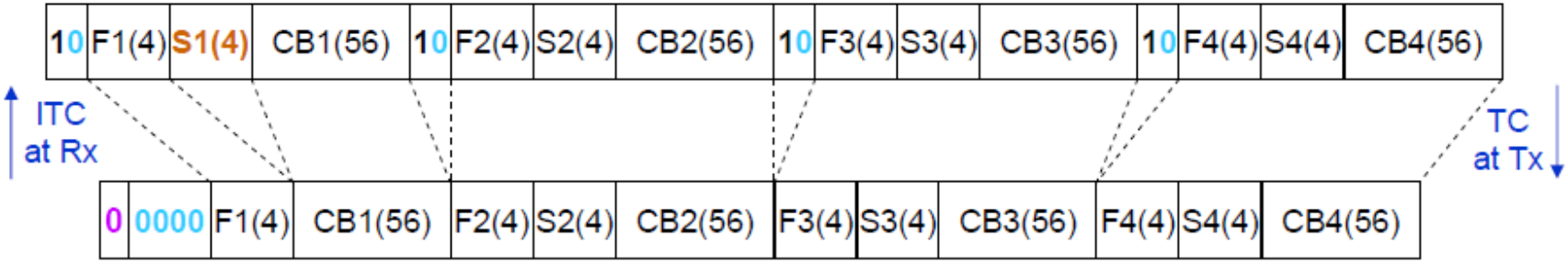


No reshuffling of the order of data blocks after transcoding

256b/257b Transcoding w/ at least 1 CB (cont.)

- Transcoded blocks are sent from left to right (leftmost bit first, rightmost bit last). Header bit of transcoded block 0 is sent first.
- Notation
  - 1st block may be a data block DB1(64) or control block F1(4), S1(4) and CB1(56)
  - 2nd block may be a data block DB2(64) or control block F2(4), S2(4) and CB2(56)
  - 3rd block may be a data block DB3(64) or control block F3(4), S3(4) and CB3(56)
  - 4th block may be a data block DB4(64) or control block F4(4), S4(4) and CB4(56)

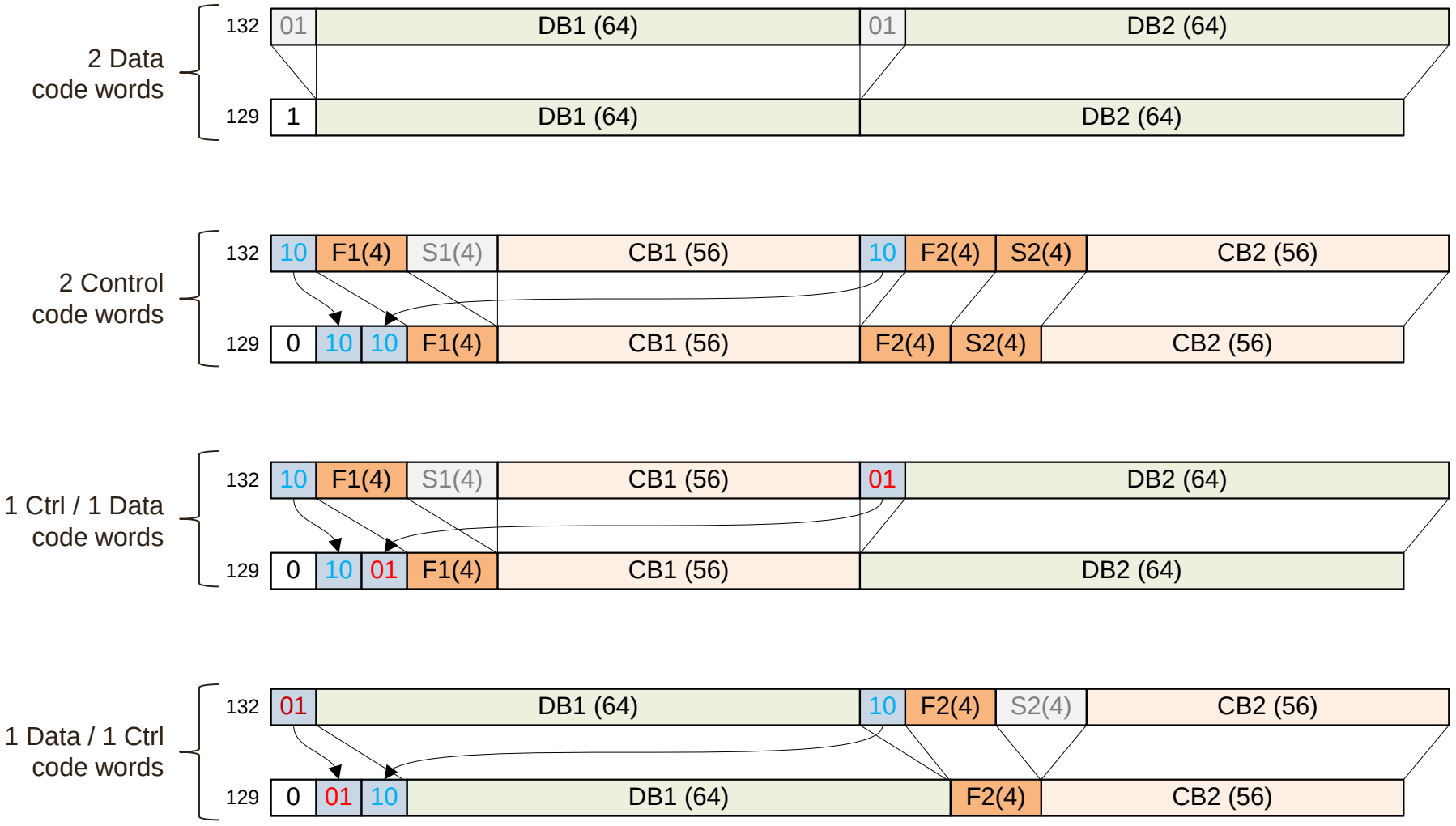
Case 1: CB #1, CB #2, CB #3 and CB #4



128B/129B line code

- **Based on 256B/257B line code**
  - Cl 91
  - 256b/257b Transcoding for 100 Gb/s Backplane and Copper Cable by Roy Cideciyan (http://www.ieee802.org/3/bj/public/mar12/cideciyan_01a_0312.pdf)

128B/129B transcoding proposal



128B/129B transcoding rules

If both 64B66B CWs are data

Delete sh (01) header from both 66B blocks

Insert single “1” header bit

Insert both 64b blocks in order

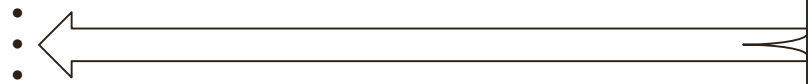
Else If one or more 64B66B CW are control

Delete sh header (10 or 01) from both blocks

Insert single “0” header bit

Insert 4b placeholder for two 2b Keywords (K_i)

For each 64b block



Next 64b block

End If

For each 64b block

If block is data

K_i = “01”

Insert 64b block

Else If first control block

K_i = “10”

Insert first 4b nibble of first byte

Delete second 4b nibble of first byte

Insert 56b control block info

Else If second control block

K_i = “10”

Insert 64b block

End If

Next 64b block

Complexity / Power / Latency

- **Complexity**
  - 64B/66B is clearly the least complex
  - 64B/65B is trivially more complex (maybe)
  - 128B/129B is slightly more complex than 64B/66B
  - 256B/257B is slightly more complex than 128B/257B
  - **None are overly complex**
- **Power**
  - Guess:
    - 64B/66B ≈ 64B/65B < 128B/129B < 256B/257B
    - 1 : 1 : 1.1 : 1.2
  - **But all are tiny compared to the overall ONU/OLT power budget**
- **Latency**
  - 64B/66B ≈ 64B/65B < 128B/129B < 256B/257B
  - 1.000 : 0.985 : 1.955 : 3.894
  - **But all will be overwhelmed by FEC**

Line code overhead

- **For large bursts and downstream traffic (which is streamed) the overhead associated with the line code will approach the minimum for that line code:**

64B/66B	3.125%	64B/65B	1.563%
128B/129B	0.781%	256B/127B	0.391%

- **In 10G-EPON the line code was 64B/66B truncated to 65B**
  - contributed 1 bit of overhead for each LC code word
  - Each RS(255,223) code word included
    - 27 x 64-bit data blocks (1728 bits)
    - 27 bits line code overhead (1.56 %) – truncated 64B/66B
    - 35 bits synchronization information (2.03 %) – 27 + 4x2
    - 256 bits parity (14.81 %)
    - Total overhead was 18.4 % (27 + 35 + 256) /1728
- **100G-EPON will need to include a synchronization mechanism also**

Conclusions

- **Select a FEC that will provide the needed gain**
- **Then select the largest line code that fits well into the selected FEC payload with a smidgen of room left over for sync bits.**

Thank you

www.huawei.com

Clause 49 mapping

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	0 1	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	1 0	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	1 0	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	1 0	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	1 0	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	1 0	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	1 0	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	1 0	0x4b	D_1	D_2	D_3	O_0	C_4	C_5	C_6	C_7
T_0 D_1 D_2 D_3 / C_4 C_5 C_6 C_7	1 0	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	1 0	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	1 0	0xaa	D_0	D_2		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	1 0	0xb4	D_0	D_2	D_3		C_4	C_5	C_6	C_7
D_0 D_1 D_2 D_3 / T_4 C_5 C_6 C_7	1 0	0xcc	D_0	D_2	D_3	D_4		C_5	C_6	C_7
D_0 D_1 D_2 D_3 / D_4 T_5 C_6 C_7	1 0	0xd2	D_0	D_2	D_3	D_4	D_5		C_6	C_7
D_0 D_1 D_2 D_3 / D_4 D_5 T_6 C_7	1 0	0xe1	D_0	D_2	D_3	D_4	D_5	D_6		C_7
D_0 D_1 D_2 D_3 / D_4 D_5 D_6 T_7	1 0	0xff	D_0	D_2	D_3	D_4	D_5	D_6	D_7