# Alignment Marker Mapping and Insertion - Supporting Material for Comment #3 and #2

Adrian Butter (on behalf of the P802.3bs Logic Ad Hoc team)

GLOBALFOUNDRIES®

# Contributors

- Mark Gustlin, Xilinx
- Pete Anslow, Ciena
- Ben Jones, Xilinx
- Jeff Slavick, Avago Technologies
- Eric Baden, Broadcom
- Juan Carlos Calderon, Inphi
- David Ofelt, Juniper Networks
- Tongtong Wang, Huawei
- Phil Sun, Credo Semiconductor

# Supporters

- Gary Nicholl, Cisco

# Introduction

- Alignment Marker format definition is one of the Logic Ad Hoc BTIs to resolve in support of a technically complete P802.3bs draft:

  - Sublayer delay constraints are TBD, same with skew limitations
  - Define 400G AM fields
    - Made progress, but some more work to do on this
  - Exact criteria for achieving AM lock
    - AMP_valid

- Since December 2015, several Logic Ad Hoc team members have made ***excellent contributions*** towards resolving this BTI! For example:
  - Toward 400GbE AMs and PAM4 test pattern characteristics
  - 400GbE AMs revised proposal
  - Proposed AM Format
- Logic Ad Hoc team consensus has been attained on Alignment Marker formatting.
- Comment #3 was logged against P802.3bs D1.2 to support technical updates capturing this consensus.
  - Comment #2 was also logged identifying an ancillary issue found during this activity.
  - As a bonus, the proposed response to comment #69 is also included...
- Editing details supporting these technical updates are contained in “butter_3bs_02*a*_0316”.
- This presentation identifies those updates with supporting material...

**Note:** Slide changes based on suggestions received during the 23-Feb-2016 400Gb/s Logic Ad Hoc call are denoted with in the upper right corner.

**Note:** Slide changes since submission of butter_3bs_01_0316 are denoted with in the upper right corner.

# D1.2 Comments Regarding Alignment Markers

*Cl* **119** *SC* **119.2.4.4** *P* **98** *L* **6** # 3

Butter, Adrian — GLOBALFOUNDRIES

*Comment Type* **TR** *Comment Status* **X**

The alignment marker encodings in Table 119-1 contain many "TBDs". Further analysis of this alignment marker structure (with 64-bit common part and 56-bit unique part) reveals undesirable clock content which is reduced using a shorter alignment marker (with 48-bit common part and 48-bit unique part). To reduce the complexity of alignment marker processing logic for the shorter marker, as well as increase format compability of the shorter marker with that defined in 802.3bj, padding based on PRBS9 sequences is both interleaved with and appended to the marker. Refer to http://www.ieee802.org/3/bs/public/adhoc/logic/feb9_16/gustlin_01_0216_logic.pdf for details.

*Cl* **119** *SC* **119.2.4.5** *P* **100** *L* **32** # 2

Butter, Adrian — GLOBALFOUNDRIES

*Comment Type* **T** *Comment Status* **X**

There is no clear connection between variables tx_scrambled_am and tx_scrambled_am_j. Also, defining tx_scrambled_am as 257 bits does not align with the width implied in 119.2.4.4, page 97, line 25.

*Cl* **119** *SC* **119.2.4.4** *P* **97** *L* **15** # 69

Ran, Adee — Intel

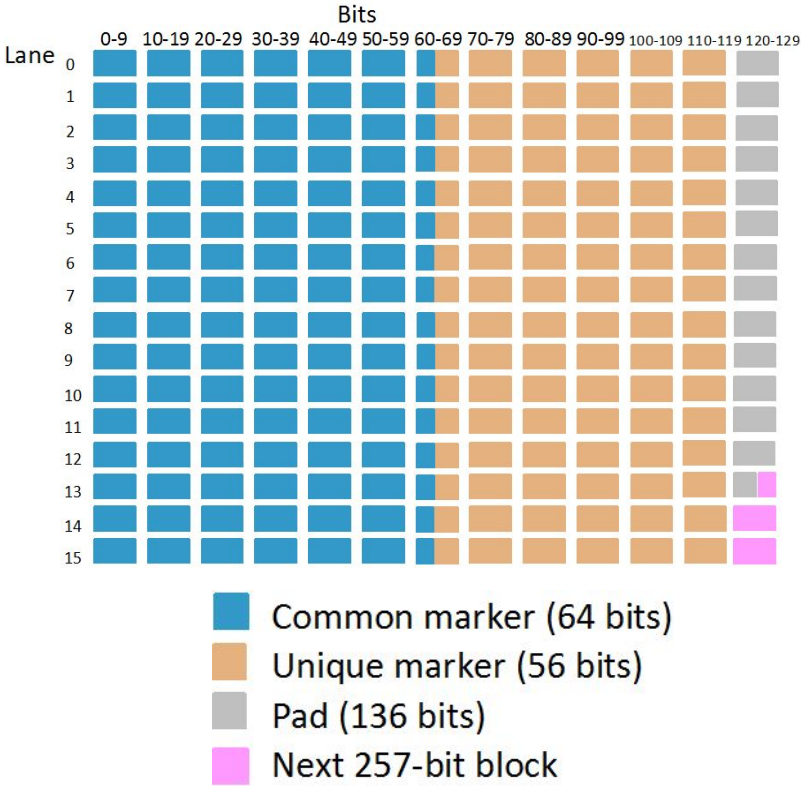
*Comment Type* **T** *Comment Status* **X**

"The pad shall not be checked on receive" - no PICS - and why is that a normative requirement? Should if be verified? How?

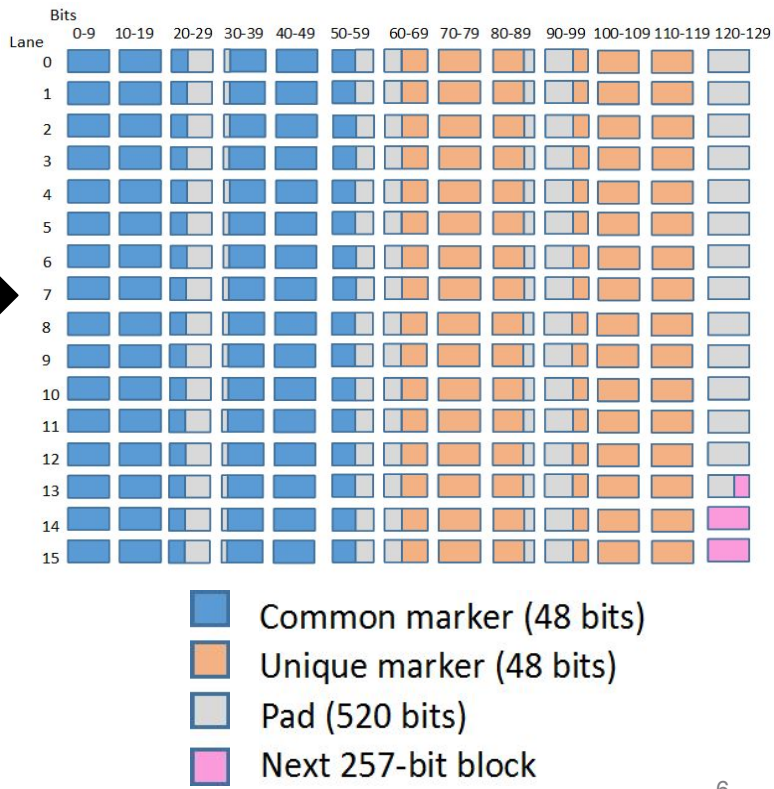
This subclause describes insertion so receive operation is out of place here.

# Objectives

Change alignment marker format from:

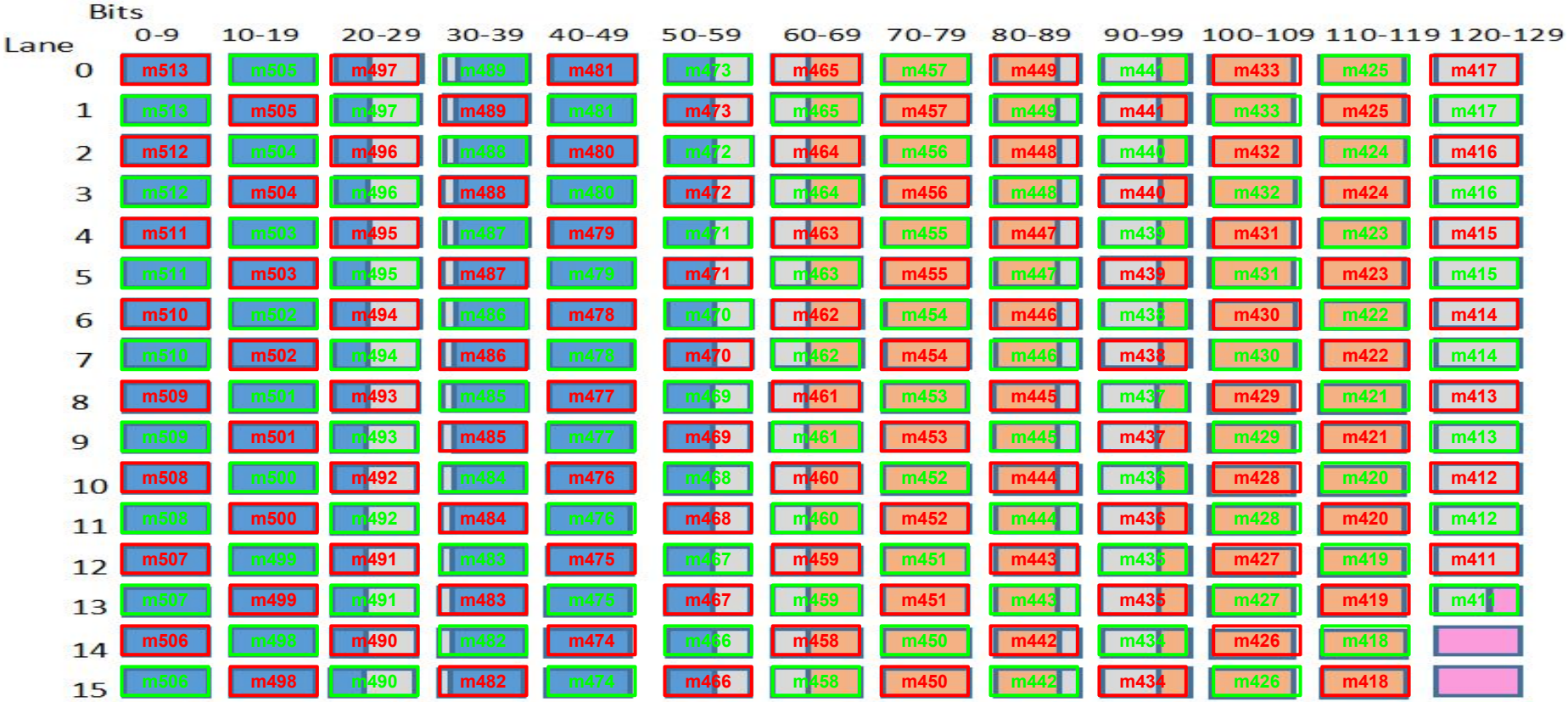



… to:



# Objectives

Map alignment markers to FEC codewords in the following manner:

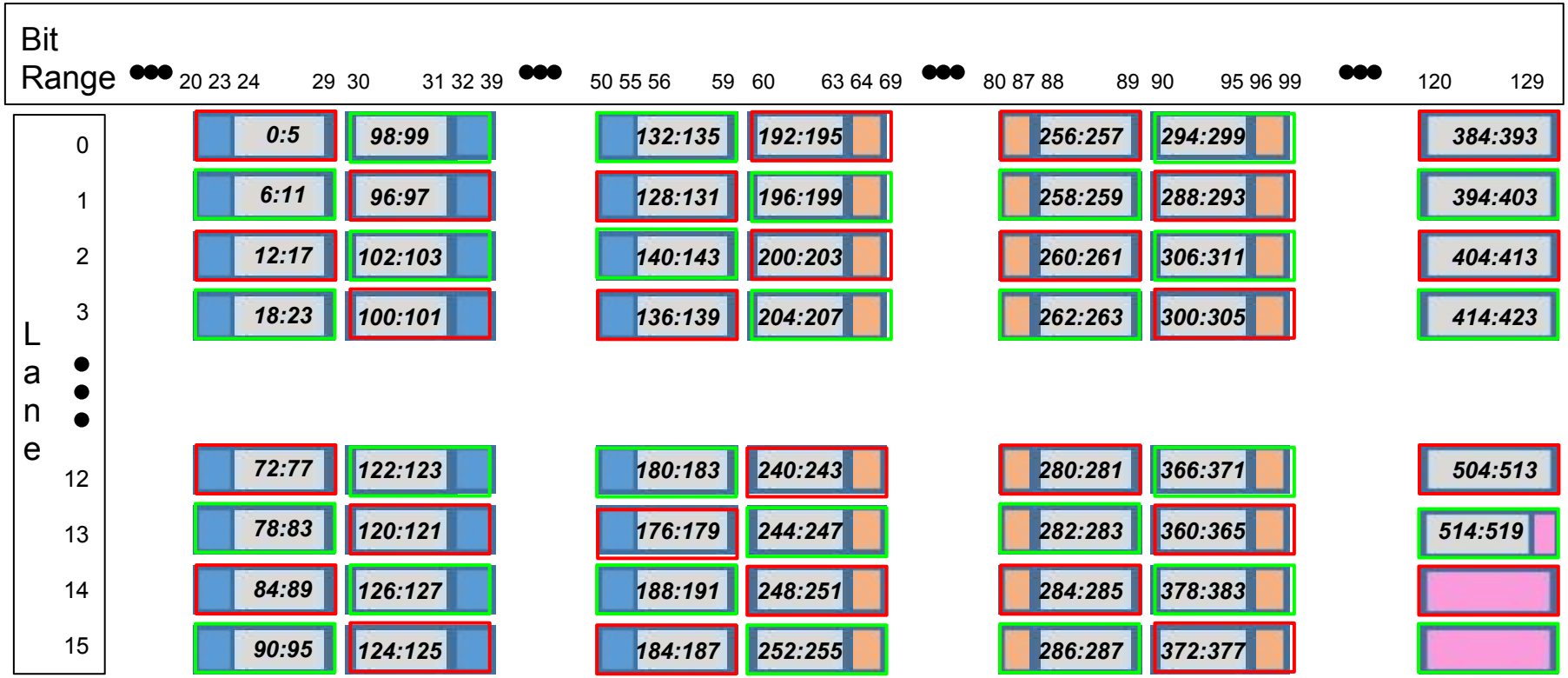



mNNN 10-bit Message symbol NNN of FEC codeword A.

mNNN 10-bit Message symbol NNN of FEC codeword B.

# Objectives

PRBS9 padding is organized in the following manner:




*x:y* = PRBS9 bit range for the current Alignment Marker insertion operation (bit 0 generated first).

10-bit Message symbol of FEC codeword A.

10-bit Message symbol of FEC codeword B.

# Comment #3 and #69 Update - 119.2.4.4 Paragraph 1

**119.2.4.4 Alignment marker mapping and insertion**

In order to support deskew and reordering of the 16 individual PCS lanes at the receive PCS, alignment markers are added periodically for each PCS lane. The alignment marker for each PCS lane is *composed of* a ~~unique 120~~ fixed 96-bit block *interleaved with 24 pad bits to achieve alignment marker field positioning identical to that defined in 91.5.2.6. An alignment marker group is composed of the alignment markers for all PCS lanes plus an additional 136-bit pad to yield the equivalent of eight 257-bit blocks.* The alignment marker~~s for all PCS lanes are inserted as a group,~~ *group is* aligned to the beginning of ~~a~~ two FEC *messages*~~block~~, and interrupt any data transfer that is already in progress. ~~*A 136-bit pad is appended to the alignment markers to yield the equivalent of eight 257-bit blocks.*~~ The pad bits shall be set to a free running PRBS9 pattern, defined by the polynomial $x^9 + x^5 + 1$. The initial value of the PRBS9 pattern generator may be any pattern other than all zeros. The pad *contents are ignored*~~shall not be checked~~ on receive.

Proposed response to comment #69...

**Note:** Changes since 23-Feb-2016 400Gb/s Logic Ad Hoc call appear in *red italicized* text.

# Comment #3 Update - 119.2.4.4 Paragraph 2

Room for the alignment marker~~s~~ *group* is created by the transmit PCS (see 119.2.4.1). Special properties of the alignment marker~~s are~~ *group is* that *it is*~~they are~~ not scrambled, do*es* not conform to the encoding rules as outlined in Figure 82-5 and *is*~~are~~ not transcoded. This is possible because the alignment marker~~s are~~ *group is* added after encoding, transcoding, and scrambling, and removed before descrambling, transcoding, and 64B/66B decoding. The alignment marker~~s are~~ *group is* not scrambled, *which allows* ~~in order to allow~~ the receiver to *directly search for and* find the *individual* alignment markers, deskew the PCS lanes, and reassemble the aggregate stream before descrambling is performed. *Additionally,* ~~T~~*t*he *fixed 96-bit portion of the* alignment markers *is*~~themselves are~~ formed from a known pattern that is defined to be balanced and with many transitions and therefore scrambling is not necessary. ~~The group of alignment markers shall be inserted so they appear every 163 840 257 bit blocks. The variable tx_scrambled_am is created by inserting the group of alignment markers in the variable tx_scrambled. Alignment marker mapping and repetition rate are shown in Figure 119–5 and Figure 119–6.~~

**Note:** Changes since 23-Feb-2016 400Gb/s Logic Ad Hoc call appear in *red italicized* text.

# Comment #3 Update - 119.2.4.4 Paragraph 3, Figure 119-4

The format of ~~the~~ each PCS lane's alignment marker~~s~~ is shown in Figure 119–4. There is a portion that is common across all alignment markers (designated as $CM_0$ to $CM_{75}$), and then a unique portion per PCS lane (designated as $UM_0$ to $UM_{65}$). Common synchronization logic independent of the received PCS lane number can be used with the common portion~~s~~ of the alignment marker.

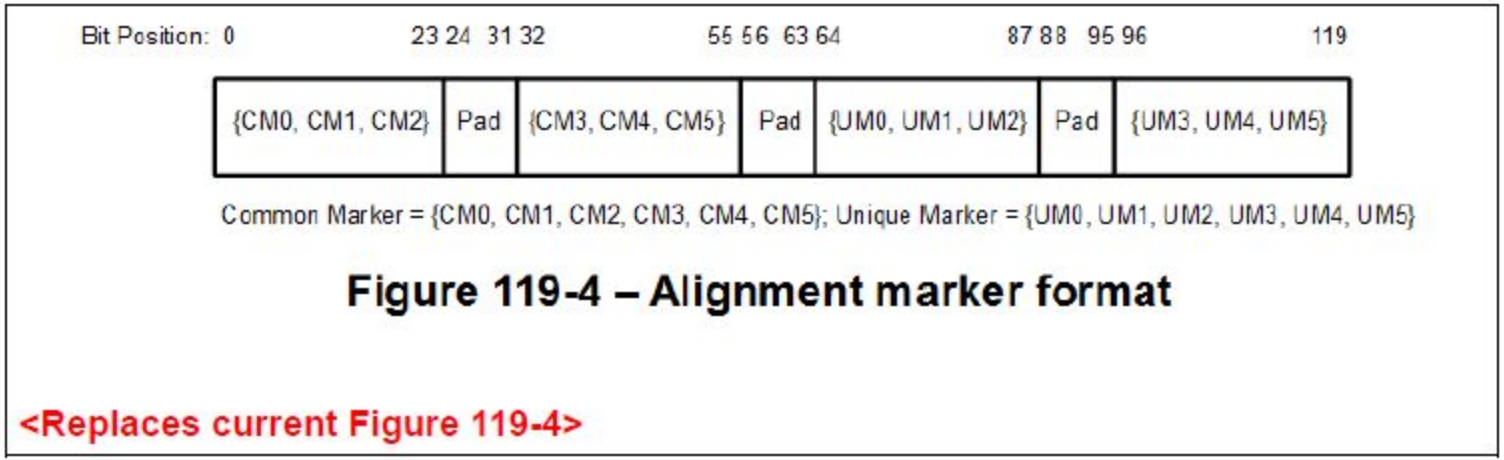



**Figure 119-4 – Alignment marker format**

**<Replaces current Figure 119-4>**

**Note:** Changes since 23-Feb-2016 400Gb/s Logic Ad Hoc call appear in *red italicized* text.

# Comment #3 Update - 119.2.4.4 Paragraph 4, Table 119-1

The content of the *fixed 96-bit portion of the* alignment markers shall be as shown in Table 119–1. The contents depend on the PCS lane number and the octet number, with the first ~~64~~ 48 bits being identical across all alignment markers to allow for common synchronization across lanes. The format shown in Table 119–1 *defines* ~~*is*~~ how the alignment markers appear on the PCS lanes at the PMA service interface. In the FEC codewords, they appear in a permuted format due to the codeword interleaving that occurs before FEC codewords are distributed to PCS lanes.

**Table 119-1 – *Encodings for fixed 96-bit portion of* 400GBASE-R Alignment marker ~~*encodings*~~**

| PCS lane number | Encoding {CM0 , CM1 , CM2 , CM3 , CM4 , CM5 , UM0 , UM1 , UM2 , UM3 , UM4 , UM5} |
|---|---|
| 0 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x9E , 0xEB , 0x27 , 0x61 , 0x14 , 0xD8 |
| 1 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x50 , 0x74 , 0x88 , 0xAF , 0x8B , 0x77 |
| 2 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0xB4 , 0xB7 , 0xEA , 0x4B , 0x48 , 0x15 |
| 3 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0xE4 , 0xFB , 0xF1 , 0x1B , 0x04 , 0x0E |
| 4 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0xDC , 0x58 , 0xEE , 0x23 , 0xA7 , 0x11 |
| 5 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0xBD , 0xA9 , 0xBF , 0x42 , 0x56 , 0x40 |
| 6 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x97 , 0x67 , 0x77 , 0x68 , 0x98 , 0x88 |
| 7 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x24 , 0x35 , 0xA5 , 0xDB , 0xCA , 0x5A |
| 8 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x57 , 0x64 , 0x51 , 0xA8 , 0x9B , 0xAE |
| 9 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x28 , 0xF9 , 0x3E , 0xD7 , 0x06 , 0xC1 |
| 10 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0xCB , 0xD1 , 0xAD , 0x34 , 0x2E , 0x52 |
| 11 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x5E , 0x1E , 0x38 , 0xA1 , 0xE1 , 0xC7 |
| 12 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x19 , 0x98 , 0xF9 , 0xE6 , 0x67 , 0x06 |
| 13 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x84 , 0xEC , 0x20 , 0x7B , 0x13 , 0xDF |
| 14 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x13 , 0xA4 , 0xED , 0xEC , 0x5B , 0x12 |
| 15 | 0x9A , 0x4A , 0x26 , 0x65 , 0xB5 , 0xD9 , 0x3F , 0x8A , 0xBE , 0xC0 , 0x75 , 0x41 |

**Note:** Changes since 23-Feb-2016 400Gb/s Logic Ad Hoc call appear in *red italicized* text.

# Comment #3 Update - 119.2.4.4 Paragraph 5+ (New Content)

The alignment marker mapping function *creates* a *set* of 16 alignment markers, *and in combination with an additional 136-bit PRBS9 pad generates an alignment marker group.* Let am_x<119:0> be the alignment marker for PCS lane x, x=0 to 15, where bit 0 is the first bit transmitted. The alignment markers shall be mapped to am_mapped<1919:0> in a manner that yields the same result as the following process.

For x=0 to 15, am_x<119:0> is constructed as follows.

a) am_x<23:0> is set to $CM_0$, $CM_1$, and $CM_2$, as shown in Figure 119-4 (bits 23:0) using the values in Table 119-1 for PCS lane number x.

b) *if even(x)*
am_x<31:24>={PRBS9<2*x+99:2*x+98>, PRBS9<6*x+5:6*x>}
*else*
am_x<31:24>={PRBS9<2*x+95:2*x+94>, PRBS9<6*x+5:6*x>}
As shown in Figure 119-4 (bits 31:24) is an 8-bit pad value of PRBS9 pattern bits, where bit 6*x is the first PRBS9 bit output of the 8-bit pad.

c) am_x<55:32> is set to $CM_3$, $CM_4$, and $CM_5$, as shown in Figure 119-4 (bits 55:32) using the values in Table 119-1 for PCS lane number x.

d) *if even(x)*
am_x<63:56>={PRBS9<4*x+195:4*x+192>, PRBS9<4*x+135:4*x+132>}
*else*
am_x<63:56>={PRBS9<4*x+195:4*x+192>, PRBS9<4*x+127:4*x+124>}
As shown in Figure 119-4 (bits 63:56) is an 8-bit pad value of PRBS9 pattern bits, where bit 4*x+128 is the first PRBS9 bit output of the 8-bit pad.

e) am_x<87:64> is set to $UM_0$, $UM_1$, and $UM_2$, as shown in Figure 119-4 (bits 87:64) using the values in Table 119-1 for PCS lane number x.

f) *if even(x)*
am_x<95:88>={PRBS9<6*x+299:6*x+294>, PRBS9<2*x+257:2*x+256>}
*else*
am_x<95:88>={PRBS9<6*x+287:6*x+282>, PRBS9<2*x+257:2*x+256>}
As shown in Figure 119-4 (bits 95:88) is an 8-bit pad value of PRBS9 pattern bits, where bit 2*x+256 is the first PRBS9 bit output of the 8-bit pad.

g) am_x<119:96> is set to $UM_3$, $UM_4$, and $UM_5$, as shown in Figure 119-4 (bits 119:96) using the values in Table 119-1 for PCS lane number x.

# Comment #3 Update - 119.2.4.4 Paragraph 5+ (New Content)

As an example, the ~~is sent as (left most bit sent first) lane marker for 400GBASE-R lane number 0~~ variable am_0 is *sent as (left most bit sent first):*

~~10000011 00010110 10000100 00101111 01111100 01111001 01111011 11010000 TBD~~
*01011001 01010010 01100100 <PRBS9(0:5), PRBS9(98:99)> 10100110 10101101 10011011 <PRBS9(132:135), PRBS9(192:196)> 01111001 11010111 11100100 <PRBS9(256:257), PRBS9(294:299)> 10000110 00101000 00011011*

**Note:** Changes since 23-Feb-2016 400Gb/s Logic Ad Hoc call appear in *red italicized* text.

# Comment #3 Update - 119.2.4.4 Paragraph 5+ (New Content)

The variable am_mapped is then derived from 10-bit interleaving the group of 16 alignment markers am_x per the following procedure.

For all k=0 to 11
    For all j=0 to 7
        if even(k)
            am_mapped<160*k+20*j+9:160*k+20*j> = am_{2*j}<10*k+9:10*k>
            am_mapped<160*k+20*j+19:160*k+20*j+10> = am_{2*j+1}<10*k+9:10*k>
        else
            am_mapped<160*k+20*j+9:160*k+20*j> = am_{2*j+1}<10*k+9:10*k>
            am_mapped<160*k+20*j+19:160*k+20*j+10> = am_{2*j}<10*k+9:10*k>

The additional 136-bit pad is appended to variable am_mapped as follows.

am_mapped<2055:1920> = PRBS9<519:384>

In this expression, PRBS9<384> is the first PRBS9 bit output of the 136-bit pad.

# Comment #3 and #2 Update - 119.2.4.4 Paragraph 5+ (New Content)

The alignment marker *group am_mapped<2055:0>* shall be inserted so *it* appears every 163 840 257-bit blocks. The variable tx_scrambled_am<10279:0> is constructed in one of two ways. Let the set of vectors tx_scrambled_i<256:0> represent consecutive values of tx_scrambled<256:0>. For a *10280-bit* block with *an* alignment marker *group* inserted:

tx_scrambled_am<2055:0> = am_mapped<2055:0>
for all i=0 to 31
    tx_scrambled_am<257*i+2312:257*i+2056> = tx_scrambled_i<256:0>

For a *10280-bit* block without *an* alignment marker *group*:

for all i=0 to 39
tx_scrambled_am<257*i+256:257*i> = tx_scrambled_i<256:0>

Alignment marker mapping and repetition rate are shown in Figure 119–5 and Figure 119–6.

**Note:** Changes since 23-Feb-2016 400Gb/s Logic Ad Hoc call appear in *red italicized* text.

# Comment #3 and #2 Update - 119.2.4.5

**119.2.4.5 Pre-FEC Distribution**

~~*Two Reed Solomon FEC codewords are interleaved before data is distributed to the PCS lanes to improve error correction capability. Data is distributed to two 5140-bit message blocks ($m_A$ and $m_B$ are both arrays of 514 10-bit symbols) by performing a 10-bit round robin distribution of the tx_scrambled_am<256:0> data as follows.*~~ *In order to improve error correction capability, each set of two consecutive Reed-Solomon FEC codewords is interleaved before being distributed to form the PCS lanes. To enable this interleaving, the Pre-FEC Distribution function receives a 10280-bit block tx_scrambled_am, and performs a 10-bit symbol round robin distribution to form two 514-symbol FEC messages, which are subsequently each encoded by the RS FEC. The following describes the 10-bit round robin distribution process.*

~~For all j=0 to 39, tx_temp<10279:0> shall be constructed as follows:~~

~~tx_temp<(257j+256):(257j)> = tx_scrambled_am_j<256:0>~~

For all i=0 to 513~~, $m_A$<513:0> and $m_B$<513:0> shall be constructed as follows:~~

$m_A$<(513-i)> = ~~tx_temp~~ tx_scrambled_am<(20*i+9):(20*i)>

$m_B$<(513-i)> = ~~tx_temp~~ tx_scrambled_am<(20*i+19):(20*i+10)>

**Note:** Changes since 23-Feb-2016 400Gb/s Logic Ad Hoc call appear in *red italicized* text.

# Thanks!

GLOBALFOUNDRIES®