Preventing clock content issues in 800GBASE-R PCS/PMA

(In support of comments #21, #74 against D1.1)

Adee Ran, Cisco

Support

- Piers Dawe, NVIDIA

Outline

- Recap: clock content issue in 200G/400G
- New possible clock issue in 800G
- Proposed solution

Clock content issue

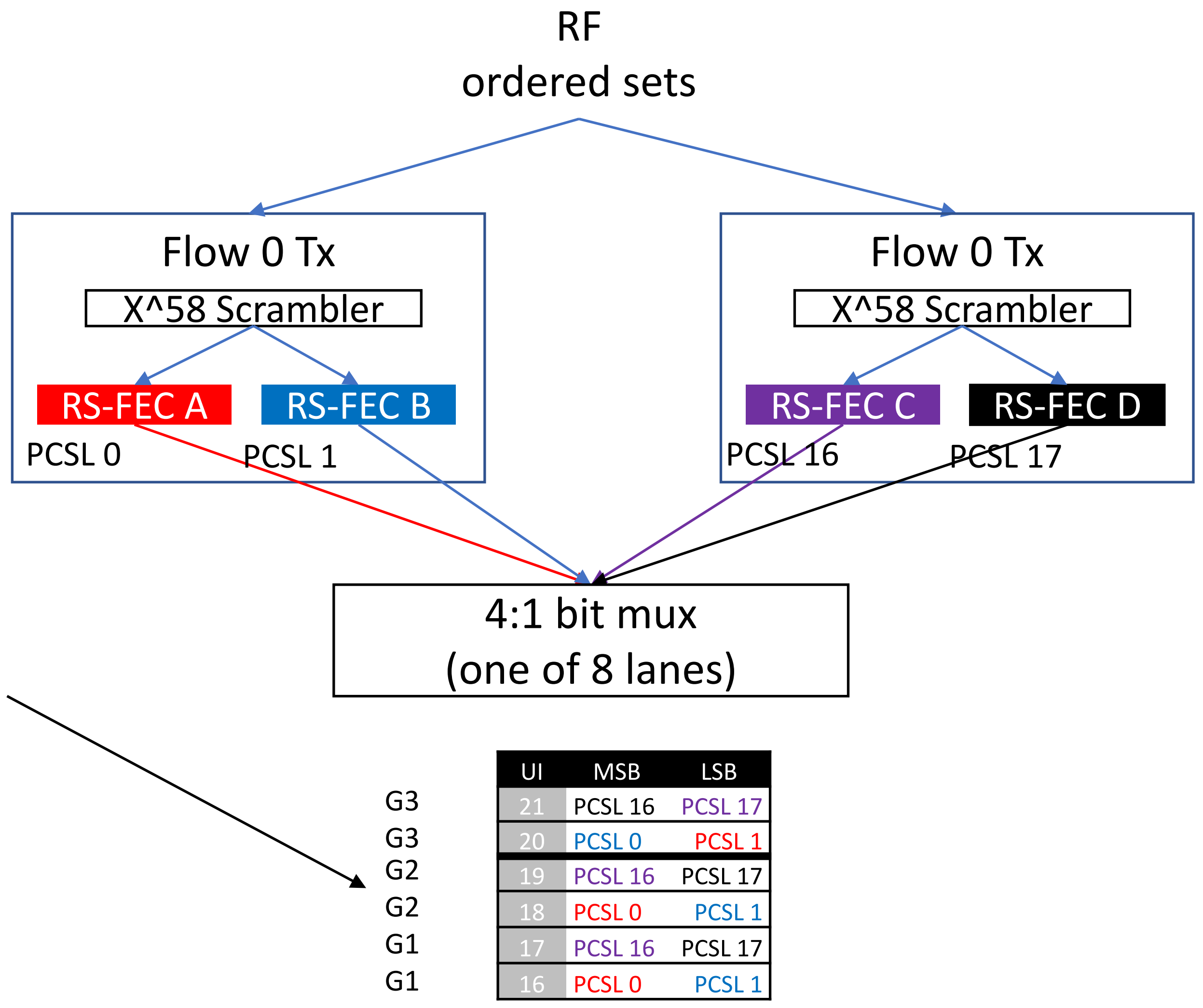
- Discussed in 802.3bs for 400G and 200G PCS with 4:1 bit muxing
  - See anslow_01_121916_elect, wong_01_0117_logic, and wertheim_3bs_01_0317
- Brief explanation
  - Due to Gray mapping, the LSBs of PAM4 symbols are created by XOR between pairs of bits
  - With bit muxing, each pair of bits comes from two different PCSLs
  - With a constant input, the scrambler generates a PRBS-like bit sequence
  - Other than the RS-FEC parity bits and AMs, the outputs of the PCSLs at each UI are samples of the PRBS with different offsets
  - XOR between two shifted versions of a PRBS creates a PRBS with another shift
  - With 4:1 muxing, we get **two pairs** of delayed versions of a PRBS – and there are 3 degrees of freedom in choosing the delays
  - With **some combinations of delays**, we get the same delayed PRBS for both pairs – causing no LSB change within a pair of UIs – and “low clock content”

Can this problem appear in 800GBASE-R?

- Short answer: No.
  - In "Option A" (see ran_3df_01a_2212), alternate LSBs are created from a pair of bits from different PCS flows – two separate scramblers.
    - To make pairs of LSBs equal, the two scramblers need to be in the same state.
    - Since the scrambler is multiplicative, its state depends on the input. The probability of the two scramblers randomly reaching the same state is negligible when the input histories differ even by a single bit.
  - There is also no correlation between "every other LSB" (LSBs at 2 UI distance)
    - Although they come from a single scrambler – it's always just one pair of lanes (e.g. 0 and 16)
    - As explained in wertheim_3bs_01_0317, correlation can occur in data from two pairs of lanes, but not within one pair of lanes.
  - In "Option B" this also practically never happens
    - However, Option B should not be allowed due to its FLR impact.
- Long answer: No, **unless the two scramblers are at the same state**.

New potential problem

- The initial PCS input is a periodic ordered set (Remote Fault)
- After transcoding, this creates the same input for both scramblers – a periodic 257-bit pattern
- If scrambler initial states are the same, their output sequences will also be the same
  - PCSL 0 = PCSL 16, PCSL 17 = PCSL 1…
- With a valid (and likely) muxing choice where the same PCSLs from both flows are muxed together to a single lane (such as: 0, 1, 16, 17), **each pair can consist of two identical PAM4 symbols…**
  - Not only the LSBs but also the MSBs are equal
  - Effectively creating a 26.5625 GBd PAM4 stream (instead of 53.125 GBd)
- Other muxing choices can lead to different problems…
  - Equal MSB and LSB
  - Skew-dependent “clock content” issues as in 400G



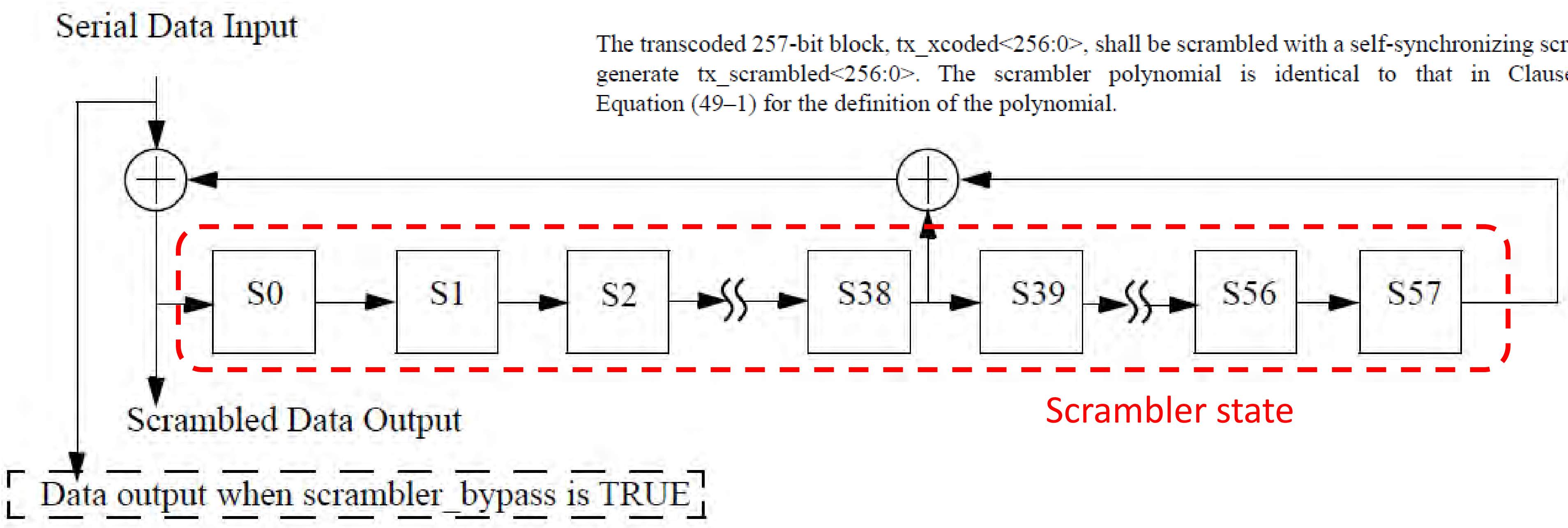
	UI	MSB	LSB
G3	21	PCSL 16	PCSL 17
G3	20	PCSL 0	PCSL 1
G2	19	PCSL 16	PCSL 17
G2	18	PCSL 0	PCSL 1
G1	17	PCSL 16	PCSL 17
G1	16	PCSL 0	PCSL 1

PCS scramblers

Each scrambler is described by the binary linear feedback shift register (LFSR) representation shown below.

119.2.4.3 Scrambler

The transcoded 257-bit block, tx_xcoded<256:0>, shall be scrambled with a self-synchronizing scrambler to generate tx_scrambled<256:0>. The scrambler polynomial is identical to that in Clause 49, see Equation (49–1) for the definition of the polynomial.



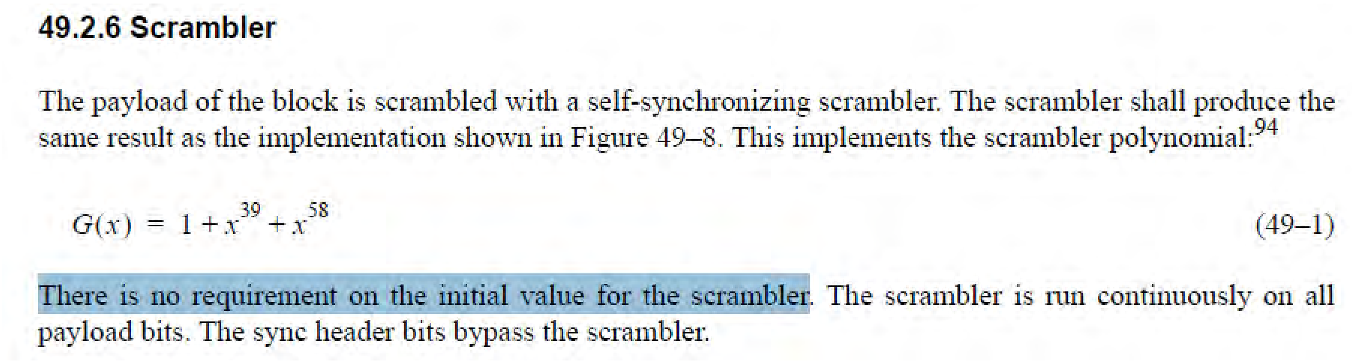
NOTE—Scrambler_bypass is only required to support EEE capability.

Figure 49–8—Scrambler

Solution: prevent scrambler from having equal seeds

- The LFSR state is a function of both the previous state and the input.
- With zero input, the LFSR period is $2^{58} - 1 \approx 3 \times 10^{17}$ bit times.
- With a constant non-zero transcoded scrambler input (a periodic sequence with a period of 257 bits), the output will also be periodic
  - 257 is a prime number, and is not a factor of $2^{58} - 1$, so the periods are coprime; the combined period is 257 times longer, $\approx 10^{20}$ bit times
  - With a bit time of 2.5 ps (per flow), the period is about 6 years.
  - Different scrambler seeds will create different "random" offsets of the periodic output sequence.
  - For a given seed, finding another seed that, with the same input, creates an output with a small offset is "hard".
- With the same input, the probability that two different initial scrambler states would create outputs separated by a small delay is negligible.
- **Practically, different initial seeds should completely eliminate the problem.**

Can we specify different reset values?



49.2.6 Scrambler

The payload of the block is scrambled with a self-synchronizing scrambler. The scrambler shall produce the same result as the implementation shown in Figure 49–8. This implements the scrambler polynomial:[94]

$$G(x) = 1 + x^{39} + x^{58} \qquad (49\text{–}1)$$

There is no requirement on the initial value for the scrambler. The scrambler is run continuously on all payload bits. The sync header bits bypass the scrambler.

- A self-synchronizing scrambler does not require initialization.
- It can be implemented with no reset function…
  - "Random" initial values would eliminate the issue
- However, if reset is implemented, synchronous reset of both scramblers to the same value would be bad.
  - We should address possible implementations that reset the scrambler.

Proposed change

172.2.4.3 Scrambler

The scrambler in each flow is identical to that specified in 119.2.4.3.

Although there is no requirement on the initial value of each scrambler, if an implementation sets the scrambler state to a fixed value (e.g., when reset is asserted), the two scramblers should be set to different states.