FEC Framing and Synchronization

Jeff Mandin
PMC-Sierra
jeff_mandin@pmc-sierra.com

Contribution to 802.3av FEC Framing Adhoc
Monterey - Jan 2007

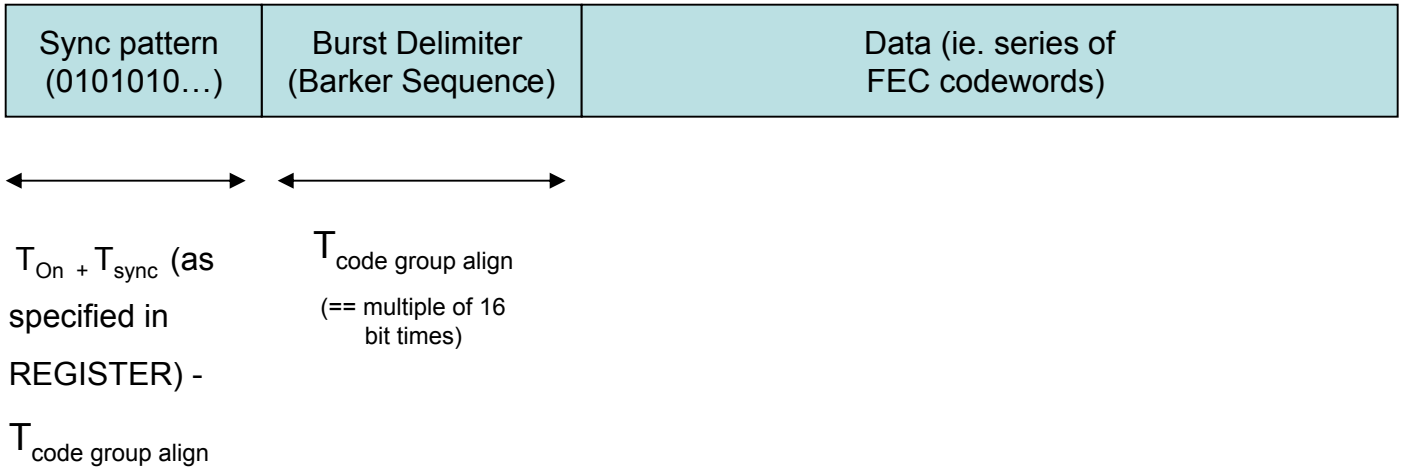
Agenda

1. Upstream
   a) Frame structure
   b) Logical Framework (ie. PCS stack) at ONU
   c) Synchronization sequence

2. Downstream
   a) Option 1: 2bit sync header
      I. Frame Structure
      II. Synchronization sequence
   b) Option 2: Serial lock
      I. Frame Structure
      II. Synchronization sequence

Upstream

Upstream Burst Frame (as xmitted over PMA at ONU)



Observations on PCS and Data Detector for 64b/66b code

1. Data detector's job: to turn on the laser in advance of the actual data transmission and turn it off afterward
    - Must interpret the GMII/XGMII control and data codes to do this
    - In 802.3ah the data detector resides at the bottom of the stack (ie. after IDLE deletion) and decodes the 10bit symbol
2. In 10GEPON we need a new logical function (either co-located with the data detector or separate from it) to manage the control sequences, transmissions, and state transitions for AGC/CDR and Burst Mode Frame Synchronization
    - Call this function the *burst mode synchronizer*
3. PCS layer *behaves differently* before frame synchronization is complete (ie. IDLE substitution, no scrambling, etc.) and afterward
    - Logical stack should capture this
4. /S/ may appear in the first lane of any 32bit XGMII word. PCS must manage the synchronization process in a manner that assures that the /S/ will be in the data position of the 66b block (ie. first or fifth) required by coding rules
5. PCS needs ability to operate with the PMA layer regardless of its nominal clock rate

Observations on the Scrambler

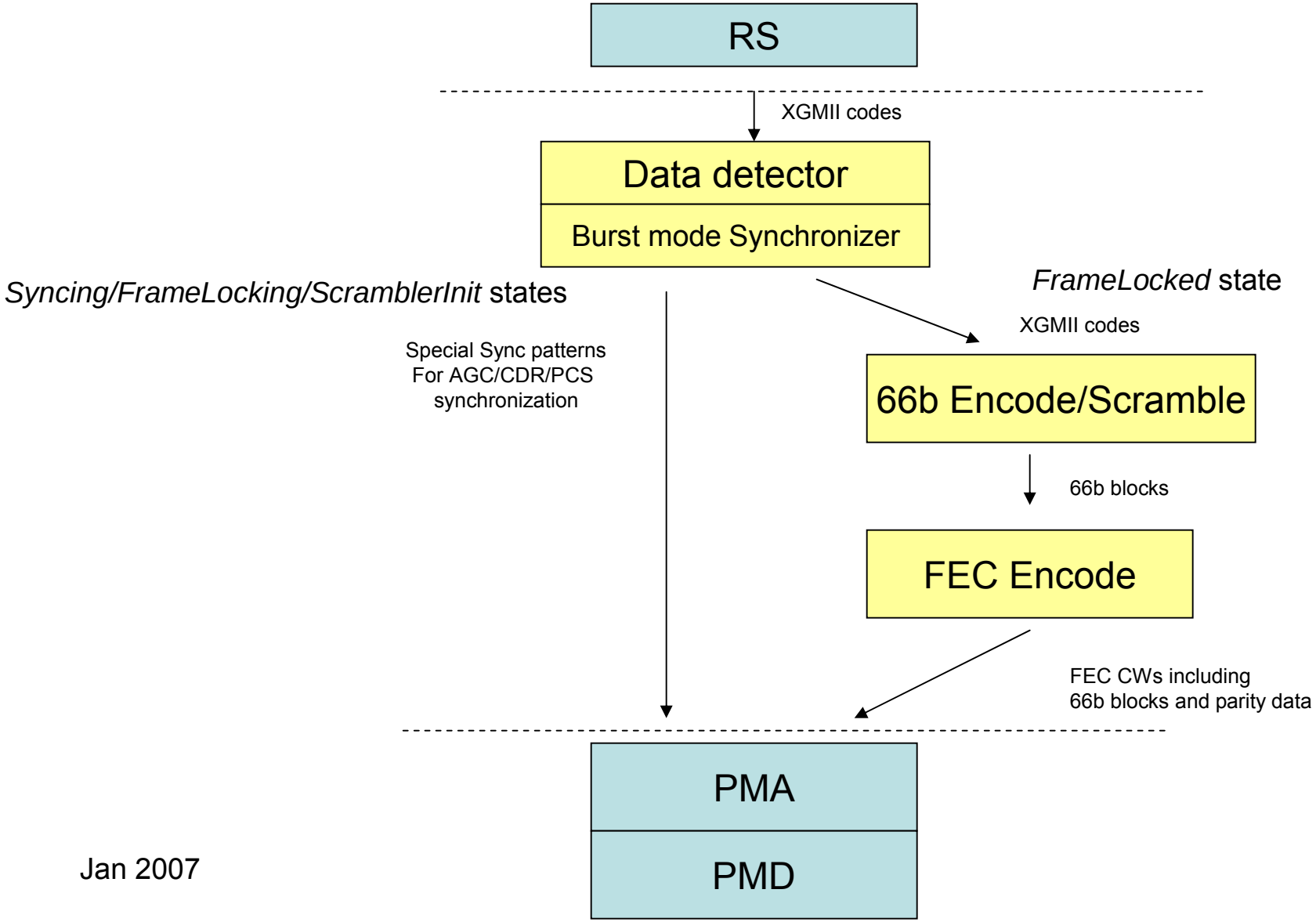
- Preventing “killer frames” seems to be regarded as a requirement on the upstream
- Scrambler must therefore reinitialize for each burst into a random state (ie. *seed*) that will be known both to the ONU and OLT

PCS/Data Detector/Scrambler for 64b/66b code (ONU)

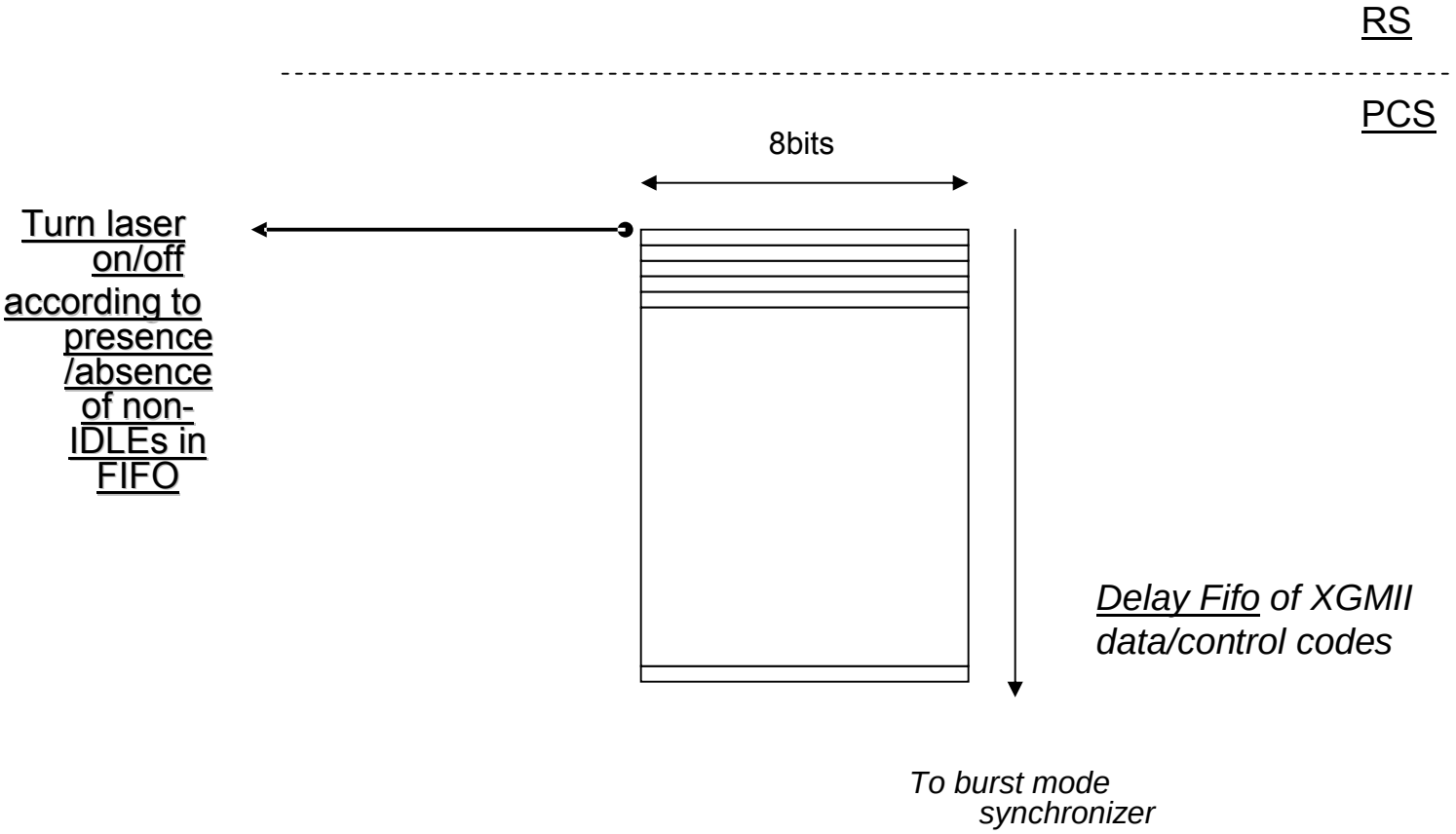
Consequently:

- Natural location in the stack for the 10GEPON data detector and syncing function is at the top of the PCS (just below the XGMII).
  - This way no decoding/recoding is necessary.
- Logical path thru the PCS is determined by *burst mode synchronizer* state
- Synchronizing function includes alignment logic
- OLT's Descrambler receives its pseudorandomly generated initial seed (*tbd* bits long) from ONU as part of the synchronization process.
  - Seed must be protected by FEC, but is not contained in a 66b block
  - After receiving seed, OLT needs time to initialize its descramber.

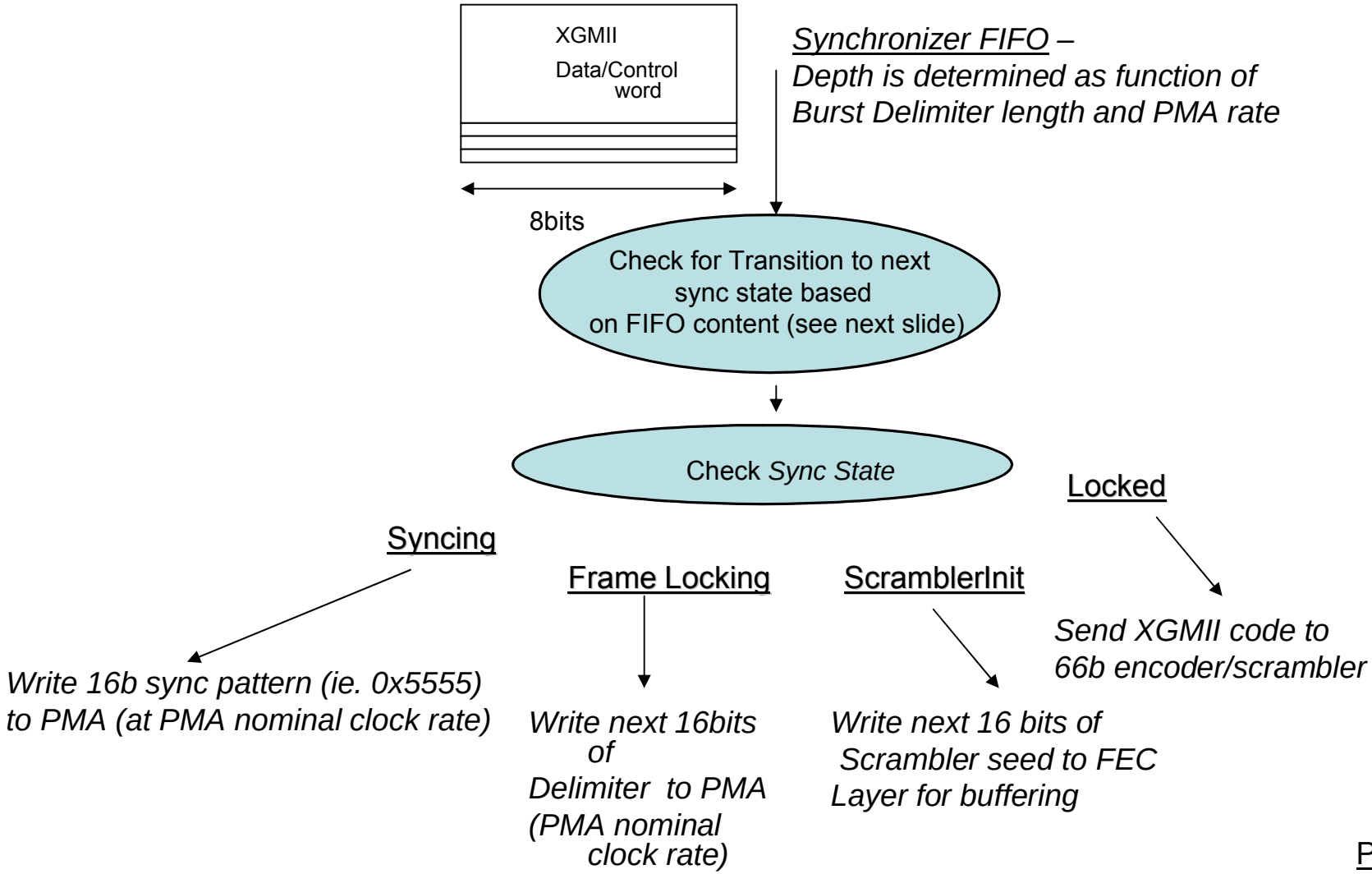
Placement of Data Detector/Synchronizer



Data detector internals



Burst Mode Synchronizer internals



Synchronizer State Transitions

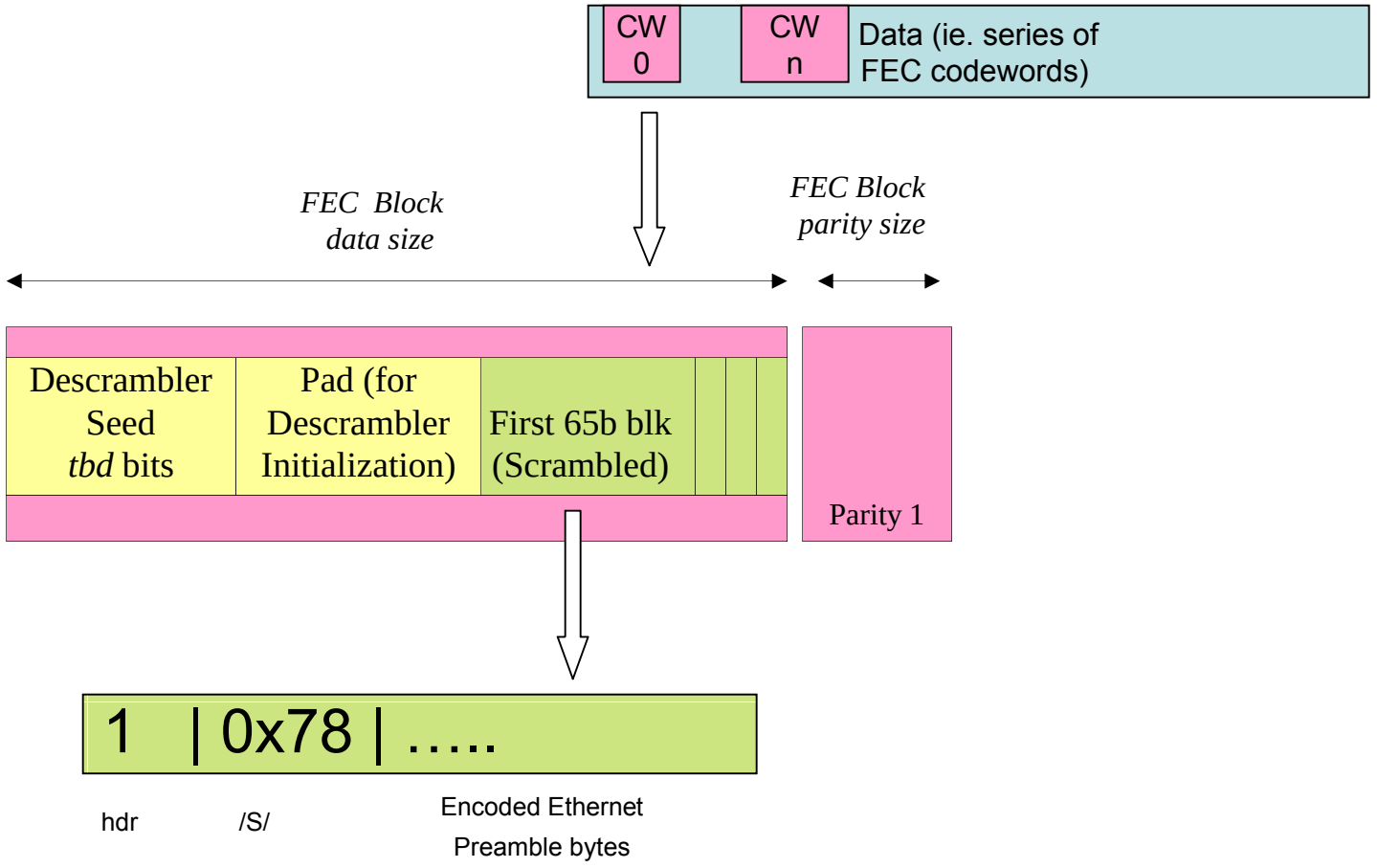
- When FIFO contains only IDLE
  => enter *Syncing*

- When “trigger” position in the FIFO (corresponding to lead time needed for delimiter transmission) contains START
  => enter *FrameLocking*

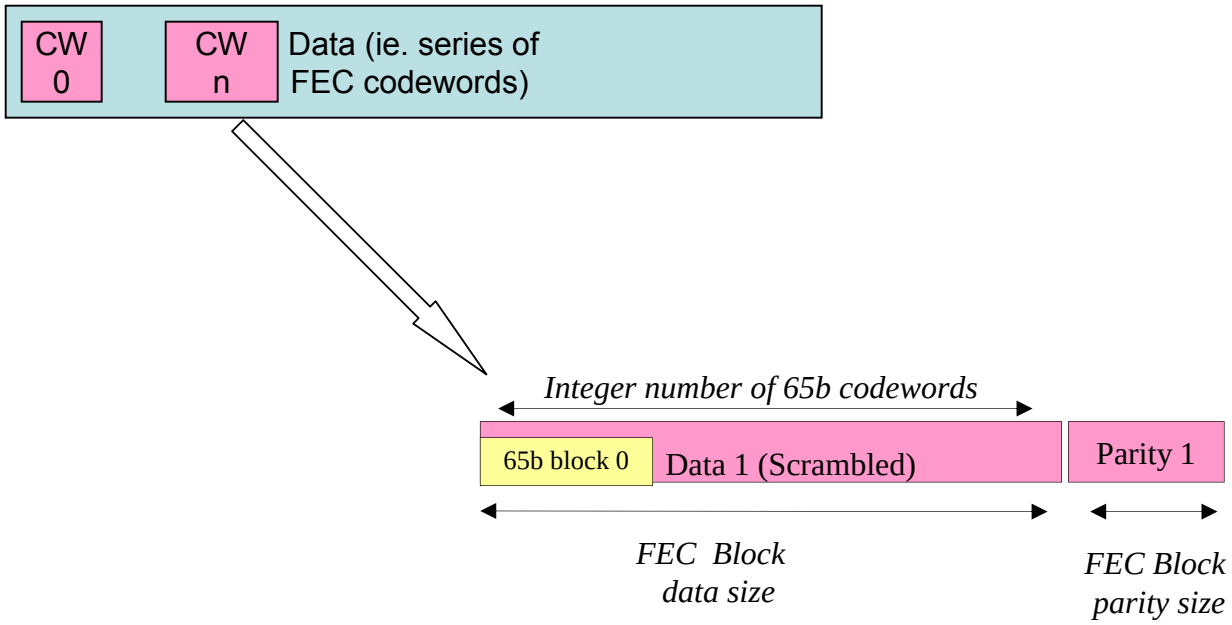
- When seventh position from the front in the FIFO contains START
  => enter *ScramblerInit*

- When front position in the FIFO contains START
  => enter *Locked*

Structure of the data in the upstream FEC Codewords (CW 0)



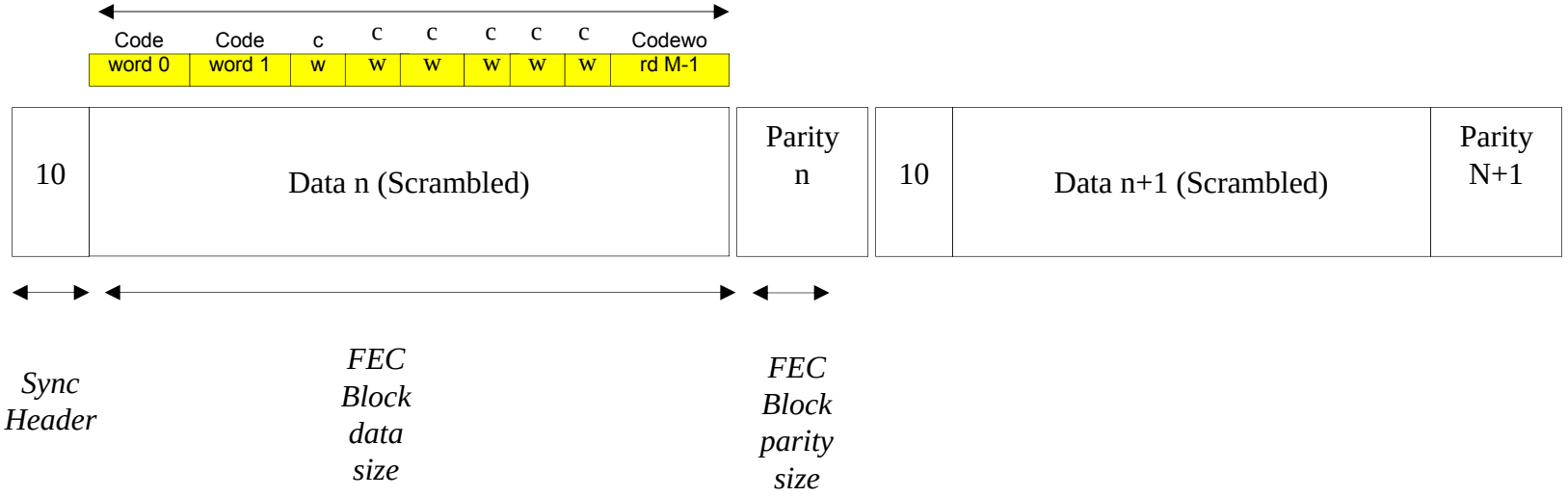
Structure of the data in the upstream FEC Codewords



Downstream

Option 1: Frame structure for DS lock via 2 bit sync header

*29 * **65b** codewords*



Option 1: Frame structure for DS lock via 2 bit sync header

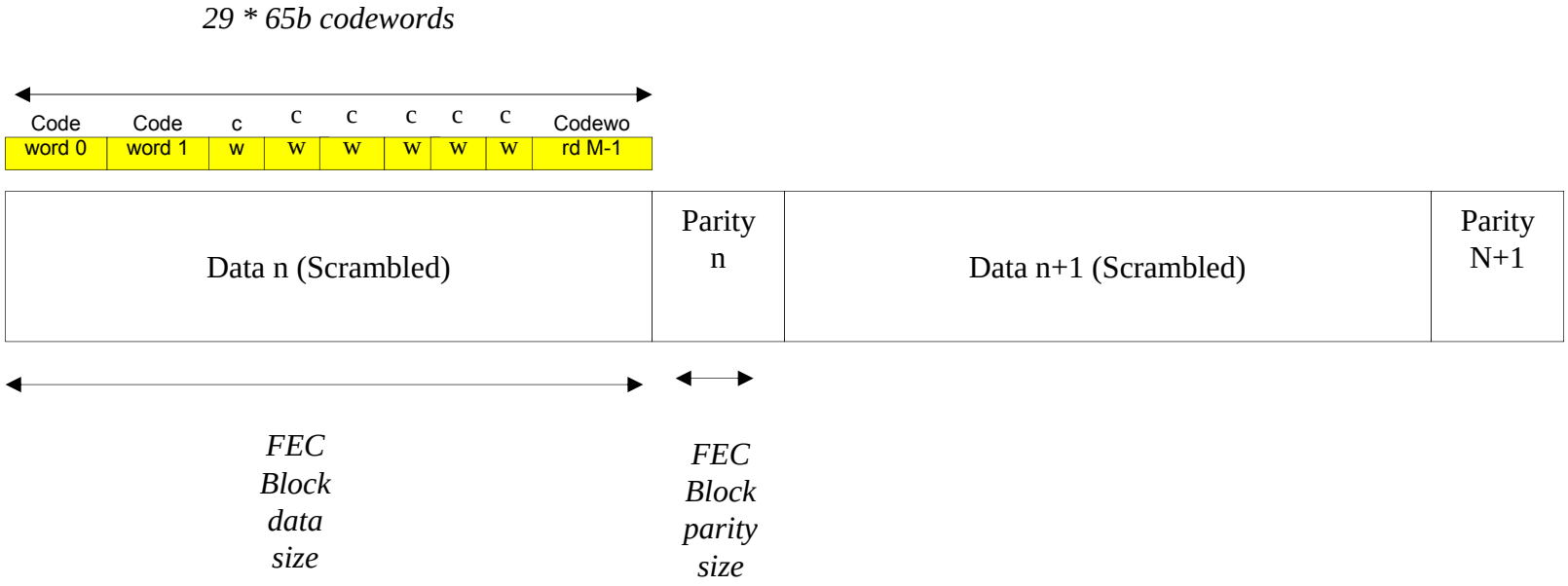
- First bit of 66b is truncated before building FEC word (as in 802.3ap) – creating a 65b block

- Data Region of FEC CW is preceded by “10” sync header

- ONU uses the standard 10GBASE-R mechanism to find the location which always has a bit transition. Lock acquisition and loss mechanisms are entirely standard

- **Benefits:** Well-known algorithm; 29 (instead of 28) codewords are contained in a frame; Sync header is DC-balanced; no risk of killer patterns

Option 2: Frame structure for Downstream Serial Lock



Option 2: Sequence for Downstream Serial Lock

- Serial locking algorithm operates thus:

  1. Select a bit position as potential beginning of a FEC block
  2. Run the FEC parity check/correction function
  3. If parity check fails ("uncorrectable block"), then slip to the next bit position

- If parity check passes for 4 consecutive blocks then we declare that FEC Codeword sync has been established
- 65b block sync has also been established implicitly
- No need for additional FEC block (in typical implementation)

- **Benefits:** Simple algorithm; 29 (instead of 28) codewords are contained in a frame; no byte overhead

Thank you