PCS and RS-FEC LPI behavior

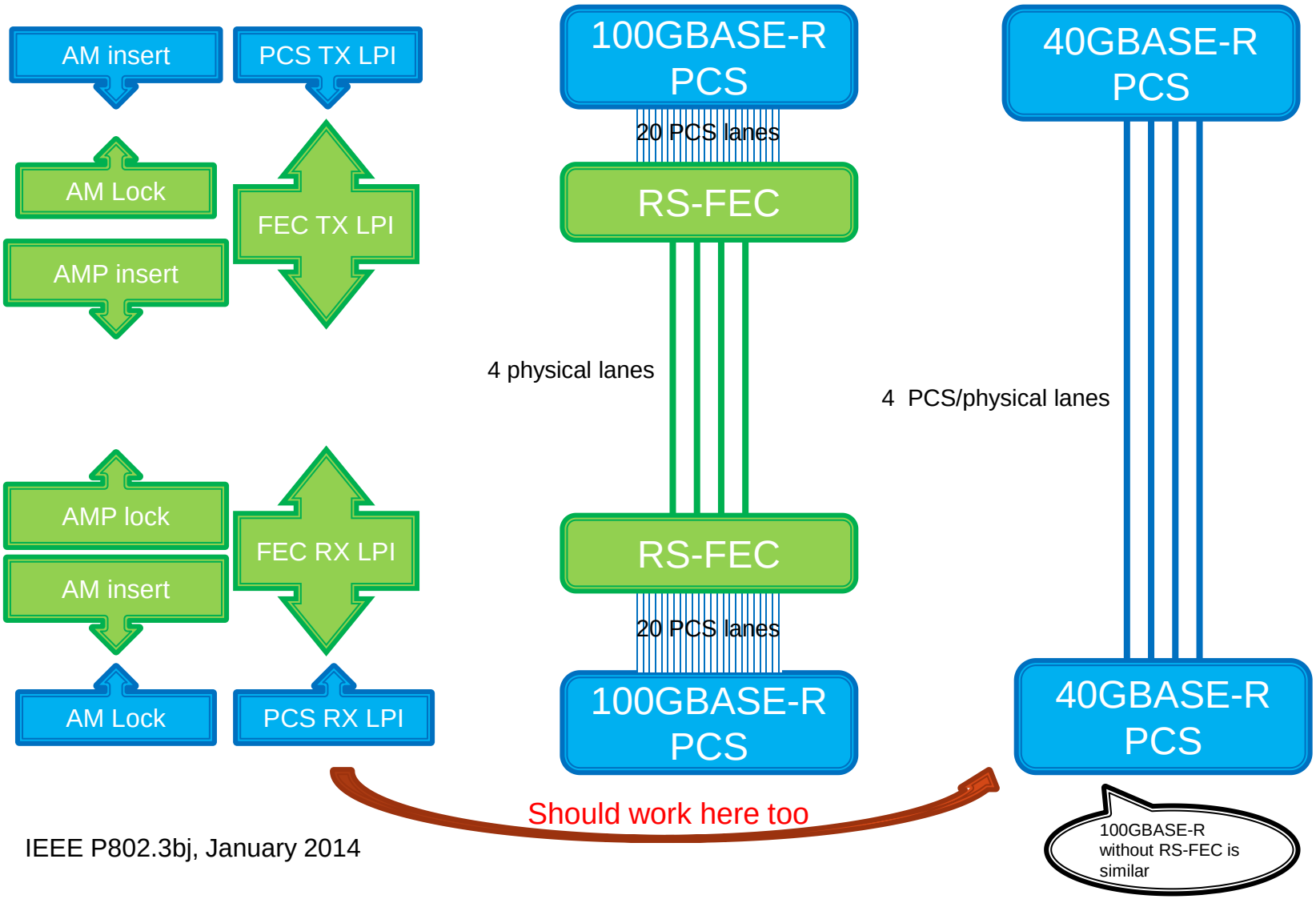
In support of comment #i-104 against D3.0

Adee Ran, Intel Corp.

Contents

- LPI functionality should work in the same way when a direct PCS-to-PCS communication is used, and when RS-FEC is in the path.
- To enable that, the following may need to be modified
  - State diagrams:
    - PCS Receive AM lock
    - PCS LPI Receive
  - Variable definitions:
    - am_counter
    - am_valid
    - first_am
    - current_am
    - ramps_valid

New LPI use cases



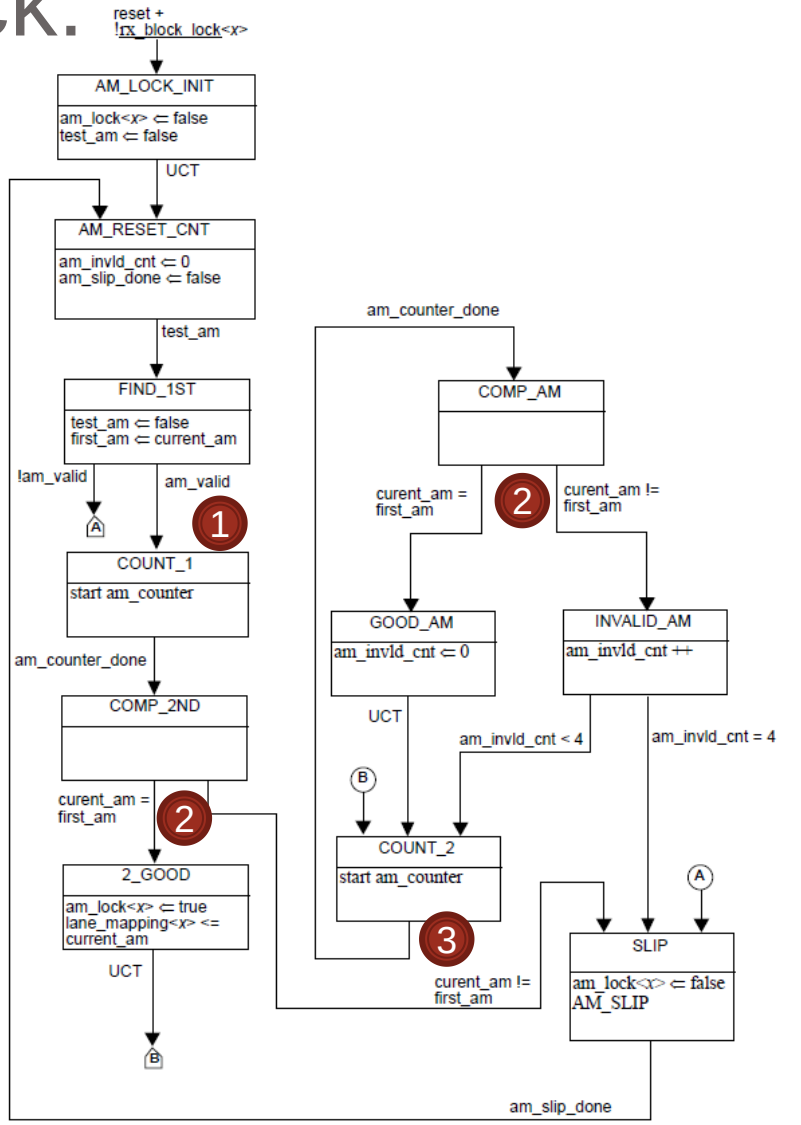
General requirements

- Exiting LPI, the RX should perform quick alignment and deskew
  - This should be possible using RAMs, and should be maintained when RAMs change to normal AMs and vice versa.
  - rx_align_status should remain TRUE during these transitions.
- QUIET should be detectable by losing AM lock (rx_align_status; similar to fec_align_status when RS-FEC is used).
  - rx_align_status should turn FALSE quickly when transmission stops (based on RAMs).

PCS receive AM lock

PCS receive AM lock: Problem statement

- Current diagram doesn't address RAM→AM transitions and vice versa:
  1. RAMs are not valid AMs
  2. When first AM occurs after RAMs (e.g. WAKE) comparisons will fail, causing SLIP and setting rx_align_status to FALSE
  3. When first RAM occurs (e.g. SLEEP) the process will usually be in COUNT_2; RAM will not cause exit from this state
     - SLIP (and setting rx_align_status to FALSE) will occur after 3-4 AM cycles.
     - If TX goes directly from SLEEP to WAKE (figure 82–16), the 36 RAMs in WAKE may never be detected. AM position will change and alignment will be lost.



NOTE— am_lock<x> refers to the received lane *x* of the service interface, where *x* = 0:3 (for 40GBASE-R) or 0:19 (for 100GBASE-R)

Figure 82–11—Alignment marker lock state diagram

Controlling am_counter terminal count

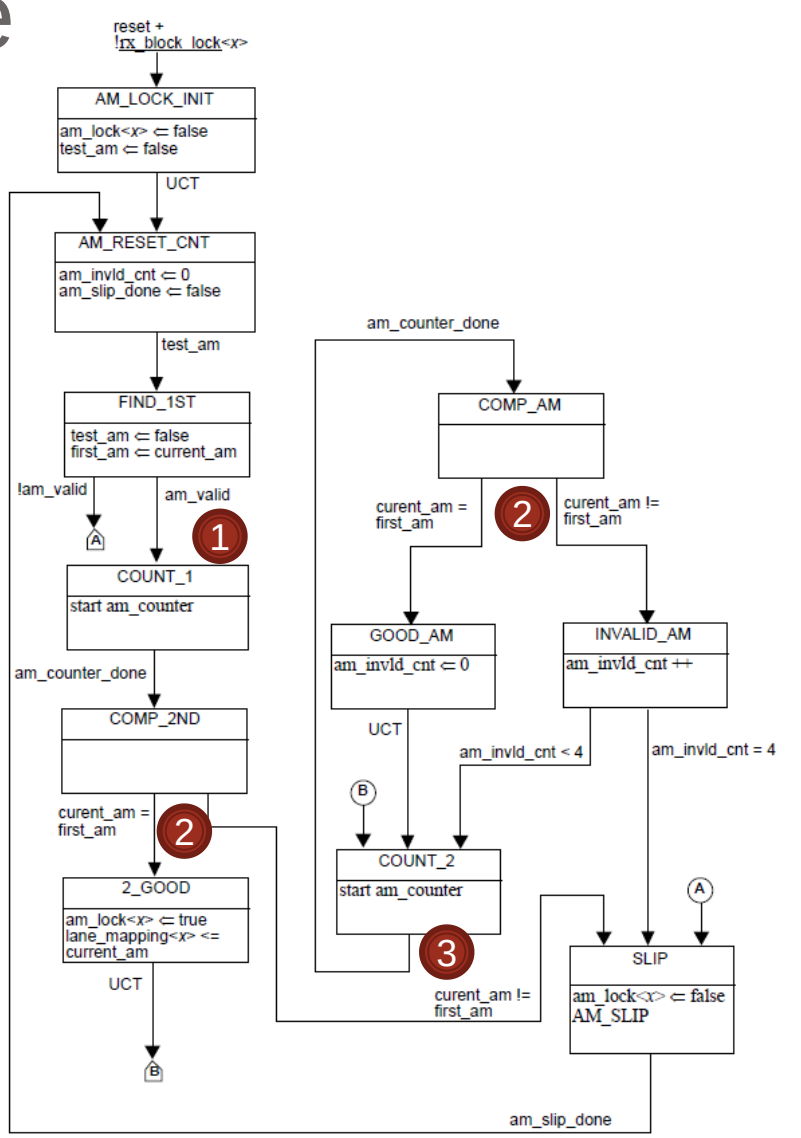
- To enable maintaining am_lock and rx_align_status during RAM↔AM transitions, switching between am_counter terminal count values should be synchronous with the change.
- The current definition of **am_counter** (82.2.18.2.4) does not clarify how this is done:

This counter counts 16383 66-bit blocks that separate two consecutive alignment markers. This counter counts 7 66-bit blocks for 100GBASE-R PCS or 15 66-bit blocks for 40GBASE-R PCS that separate two consecutive rapid alignment markers for the optional EEE capability.

- A possible solution is to use rx_lpi_active to control the counter. For that, rx_lpi_active should be *synchronized by* and *inferred from* RAMs.
- Proposed change to this definition:
  - This counter counts 66-bit blocks that separate two consecutive alignment markers. If the optional EEE deep sleep capability is supported, **when rx_lpi_active is TRUE**, the terminal count is 7 for 100GBASE-R PCS and 15 for 40GBASE-R PCS. If the optional EEE deep sleep capability is not supported, or **when rx_lpi_active is FALSE**, the terminal count is 16383.
- In addition, the counter has to be *restarted* (with a shorter terminal count) when RAMs replace AMs.

Recall: PCS receive AM lock diagram

- Required changes:
  1. RAM should pass am_valid check.
  2. AM and RAM of the same lane should be considered as matching when comparing “current_am=first_am”.
  3. First RAM should cause transition out of COUNT_2 state, to restart the counter with a shorter period on SLEEP signaling.
- RS-FEC TX refers to the same diagram, so it should also work there; alignment should be maintained regardless of whether AMs or RAMs are received.
- **Detailed solution proposed in next slide.**

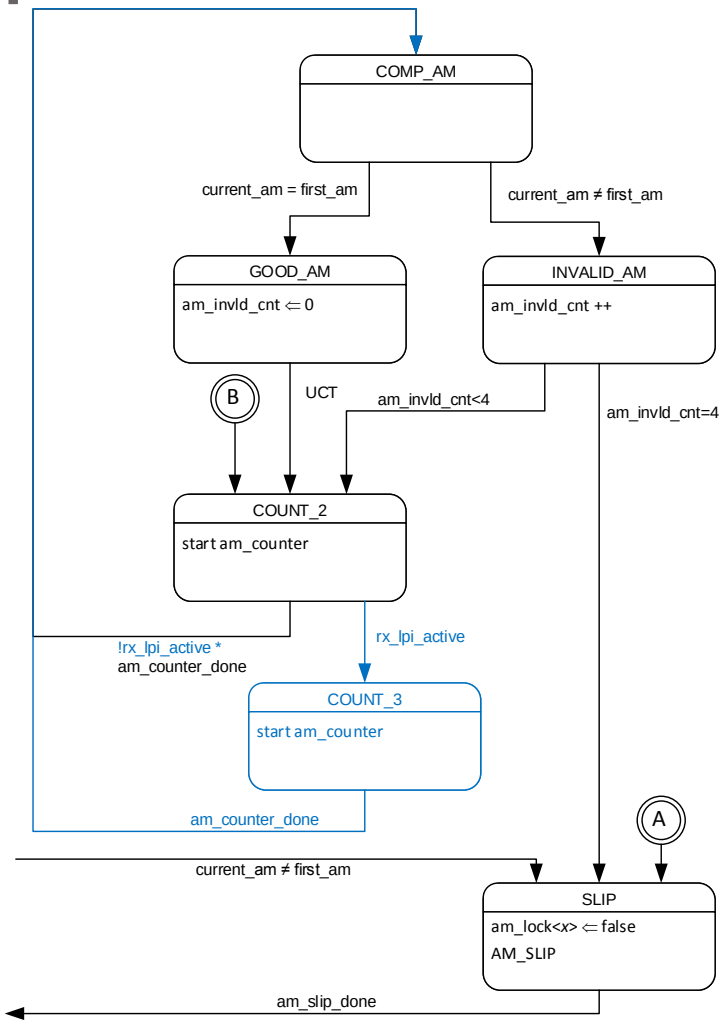


NOTE— am_lock<x> refers to the received lane *x* of the service interface, where *x* = 0:3 (for 40GBASE-R) or 0:19 (for 100GBASE-R)

Figure 82–11—Alignment marker lock state diagram

Proposed solution

- Change variable definitions (in **82.2.18.2.2)**:
  - **am_valid:** change "…in Table 82–2, ~~or~~ Table 82–3, Table 82–4a or Table 82–4b…". Delete "~~and it will be repeated every 16384 blocks~~".
  - **first_am:** A variable that holds the ~~value~~ lane number of the first alignment marker…
  - **current_am:** This variable holds the ~~value~~ lane number of the current alignment marker…
- Modify the right-hand side of figure 82-11 as shown.
  - New state COUNT_3 causes am_counter to restart in sync with transition of rx_lpi_active.



PCS LPI Receive

PCS LPI Receive Problem statement

1. The transition RX_SLEEP→RX_QUIET should occur only after transmitter stops sending RAMs (no energy); otherwise a premature transition to RX_WAKE would occur.
   - ➔ rx_align_status must be held TRUE while RAMs are received.
2. "rx_lpi_active should be *synchronized by* and *inferred from* RAMs"
   - rx_lpi_active is changed during the RX_WAKE→RX_ACTIVE transition. Currently this happens when R_TYPE is C (IDLE) – before RAMs change to Ams – so is not synchronized.
   - rx_down_count can be used for this purpose instead.
3. The RX_SLEEP→RX_ACTIVE transition is also unsynchronized with AMs.
   - Note: no equivalent scenario in PCS TX diagram (always goes through TX_WAKE).

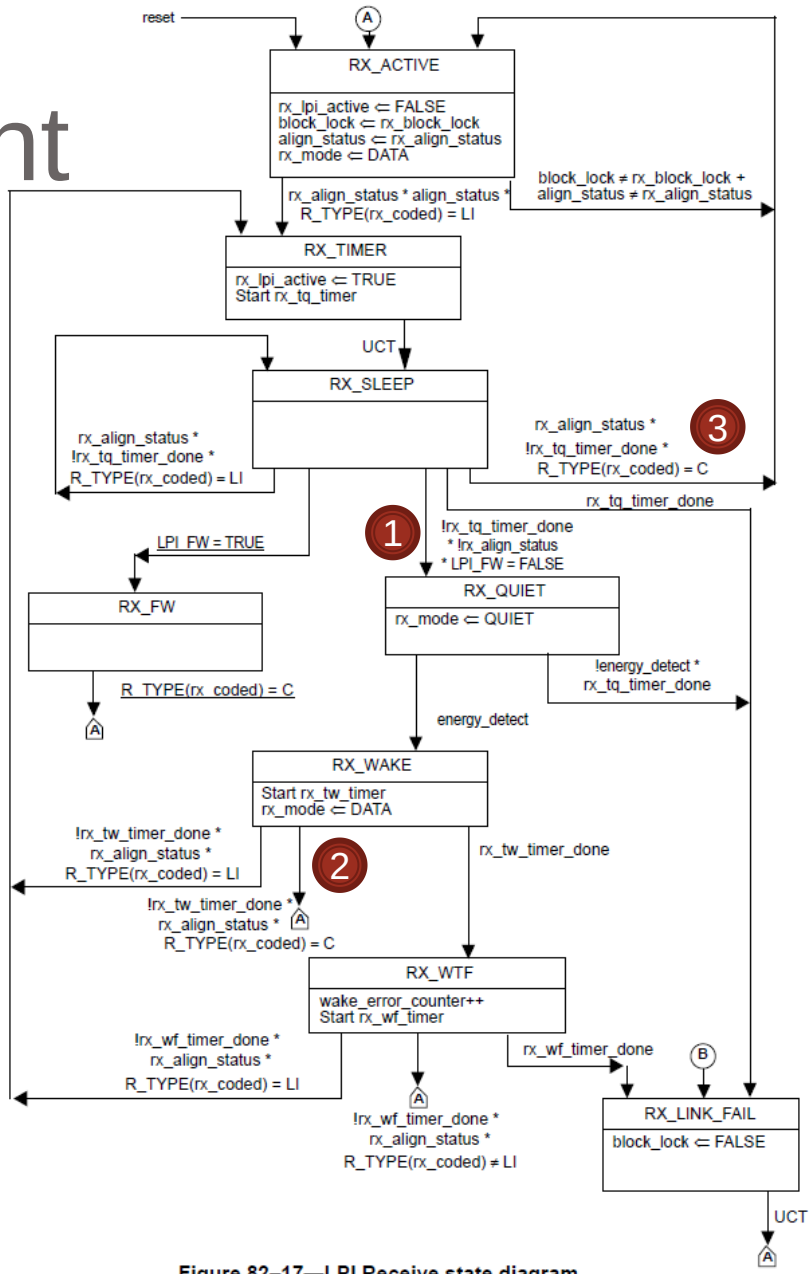


Figure 82–17—LPI Receive state diagram

Information in rx_down_count

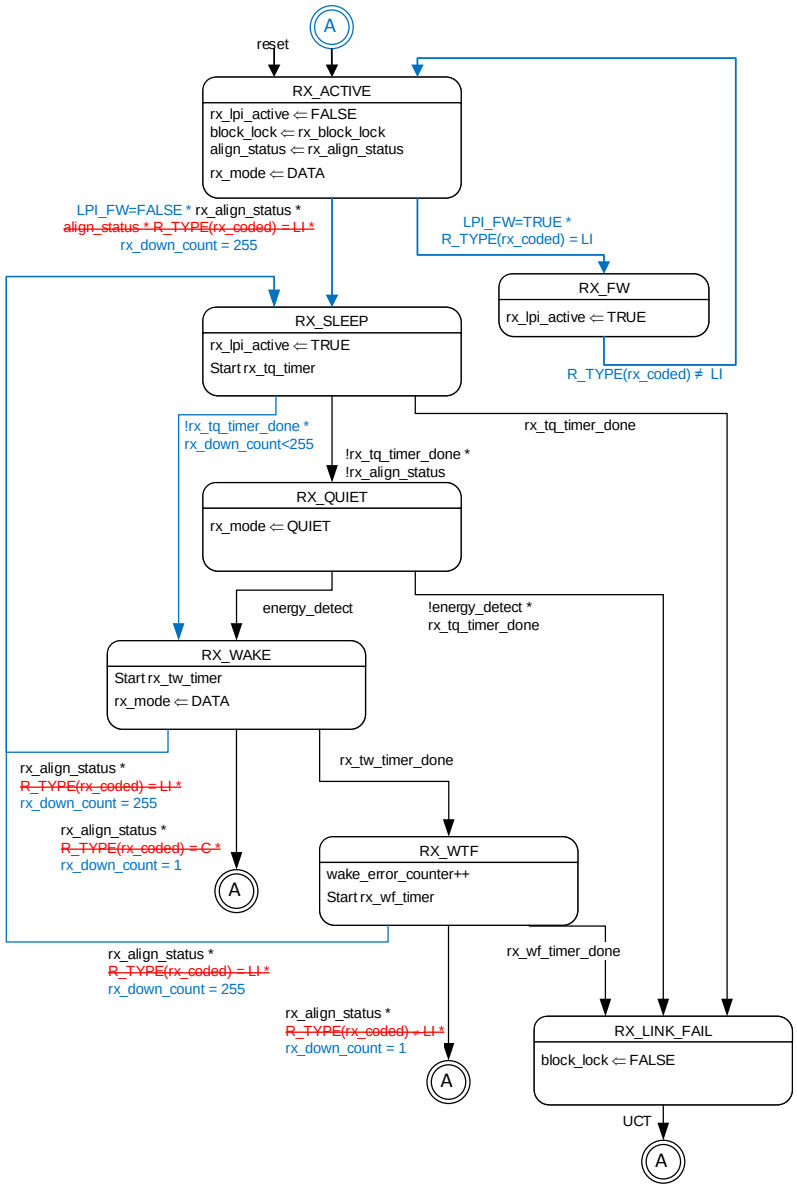
(According to figure 82-16)

Number encoded into down_count in RAM by LPI Transmit state diagram	tx_mode	Information
255	SLEEP	This down_count value *should be detected* by the RX PCS
242	QUIET	This down_count value can't be seen by the RX PCS
213	ALERT	This down_count value can't be seen by the RX PCS
54 or lower	DATA	counting down in successive RAMs; 1 designates the last RAM; enables deskewing by the RX PCS

PCS LPI receive: Proposed remedy

- Use rx_down_count instead of R_TYPE for transitions into SLEEP, out of WAKE, and between the two
  - It is obtained from RAMs synchronously.
  - After deskewing, all PCS lanes must have the same value.
- If LPI_FW is used, there are no RAMs, and rx_lpi_active is still inferred from R_TYPE.

(Some loops omitted from the diagram for clarity)



RS-FEC Receive LPI

Variables used in RS-FEC Receive LPI state diagram (figure 91-11)

- **rx_down_count**
  - Current definition (in 91.5.4.2.1): "The value that results from the bit-wise exclusive-OR of the Count Down (CD3) byte and the M0 byte of the current Rapid Alignment Marker payload (see 82.2.8a)"
  - Alignment marker payloads are per-lane, and specific to RS-FEC (defined in 91.5.2.6, not 82.2.8a).
  - Proposed change:
    - **rx_down_count_i<x>**, x=0 to 19: the value that results from the bit-wise exclusive-OR of the Count Down (CD3) byte and the M0 byte of the current Rapid Alignment Marker payload **(see 91.5.2.6) of PCS lane x** in the receive direction.
    - **rx_down_count:** the mode (value with highest count) of all rx_down_count_i<x>.
- **ramps_valid**
  - Current definition (in 91.5.4.2.1): Boolean variable that is set to true when the 66-bit blocks concurrently received on at least 2 FEC lanes are valid Rapid Alignment Marker payloads with identical values in the Count Down fields and is set to false otherwise.
  - A payload sequence is a 1285-bit block (before re-mapping) which does not comprise valid 66-bit blocks;
  - Count Down fields are down_count XOR'ed with M0 so are not identical between PCS/FEC lanes.
  - Before FEC lanes are deskewed, different lanes may have different down_count; this should not be a condition for the payload sequence to be valid.
  - Proposed change:
    - **ramps_valid_i<x>, x=0 to 3**: Boolean variable that is set to true when **a valid RAM payload sequence is received on FEC lane x**, with at least two identical values of **rx_down_count_i calculated from the payloads**, and is set to false otherwise.
    - **ramps_valid**: Boolean variable that the is the result of logical AND of ramps_valid_i<x>, x=0 to 3.

Thank you