Canova Tech

The Art of Silicon Sculpting

PIERGIORGIO BERUTO

IEEE 802.3cg

Clause 148 cleanup

September 9th, 2019

Supporters

- Tim Baggett - Microchip
- Alan Tang (志文) - Realtek

Outline

- During the initial SA ballot phase there have been a number of comments asking for clarifications and fixing state diagram semantic. These were mainly related to:
  - Timers and tolerances
  - Clocks & timings definition
  - Execution order of state diagram statements
  - Operators precedence
- None of these comments was intended to change the functionality being described
  - But they introduced a few editorial and technical regressions
- Several individuals with different affiliations worked in background to stabilize the draft
- This presentation shows the final changes that have been checked and validated by all parties

Validation

- The validation of the PLCA state diagrams has been performed by the means of Verilog testbenches including:
  - models of the Clause 4 MAC and 10BASE-T1S PHY
  - ideal model of the mixing-segment (including collisions)
  - model of the PLCA RS
- All models have been implemented exactly as they are specified in 802.3 and 802.3cg D3.3.
  - timer tolerances and best/worst case specified latencies have been modeled as well
  - these have been randomized to cover all possible combinations
- The testbenches validated the PLCA RS under several conditions, including:
  - normal TX/RX of packets when all MACs are sending back-to-back (max network load)
  - mix of PLCA enabled nodes and non-PLCA enabled nodes
  - on-the-fly enable/disable of PLCA while nodes are sending/receiving data
  - burst mode in different configurations
  - reception of corrupted/invalid data (PLCA recovery mechanism)
  - different settings of the configuration parameters (e.g. to_timer, number of nodes, . . .)
- Validations have also been performed on FPGA-based boards emulating real networks

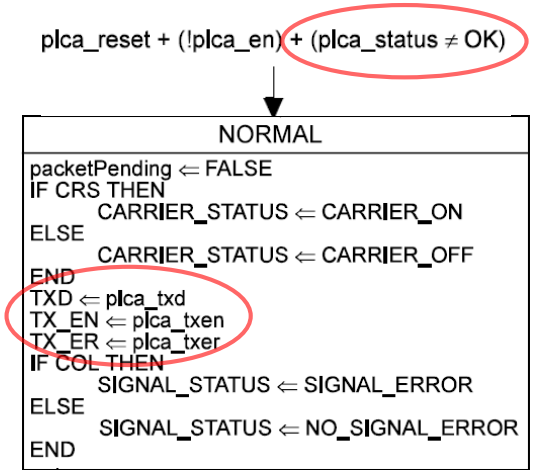
Results

- In the process of validating the state diagrams in D3.3 a few regressions have been found
  - clerical errors causing obviously broken behaviors
  - corner cases leading to performance degradation
  - potential violation of MII setup/hold requirements
  - weaknesses caused by combination of the best / worst case PHY latency allowance
- The proposed fixes have been validated and tested by the means of the same simulation environment and the same HW platforms

ISSUES & FIXES

#1: BEACON is not sent

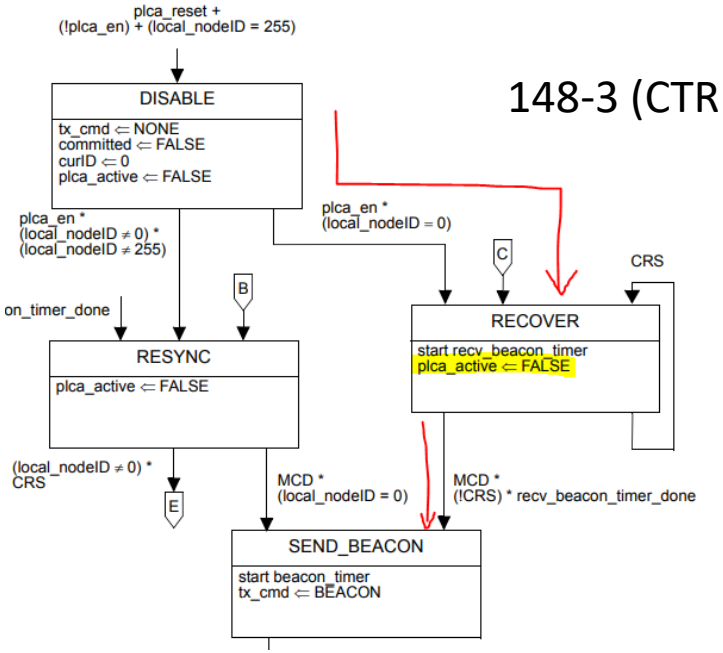
- When PLCA is disabled, plca_status is not OK
  - The PLCA Data State Diagram is held in NORMAL state
  - in NORMAL state the MII signals are driven by the MAC directly (normal CSMA/CD w/o PLCA). They cannot convey BEACON or COMMIT requests to the PHY.



148-4 (DATA)

NOTE:
plca_status is OK when plca_active is TRUE. When plca_active becomes FALSE, plca_status is set to FAIL after ~ 1s

- When PLCA is enabled for the first time, the node with ID = 0 sends a BEACON, as described in the PLCA Control State Diagram
  - since plca_active is FALSE, the DATA State Diagram remains in NORMAL state and does not send the BEACON

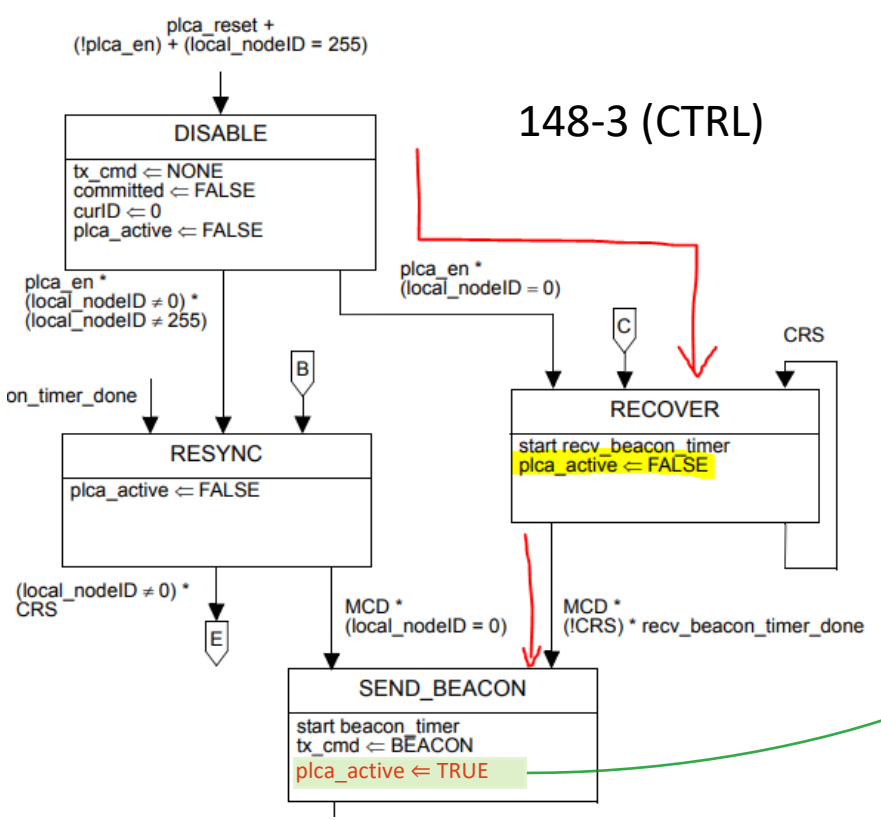


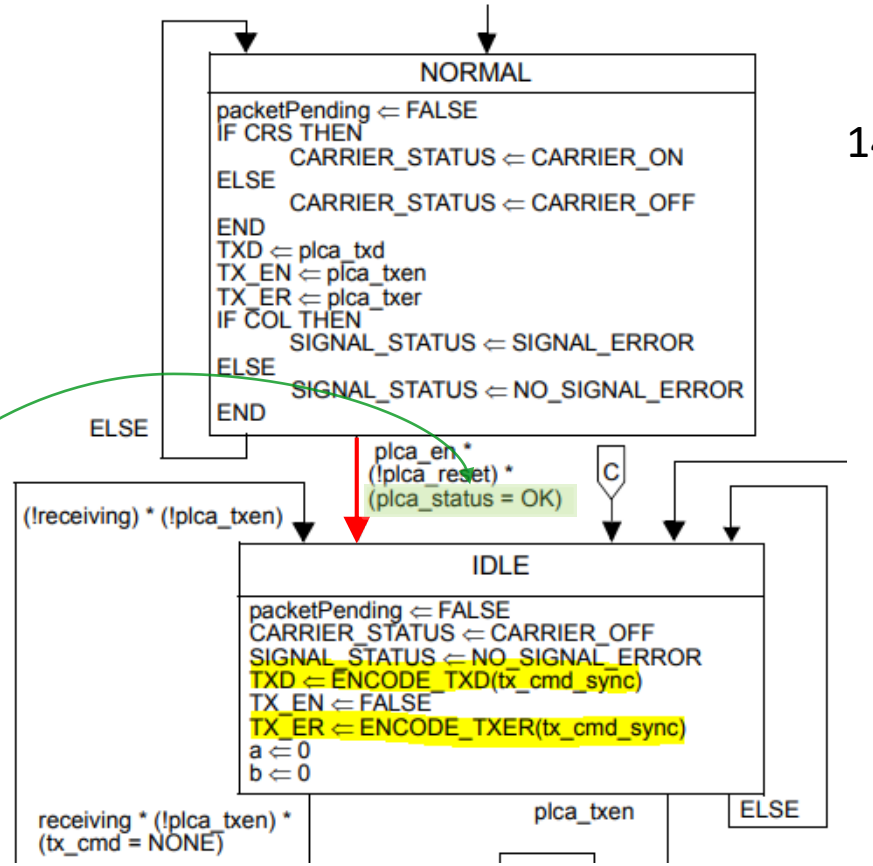
148-3 (CTRL)

#1: BEACON is not sent (fix)

The simple fix is to set plca_active TRUE in SEND_BEACON state

This makes the DATA State Diagram progress to IDLE state, where the BEACON is conveyed as intended



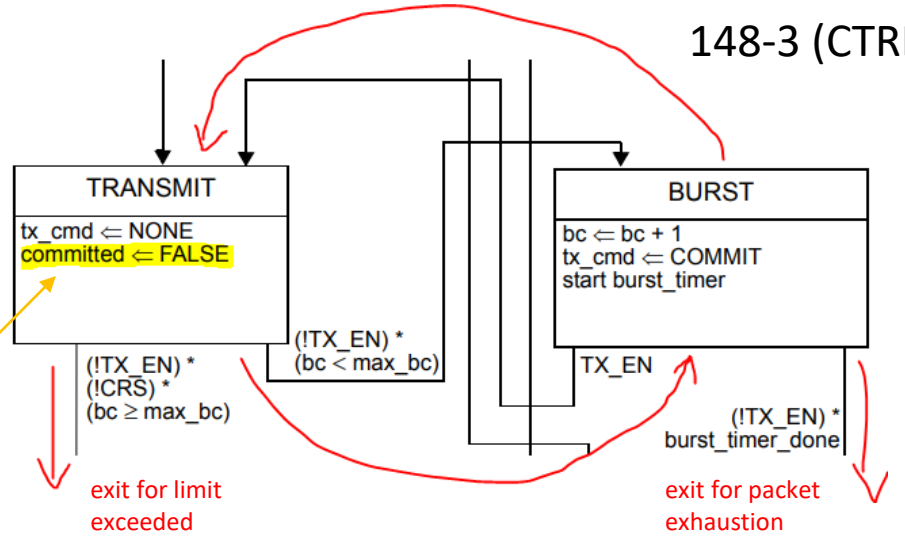


NOTE: tx_cmd_sync is the tx_cmd variable synchronized with the MII_TXCLK

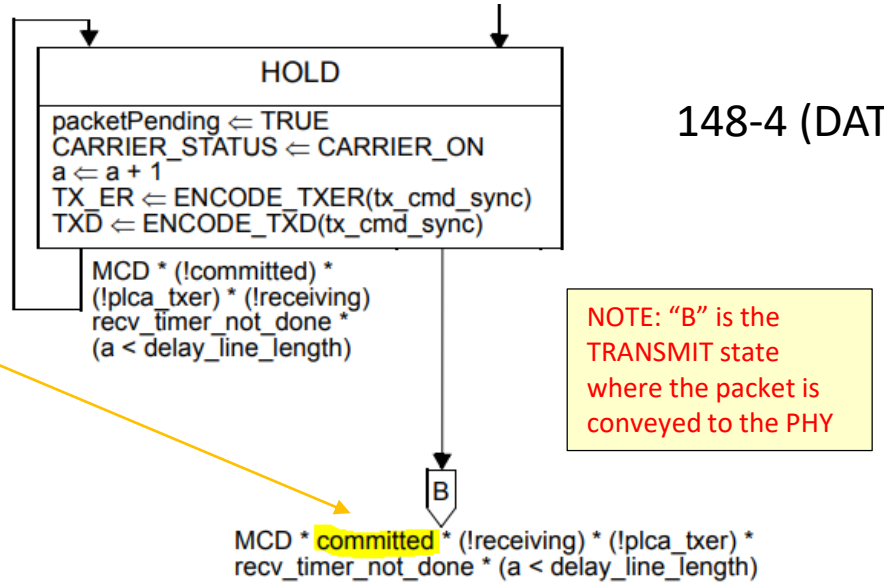
#2: packets not sent in burst mode

- When burst mode is enabled (max_bc > 0) the PLCA Control State Diagram loops between TRANSMIT and BURST states until the MAC has no more packets available to send, or the configured burst limit is reached.
  - Note that the "committed" variable is set to FALSE

148-3 (CTRL)



148-4 (DATA)

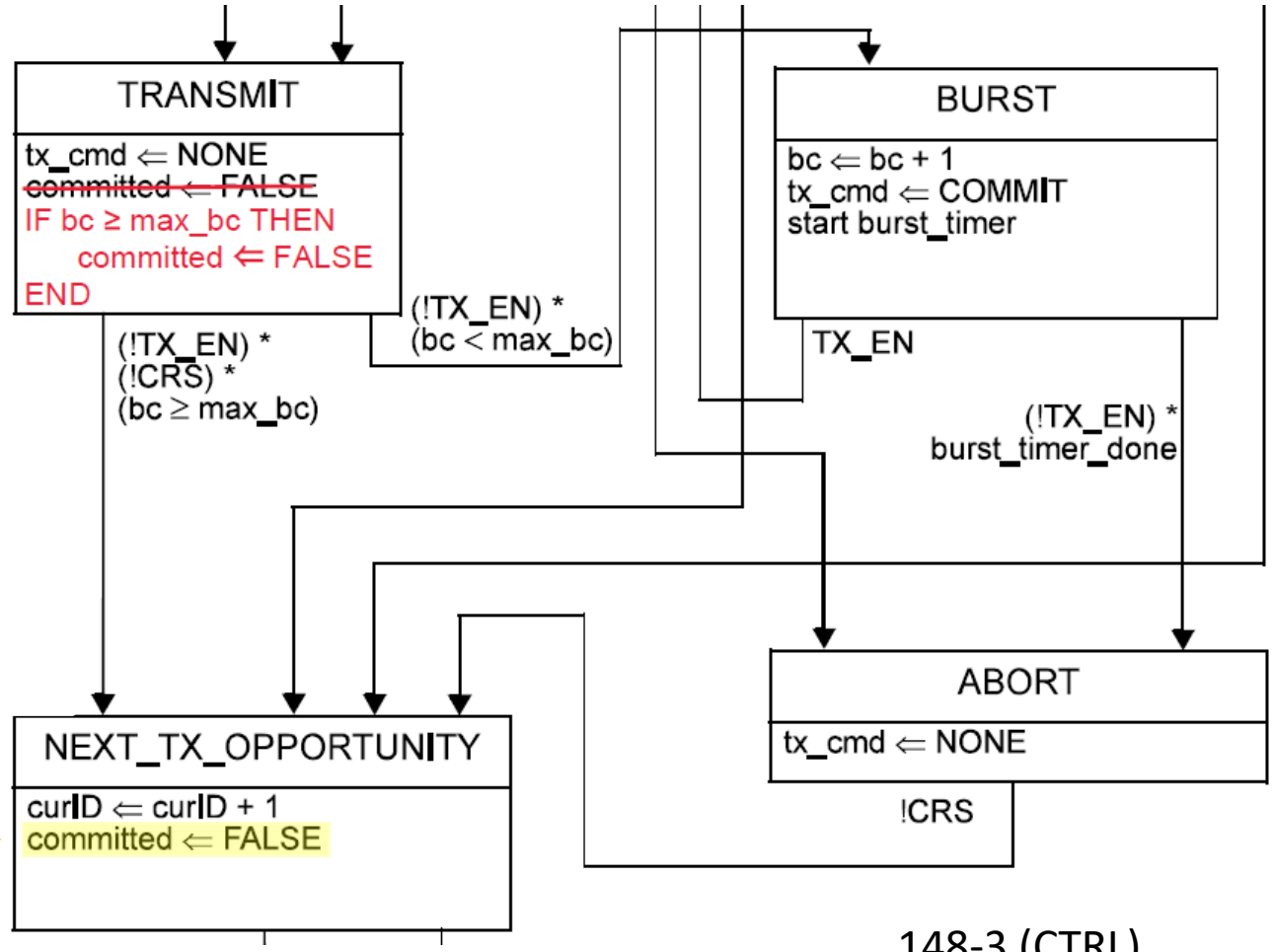


- The DATA State diagram, however, requires the "committed" variable set to TRUE to progress to the TRANSMIT state.

#2: packets not sent in burst mode (fix)

The simple fix is to prevent the "committed" variable to be reset while a burst is in progress.

In case the burst ends because the MAC has no packets available, the "committed" variable is also reset (this is already part of the draft)

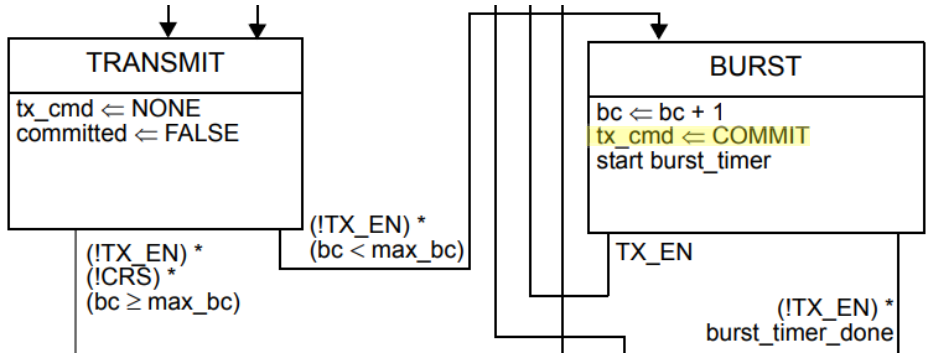


148-3 (CTRL)

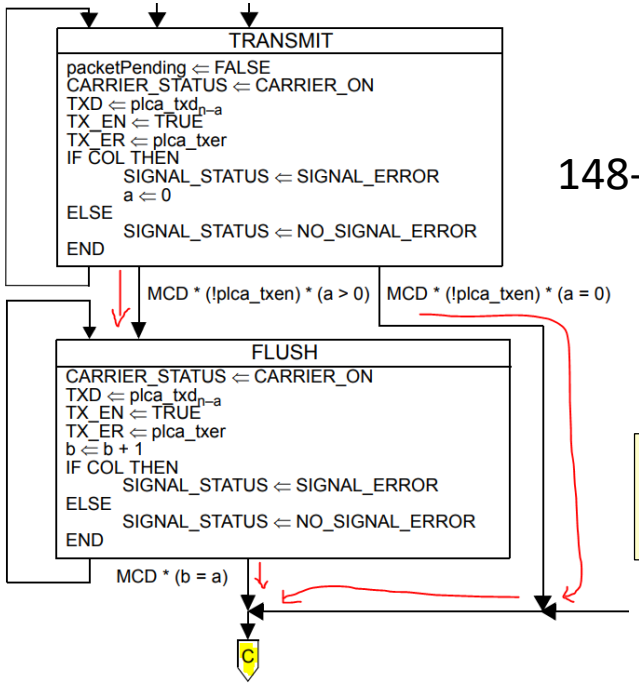
#3: fake collisions during burst (1)

- During a burst, the IPG is filled by COMMIT (IDLE symbols) to keep the transmit opportunity. See http://www.ieee802.org/3/cg/public/Nov2018/beruto_3cg_PLCA_burst_mode_revB%20.pdf.
- This is done by setting tx_cmd = COMMIT while the Control Sdtate Diagram is in the BURST state.
- After the first packet is sent, the DATA State diagram moves from TRANSMIT/FLUSH to IDLE state. This happens as soon as the delay line is empty and the MAC is done sending data.

148-3 (CTRL)



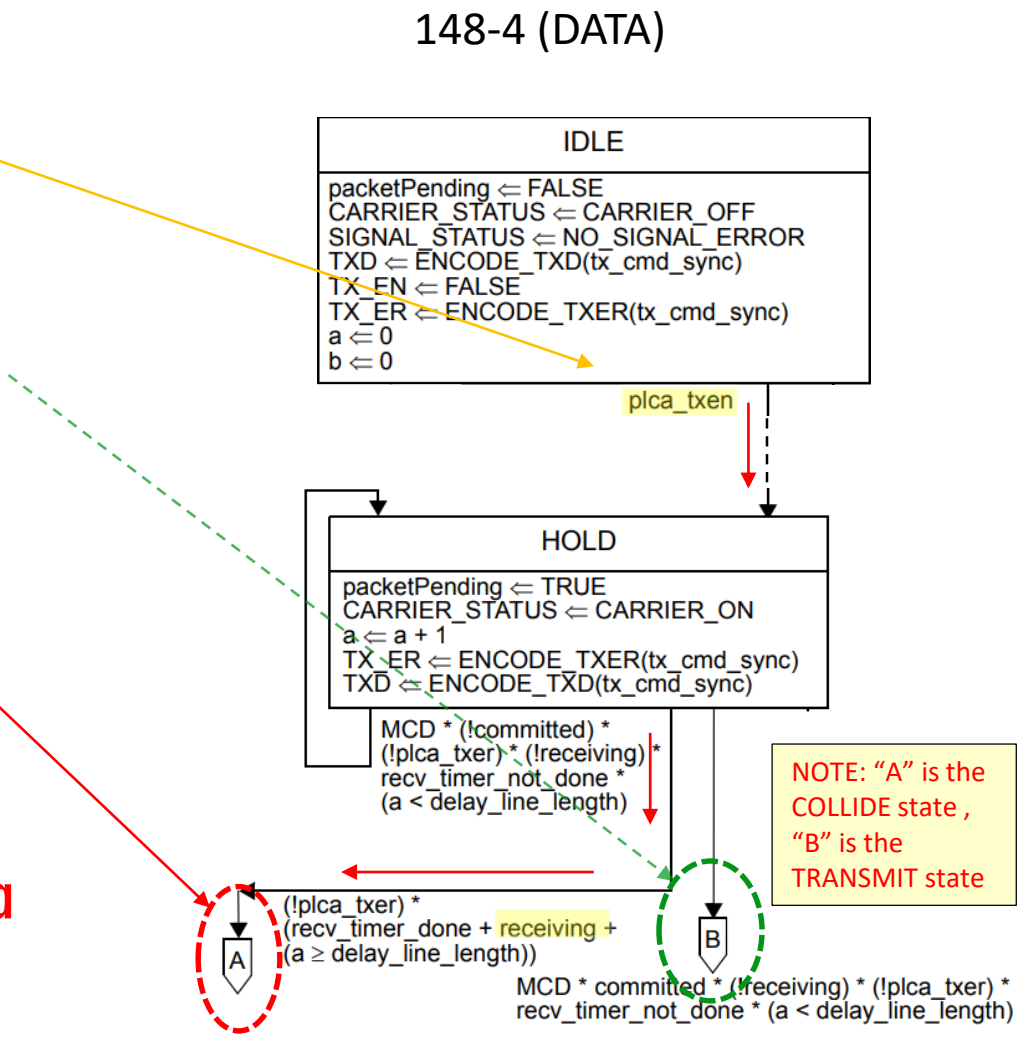
148-4 (DATA)



NOTE: "C" is the IDLE state

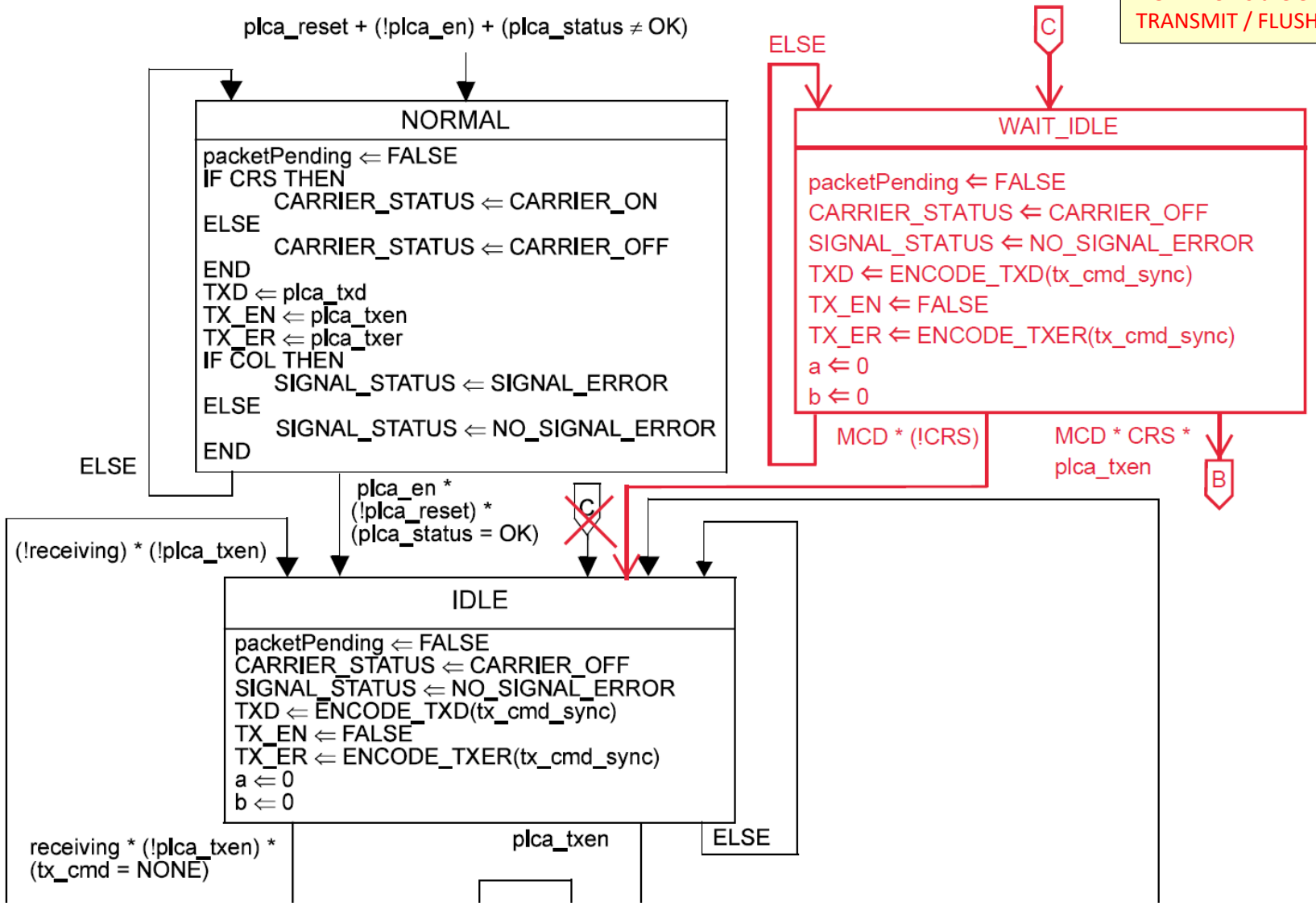
#3: fake collisions during burst (2)

- The DATA state diagram from IDLE will move to HOLD as soon as the MAC transmits the next packet.
- From HOLD state the DATA State Diagram is supposed to progress to the TRANSMIT state to have the PHY sending the packet on the line.
- However, depending on the PHY RX latency, the "receiving" variable *may* be TRUE at that time because of the COMMIT request being looped back into a COMMIT indication
  - This would make the DATA State Diagram report a false collision to the MAC.



#3: fake collisions during burst (fix)

148.4.6.7 State Diagrams



NOTE: "C" is the exit condition from TRANSMIT / FLUSH

The fix may look complicated but it is simple: have the DATA State Diagram wait for the line to be free (CRS = FALSE) before going to IDLE. Go back to TRANSMIT (B) if bursting.

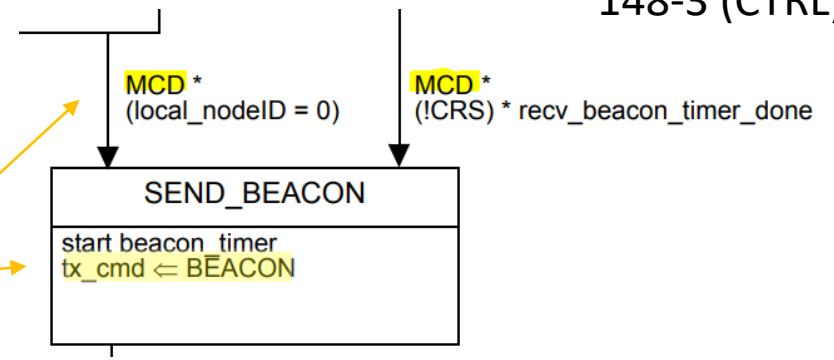
This is easily done adding a state (WAIT_IDLE) which does the same as IDLE but also detects a burst and goes to TRANSMIT as appropriate.

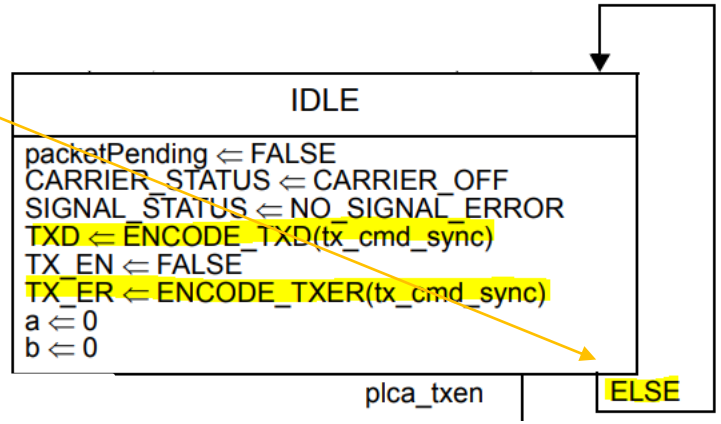
Note that during a burst the CRS is not de-asserted (it's the working principle of the burst mode), therefore the transition from WAIT_IDLE to B can only happen during such event. When NOT bursting, the transition from WAIT_IDLE to IDLE will always occur earlier than one IPG, ensuring proper behavior in normal conditions.

#4 MII setup/hold times violation

- According to Clause 22.3.1 the MII signals going from the RS to the PHY (TXD, TX_EN, TX_ER) shall be output from 0 to 25ns after the rising edge of TX_CLK.
- The SEND_BEACON state in the Control State Diagram is entered on the rising edge of TX_CLK (MCD), and the tx_cmd variable is set accordingly.
- The DATA State Diagram, which drives the MII signals, refreshes itself when the tx_cmd_**sync** variable changes.
- The tx_cmd_sync variable is defined as the sampling of tx_cmd on the **falling** edge of the MII TX_CLK.
- Since the TX_CLK clock period for 10 Mb/s is 400ns, this refresh may occur 200 ns later than the rising edge, which violates the requirement in 22.3.1 (see first bullet).

148-3 (CTRL)

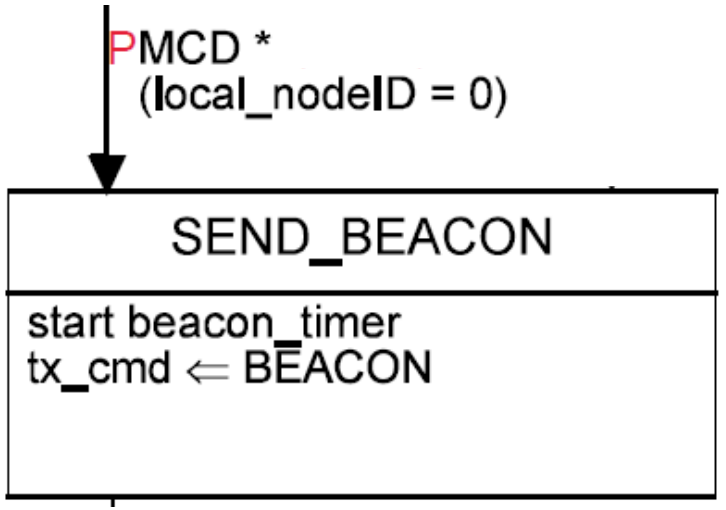




148-4 (DATA)

#4 MII setup/hold times violation (fix)

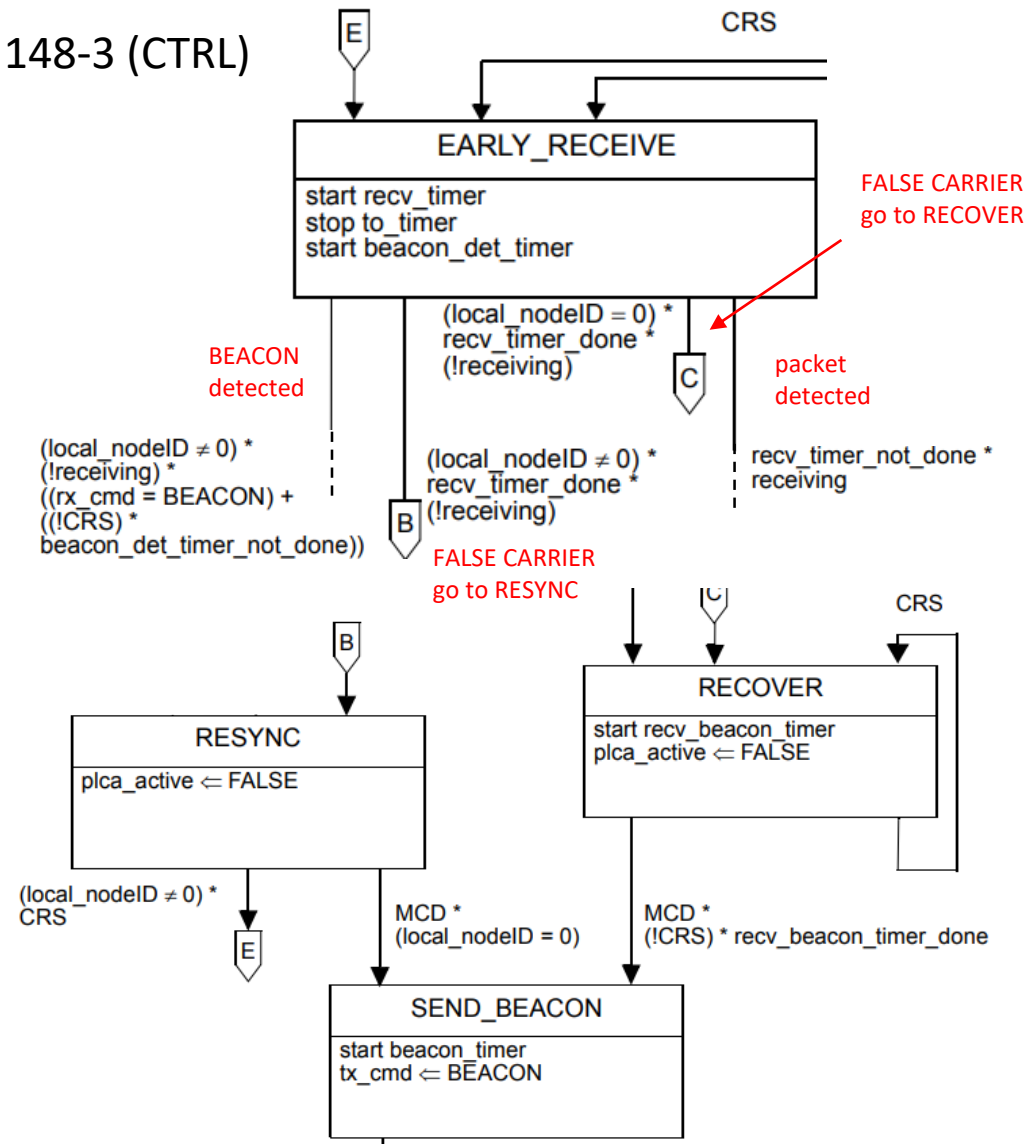
- The fix is twofold
  - Define tx_cmd_sync to be updated on the **rising** edge of the MII TX_CLK
    - This ensures the MII timings are met
  - Add a "prescient" MCD (PMCD) which occurs slightly earlier than MCD
    - This compensates for the extra clock cycle between the CONTROL and DATA State Diagrams, keeping the latencies within the required range



PMCD
Prescient mii_clock_done function.
This function becomes done
1 ± ½ bit times earlier than mii_clock_done

#5 non-PLCA nodes preventing PLCA from starting (1)

- PLCA features a "recovery" mechanism that makes a node skip its TO in case the internal TO counter is possibly misaligned.
  - This may happen when a false carrier event (e.g. impulsive noise) occurs on the line. In such situation each PHY on the network *may* assert CRS (which is involved in the counting of the TOs) for a different duration.
  - This is captured by the PLCA CONTROL State Daigram by the means of the EARLY_RECEIVE, RESYNC and RECOVER states
- The idea is that when a CRS event is not followed by the reception of a packet within recv_timer duration (i.e. the PHY could not decode the data), the node goes either in RECOVER or RESYNC state depending on whether it is the one in charge for sending the BEACON (local_nodeID = 0) or not.
  - In RESYNC state a node waits for a new BEACON before getting a new TO
  - In RECOVER state (local_nodeID = 0) the node waits for the line to be quiet (CRS = FALSE) for a certain amount of time before sending a new BEACON

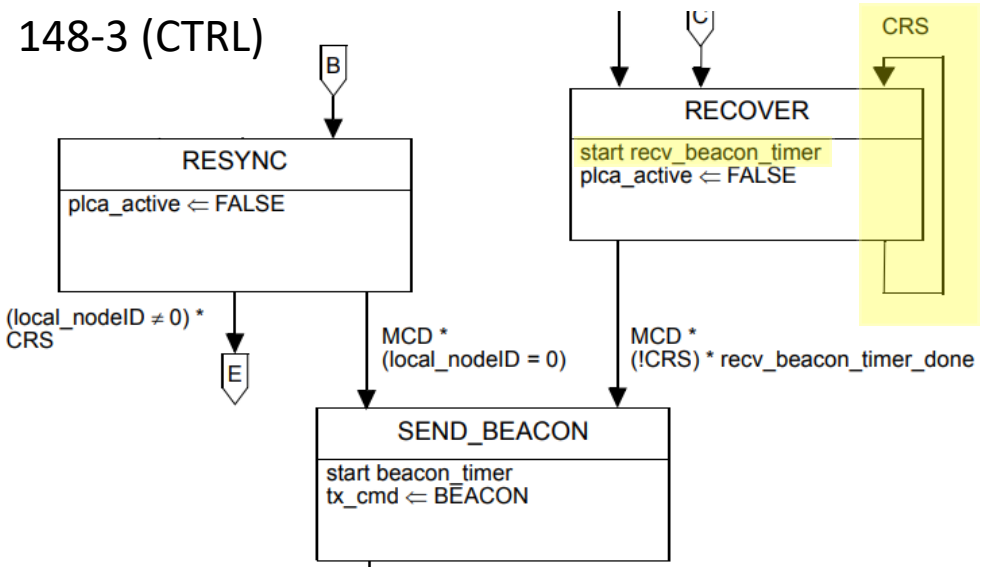


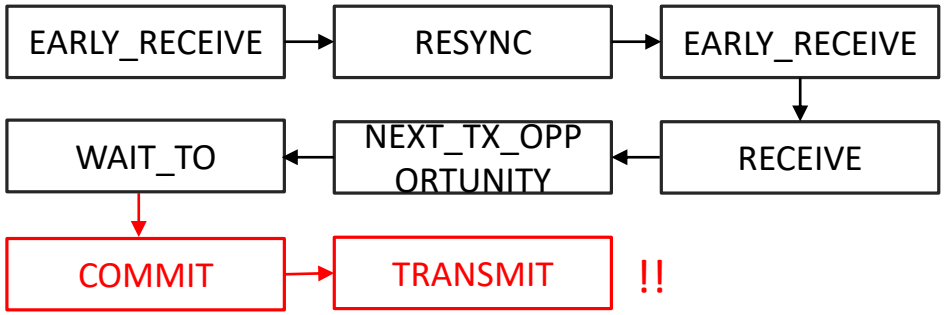
#5 non-PLCA nodes preventing PLCA from starting (2)

- In the **very unlikely** case where non-PLCA enabled nodes are sending packets at a rate which is higher than the recv_beacon_timer frequency, the node in charge of sending the BEACON may never reach the SEND_BEACON state.

- Additionally, the BEACON may be sent when CRS is asserted, resulting in a collision on the line (performance loss)

- Finally, a node detecting a false carrier (recv_timer expired) *may* receive a packet (go to RECEIVE state) and get a new TO even if a *new* BEACON has not been received first (i.e. curID may be out of sync).
  - This may create a collision (perf. loss)
  - It may also create a false BEACON detection (perf. loss)

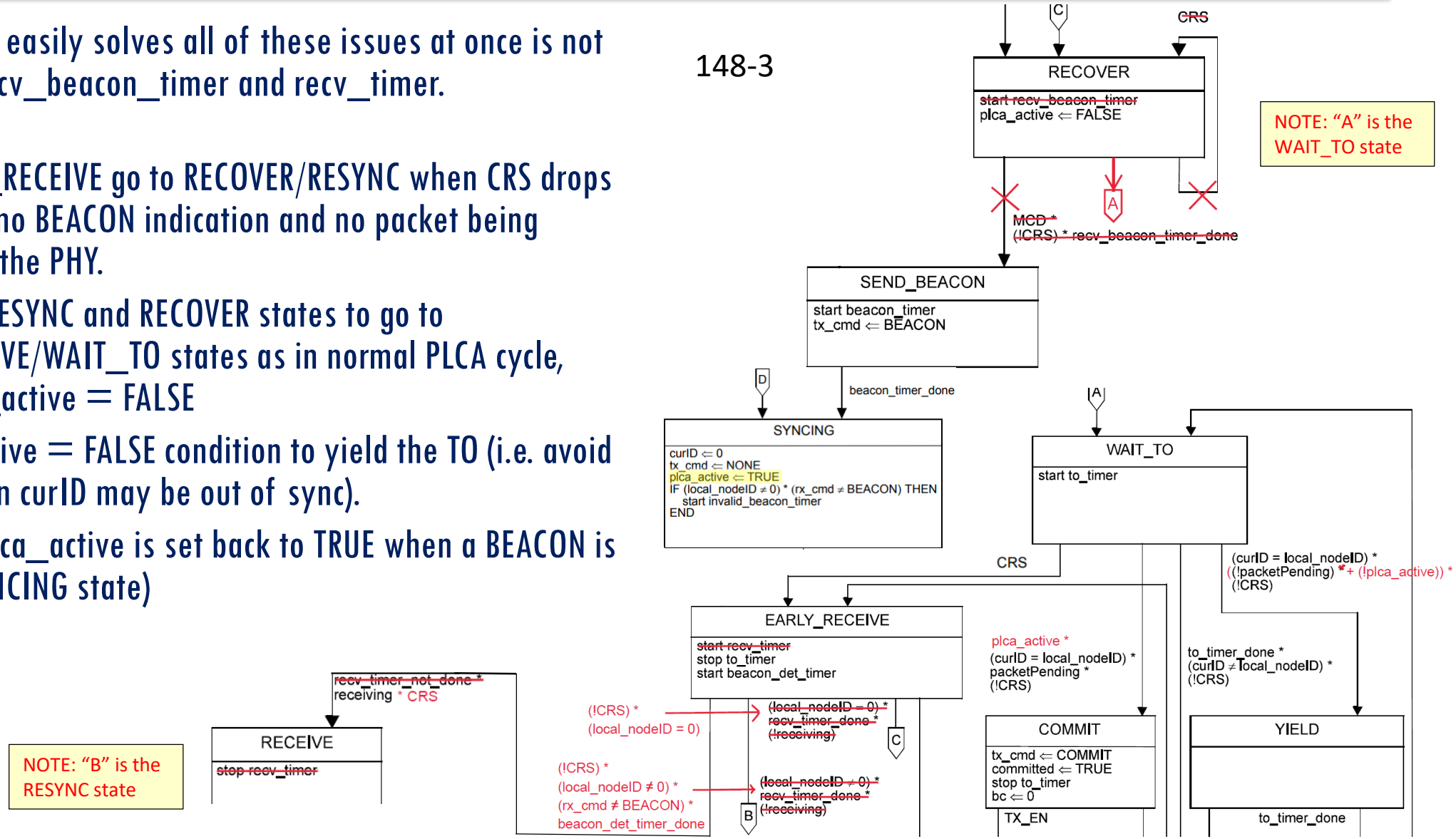




#5 non-PLCA nodes preventing PLCA from starting (fix)

The solution that easily solves all of these issues at once is not relying on the recv_beacon_timer and recv_timer.

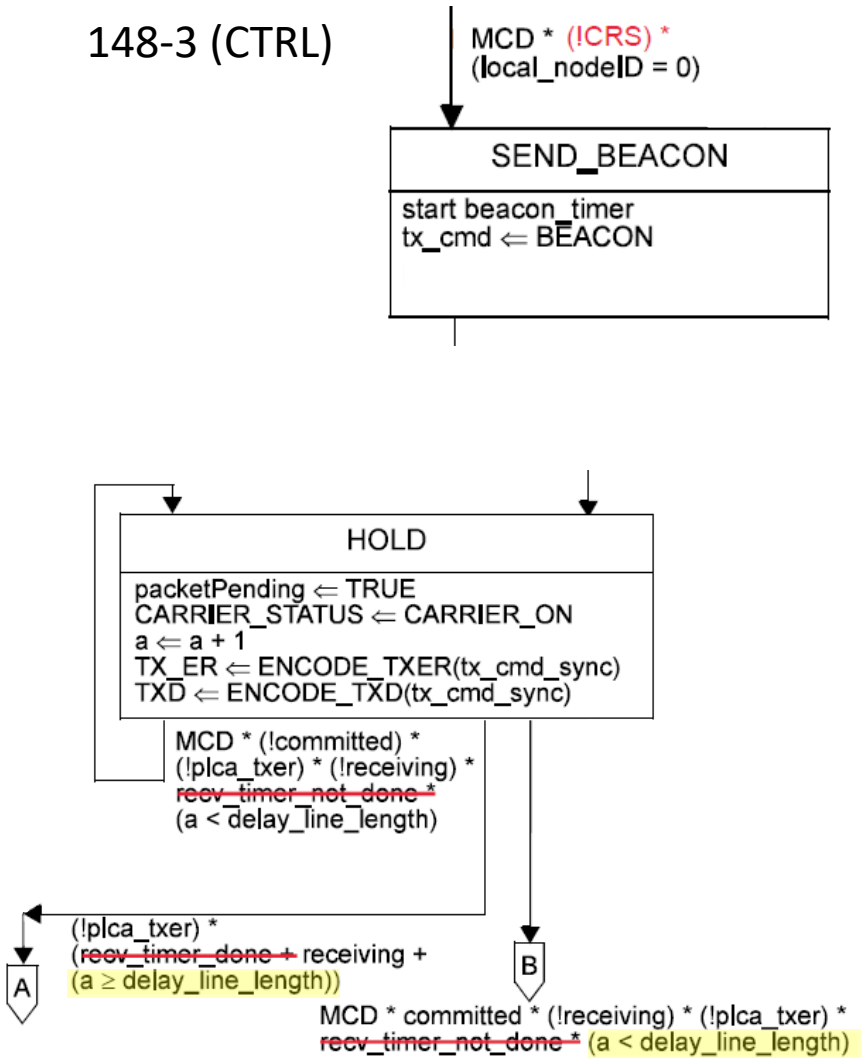
- From EARLY_RECEIVE go to RECOVER/RESYNC when CRS drops and there is no BEACON indication and no packet being conveyed by the PHY.
- Allow from RESYNC and RECOVER states to go to EARLY_RECEIVE/WAIT_TO states as in normal PLCA cycle, having plca_active = FALSE
- Use plca_active = FALSE condition to yield the TO (i.e. avoid sending when curID may be out of sync).
- Eventually plca_active is set back to TRUE when a BEACON is received (SYNCING state)



#5 non-PLCA nodes preventing PLCA from starting (fix)

Furthermore, wait for CRS to be de-asserted (line free) before sending the BEACON. This decreases significantly the chance to create a collision with a BEACON

Finally, the recv_timer was used in early drafts by the DATA State Diagram to exit the HOLD state during a RECOVERY. But this is already accomplished by the check against the delay_line_length. Therefore is safe to just remove this checks and remove the timer from the draft.



COMMENTS MAP

Comments #03-20, #03-23

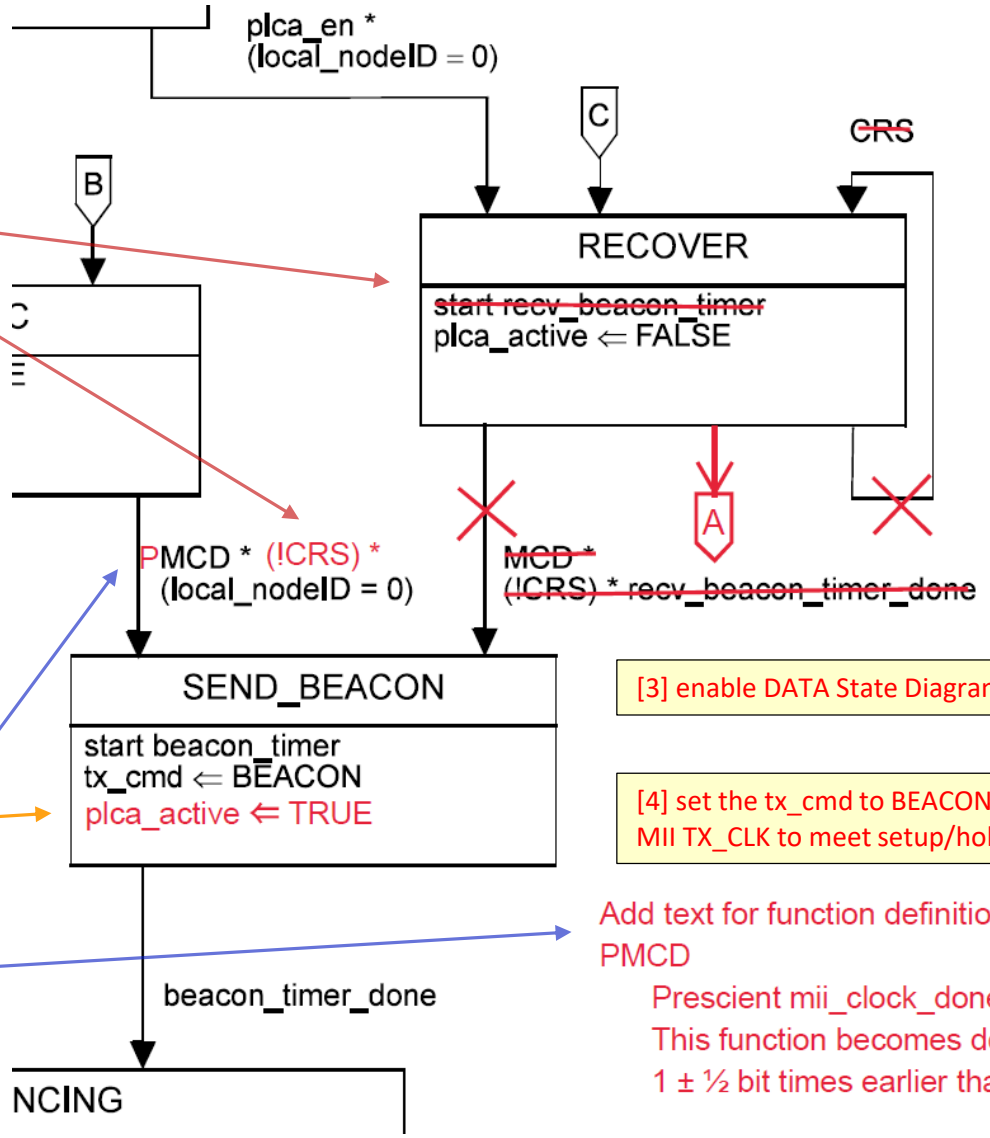
Fixes to Fig. 148-3 (PLCA Control State Diagram)

[1] The node with local_nodeID = 0 may not be able to send a BEACON in the unlikely situation that non-PLCA enabled nodes keep sending packets at a rate higher than the recv_beacon_timer expiration. In such case the PLCA Control State diagram would be stuck in the recover state.

[2] The Control state diagram may loop from EARLY_RECEIVE and RESYNC state because of CRS being continuously asserted. This may cause a false detection of the BEACON if the recv_timer elapses when CRS=TRUE at the end of a packet where the length of the CRS is compatible with the length of a BEACON.

[3] The BEACON may not be sent when PLCA is first enabled due to the Data state diagram being in normal state with plca_status = FAIL, preventing this one to become OK.

[4] Achieve proper synchronization with MII TX_CLK in the PLCA Data Stete Diagram



[1] The idea is that node with ID = 0 goes through a cycle of TOs with plca_active = FALSE before sending a BEACON. This prevents it from sending data during this cycle and allows other nodes to complete the previous cycle undisturbed. The CRS is further tested before sending the BEACON to reduce the chance of collisions in presence of non-PLCA enabled nodes.

[3] enable DATA State Diagram when BEACON is being sent

[4] set the tx_cmd to BEACON just before the posedge of the MII TX_CLK to meet setup/hold requirement.

Add text for function definition:
PMCD
Prescient mii_clock_done function.
This function becomes done exactly
1 ± ½ bit times earlier than mii_clock_done

Comment #03-23

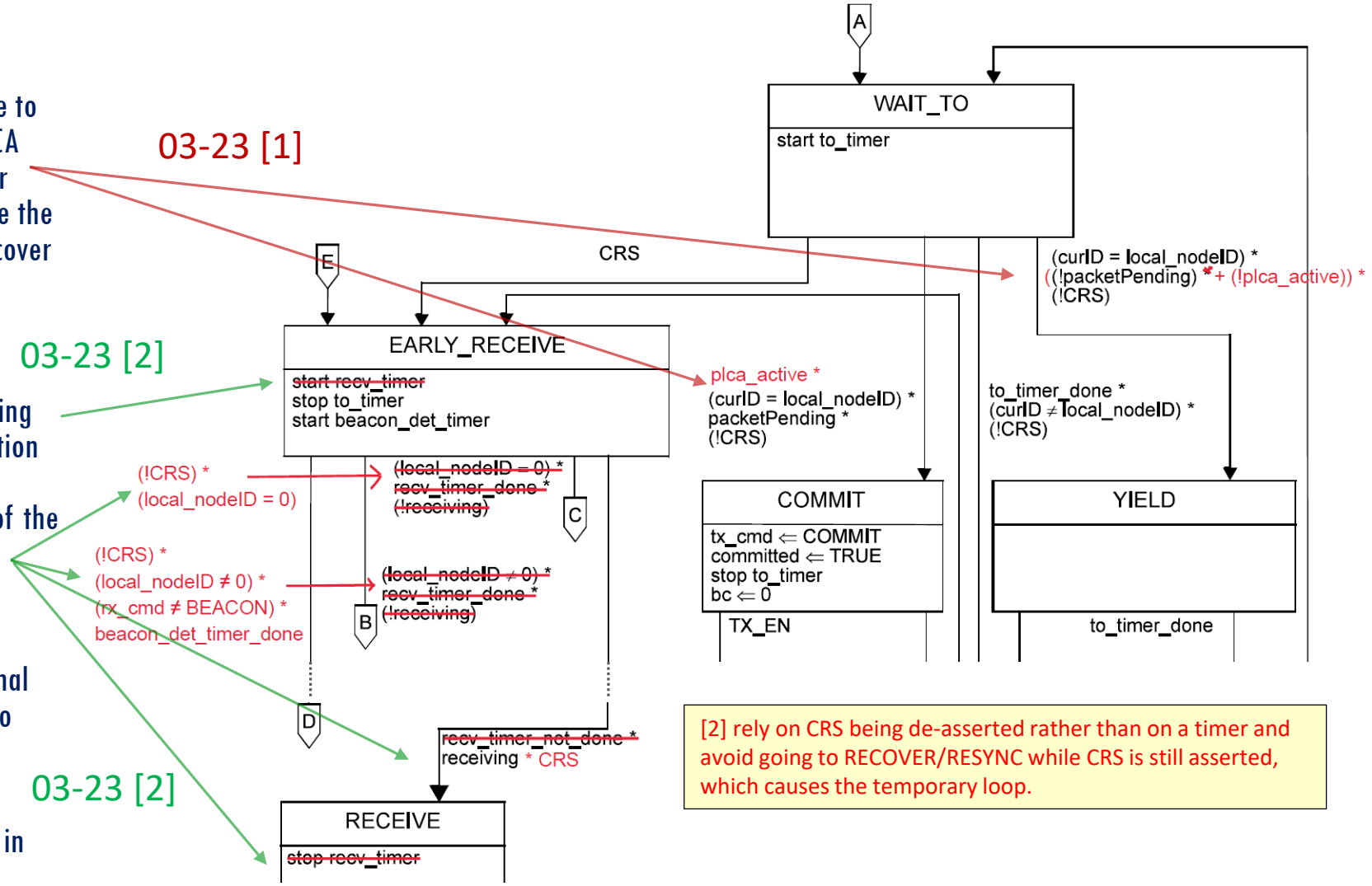
Fixes to Fig. 148-3 (PLCA Control State Diagram)

[1] The node with local_nodeID = 0 may not be able to send a BEACON in the unlikely situation that non-PLCA enabled nodes keep sending packets at a rate higher than the recv_beacon_timer expiration. In such case the PLCA Control State diagram would be stuck in the recover state.

[2] The Control state diagram may loop from EARLY_RECEIVE and RESYNC state because of CRS being continuously asserted. This may cause a false detection of the BEACON if the recv_timer elapses when CRS=TRUE at the end of a packet where the length of the CRS is compatible with the length of a BEACON.

[3] The BEACON may not be sent when PLCA is first enabled due to the Data state diagram being in normal state with plca_status = FAIL, preventing this one to become OK.

[4] Achieve proper synchronization with MII TX_CLK in the PLCA Data Stete Diagram



[2] rely on CRS being de-asserted rather than on a timer and avoid going to RECOVER/RESYNC while CRS is still asserted, which causes the temporary loop.

Comment #03-23

Fixes to Fig. 148-3 (PLCA Control State Diagram)

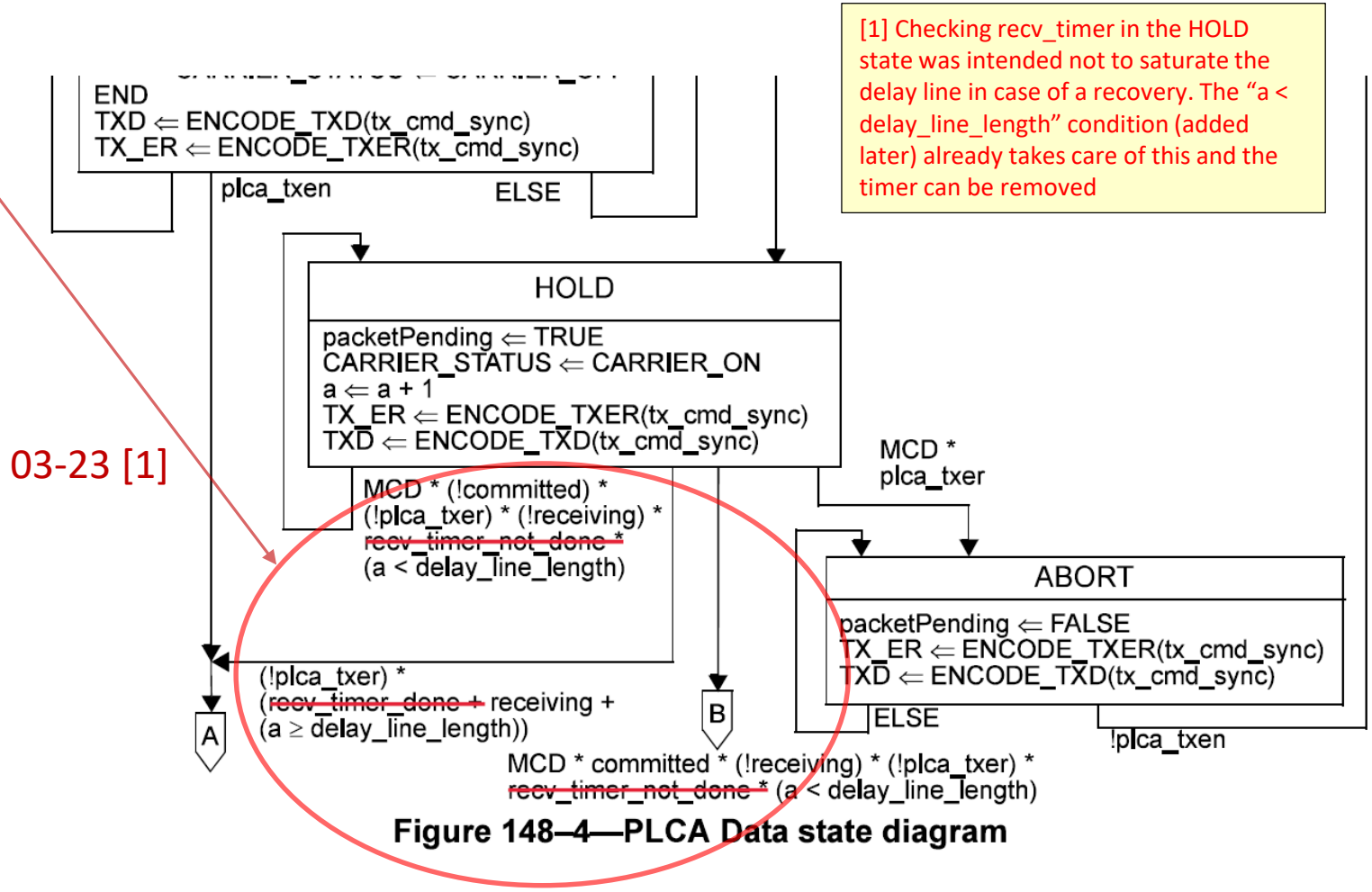
[1] The node with local_nodeID = 0 may not be able to send a BEACON in the unlikely situation that non-PLCA enabled nodes keep sending packets at a rate higher than the recv_beacon_timer expiration. In such case the PLCA Control State diagram would be stuck in the recover state.

[2] The Control state diagram may loop from EARLY_RECEIVE and RESYNC state because of CRS being continuously asserted. This may cause a false detection of the BEACON if the recv_timer elapses when CRS=TRUE at the end of a packet where the length of the CRS is compatible with the length of a BEACON.

[3] The BEACON may not be sent when PLCA is first enabled due to the Data state diagram being in normal state with plca_status = FAIL, preventing this one to become OK.

[4] Achieve proper synchronization with MII TX_CLK in the PLCA Data Stete Diagram

[1] Checking recv_timer in the HOLD state was intended not to saturate the delay line in case of a recovery. The "a < delay_line_length" condition (added later) already takes care of this and the timer can be removed

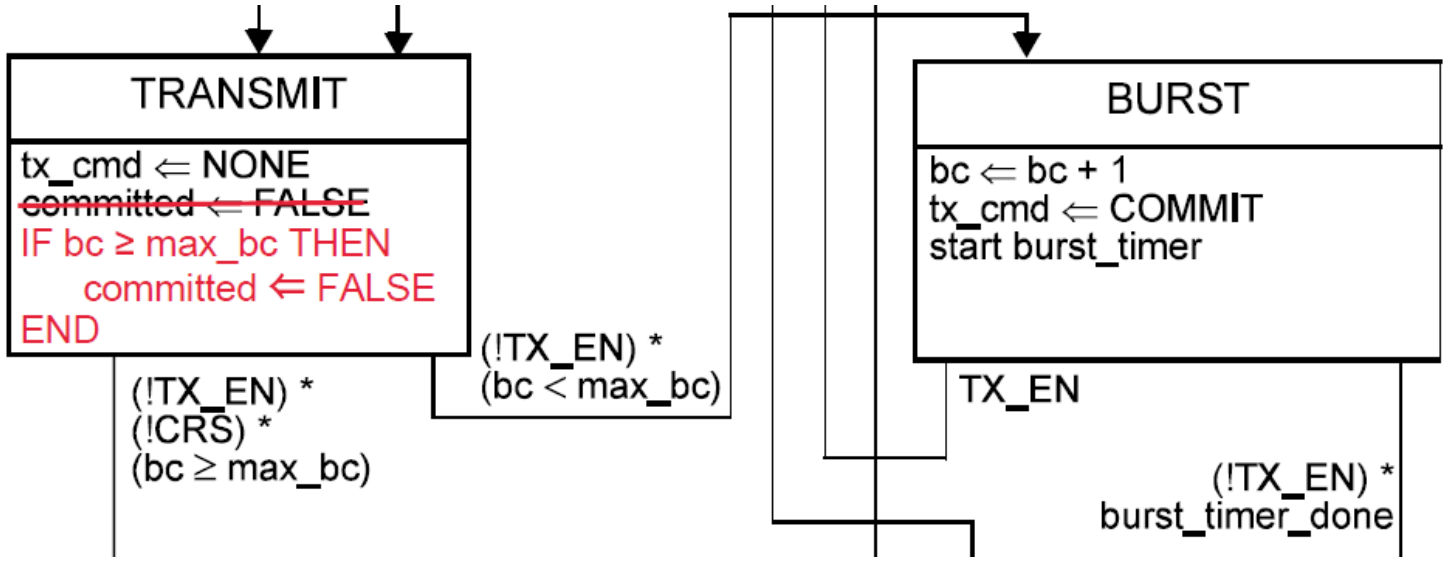


03-23 [1]

Figure 148–4—PLCA Data state diagram

Fixes to Fig. 148-3 (PLCA Control State Diagram)

[5] setting the "committed" variable to FALSE, unconditionally, may prevent the PLCA Data State Diagram from bursting.

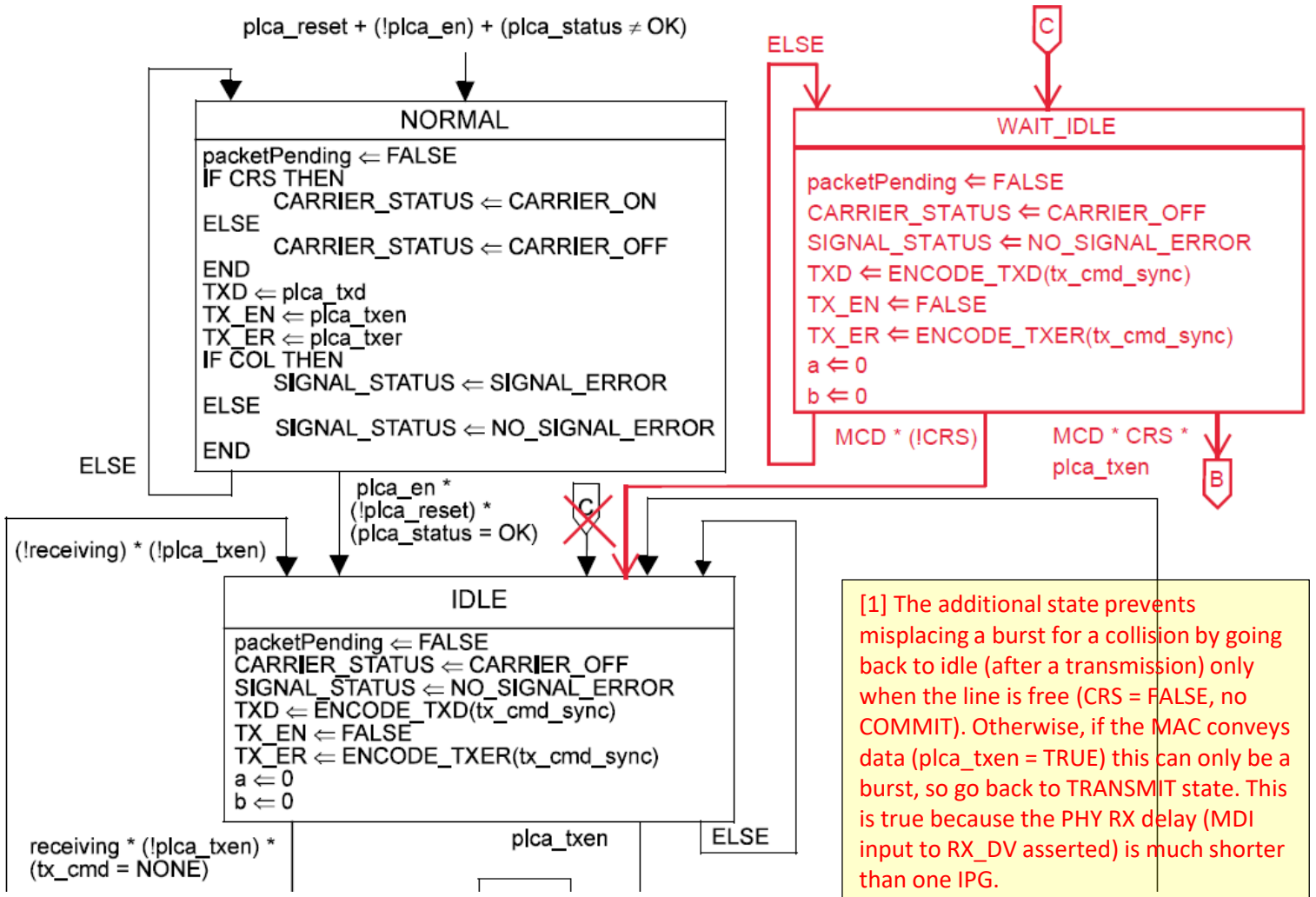


[5] just keep the committed variable set during a burst, so that the DATA Stat Diagram will always be able to send

Comment #03-22

Fixes to Fig. 148-4 (PLCA Data State Diagram)

[1] According to Clause 147 PCS Receive State Diagram the COMMIT request is looped back into a COMMIT indication via MII. If the implementation does not handle this correctly, the burst mode may not work (always trigger a collision). An additional state is required to clarify the behavior of the state diagram when bursting.



[1] The additional state prevents misplacing a burst for a collision by going back to idle (after a transmission) only when the line is free (CRS = FALSE, no COMMIT). Otherwise, if the MAC conveys data (plca_txen = TRUE) this can only be a burst, so go back to TRANSMIT state. This is true because the PHY RX delay (MDI input to RX_DV asserted) is much shorter than one IPG.

THANK YOU!