Analysis of scenarios in MACMERGE in 10M half-duplex

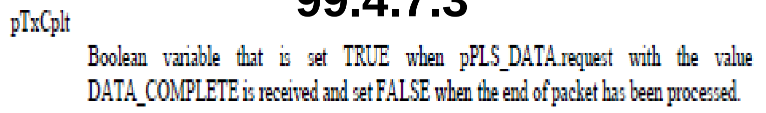
Lokesh Kabra

Synopsys

MACMERGE TX/RX S/M transitions

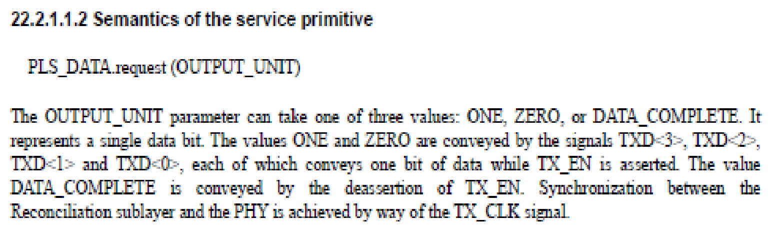
- In Transmit Processing, packet transfer ends when eMAC/pMAC triggers e/pTxCmplt = 1 respectively.
  - As per definition, e/pTxCmplt = 1 , when e/pPLS_DATA.request = DATA_COMPLETE
  - As per Clause 22.2.1.1.2, when DATA_COMPLETE=1, TX_EN is de-asserted
  - On collision event, p/eMAC/RS desserts TX_EN after transmitting JAM pattern
  - Hence packet transfer ends with collision event

99.4.7.3



pTxCplt

Boolean variable that is set TRUE when pPLS_DATA.request with the value DATA_COMPLETE is received and set FALSE when the end of packet has been processed.



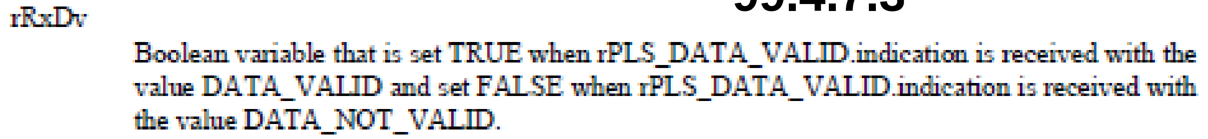
22.2.1.1.2 Semantics of the service primitive

PLS_DATA.request (OUTPUT_UNIT)

The OUTPUT_UNIT parameter can take one of three values: ONE, ZERO, or DATA_COMPLETE. It represents a single data bit. The values ONE and ZERO are conveyed by the signals TXD<3>, TXD<2>, TXD<1> and TXD<0>, each of which conveys one bit of data while TX_EN is asserted. The value DATA_COMPLETE is conveyed by the deassertion of TX_EN. Synchronization between the Reconciliation sublayer and the PHY is achieved by way of the TX_CLK signal.

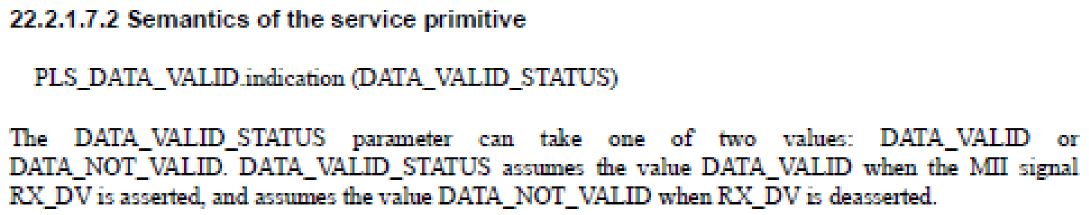
- Similarly in Receive processing, the ingress packet/fragment transfer ends when rRxDv = 0
  - rRxDv is defined as shown here
  - As per Cl 22.2.1.7.2, when RX_DV is de-asserted (DATA_NOT_VALID), rRxDv becomes 0
  - When a collision event occurs, the remote transmitter will abort packet/fragment transmission after sending JAM pattern

99.4.7.3



rRxDv

Boolean variable that is set TRUE when rPLS_DATA_VALID.indication is received with the value DATA_VALID and set FALSE when rPLS_DATA_VALID.indication is received with the value DATA_NOT_VALID.

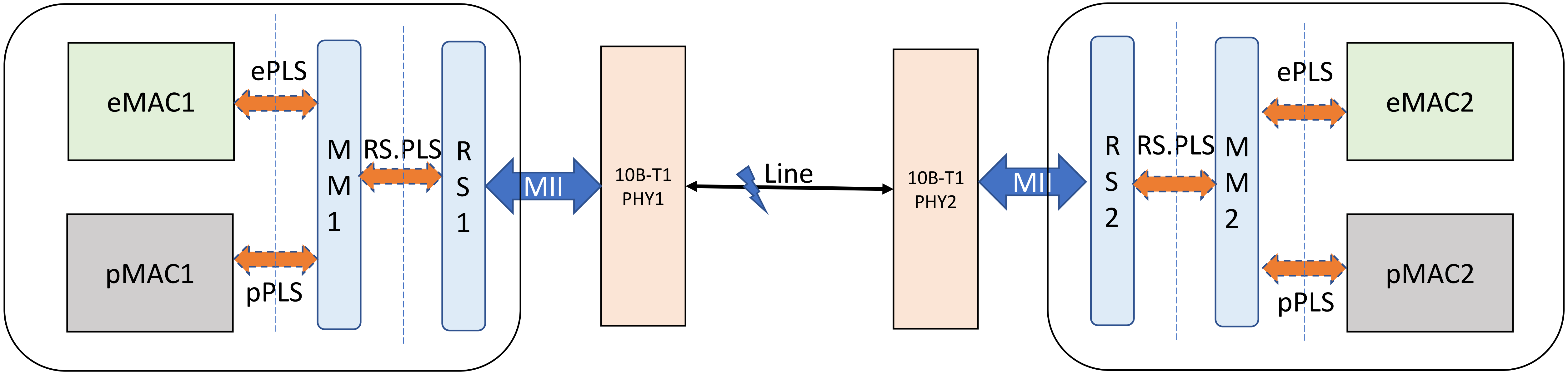


22.2.1.7.2 Semantics of the service primitive

PLS_DATA_VALID.indication (DATA_VALID_STATUS)

The DATA_VALID_STATUS parameter can take one of two values: DATA_VALID or DATA_NOT_VALID. DATA_VALID_STATUS assumes the value DATA_VALID when the MII signal RX_DV is asserted, and assumes the value DATA_NOT_VALID when RX_DV is deasserted.

MACMERGE Reference setup



Block diagram representation of the point-2-point 10BASE-T1S link with MACMERGE

- The ePLS & pPLS are the respective PLS service primitives to the eMAC & pMAC respectively while the RS.PLS is the service primitive between the MM and the RS layer.
- The MM function controls/transfers the PLS service primitives between the eMAC/pMAC and the RS layer which in turn interfaces with the PHY on a MII interface.

Half Duplex in MACMERGE : Issue #1

- In Clause 99.4.4, Transmit Processing states "*If a PLS_CARRIER.indication is received from the PLS, PLS_CARRIER.indications with the same CARRIER_STATUS shall be sent to the pMAC and the eMAC*"

- Half-duplex MAUs and PHYs inherently loop back their own transmissions onto the media through their receivers. The half-duplex MAC looks for this, and if carrier drops out during a transmission, it increments the carrierSense error counter (Clause 5.2.4.2 CarrierSenseTest & 5.2.2.1.13 aCarrierSenseErrors). This results in a basic problems

- When say, pMAC1 is transmitting a preemptable packet, the CRS is transferred back to the associated eMAC1 also which in turn will defer its transmission until the CRS is removed. This means the **eMAC1 will NOT be able to pre-empt the packet transfer of pMAC1 after the CRS loop delay**. This defeats the objective of pre-emption of Low-priority packets with Express packet transfer.
  - **Solution :** MM blocks the ePLS_CARRIER.indication to the eMAC when the associated pMAC is transmitting a packet (ePLS_CARRIER.indication = CARRIER_OFF). But when pMAC (MM) is not transmitting, the PLS_CARRIER.indication with the same CARRIER_STATUS shall be sent to both the pMAC & eMAC.
    - When eMAC is transmitting, pMAC will defer (lower priority)
    - When neither is transmitting, collisions are reduced due to carrierSense deference process during reception of data from link partner

Half Duplex in MACMERGE : Issue #2

- Considering that Clause 99.4.4 is fixed as proposed in previous slide, it has a side effect as explained below

- When pMAC1 is pre-empted by eMAC1, the state machine transitions as shown with a IPG period of 96bits in RESUME_WAIT state during which there is no data transfer and line is IDLE

- This makes the CRS go low for the which in turn makes the pPLS_CARRIER.indication as “CARRIER_OFF”. This will **result in the pMAC to increment the “aCarrierSenseErrors” counter for the pre-emption event.**

Solution

- The MM sets the pPLS_CARRIER.indication as “CARRIER_ON” when the state enters P_TX_COMPLETE from TX_MCRC with *preempt* (*pTxCplt = 0*)
- It is made same as RS.PLS_CARRIER.indication only when state machine enters when state enters IDLE_TX_PROC from P_TX_COMPLETE with *pTxCplt = 1*

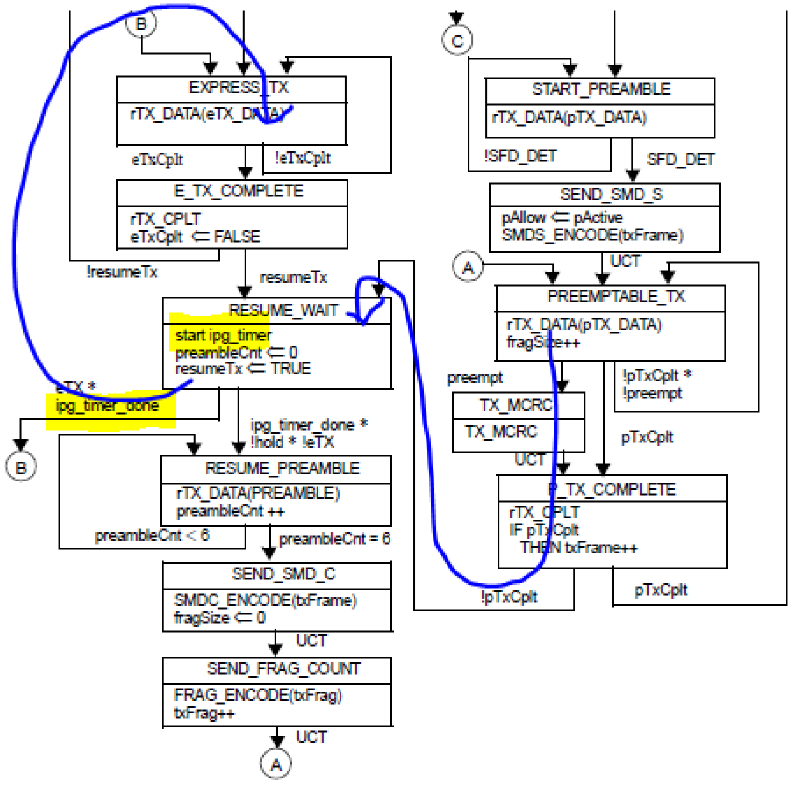
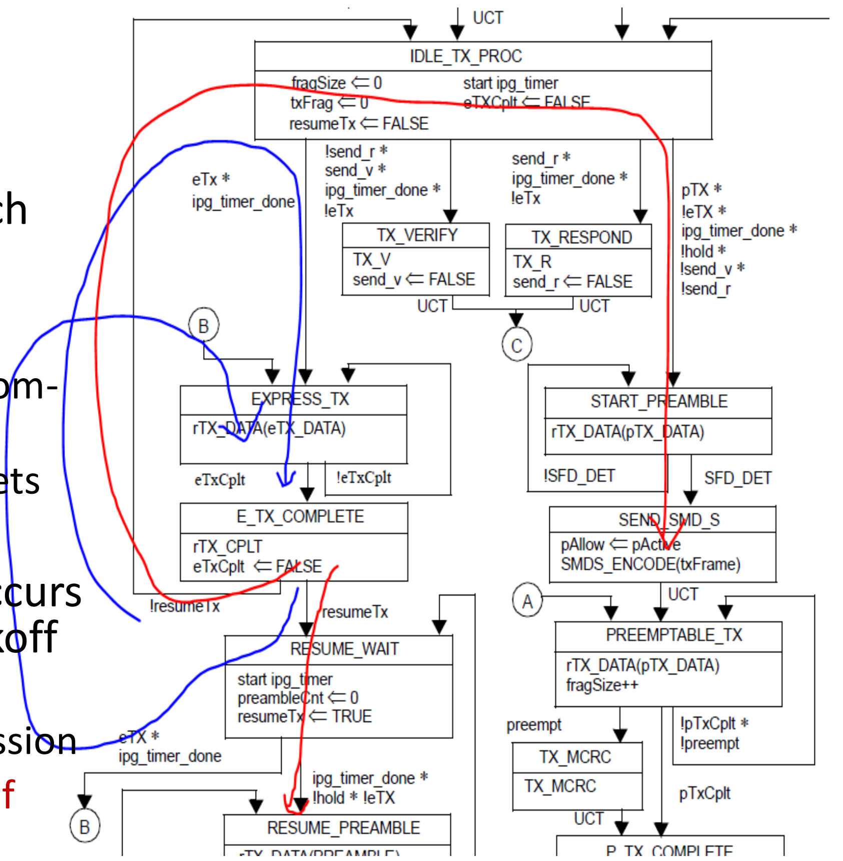


Figure 99–5—Transmit Processing state diagram

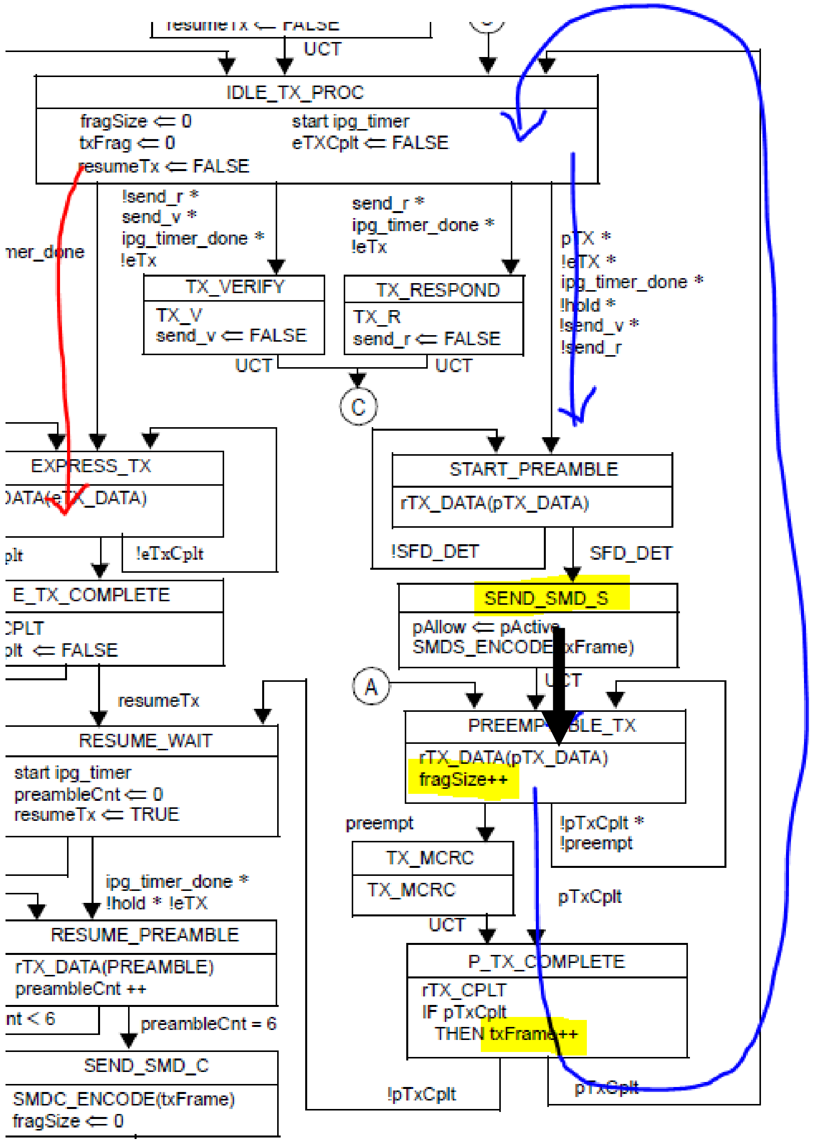
Collision Scenarios in MACMERGE - 1

- Collision during Express Frame transmission in EXPRESS_TX state
- On collision, eMAC will make *eTxCplt = 1* & hence state moves to E_TX_COMPLETE
- IPG period of 96 bits in IDLE_TX_PROC or RESUME_WAIT states during which line is IDLE
- 4 possible transition paths as shown
  a. Blue (okay) paths only when Express packets gets retransmitted due to random-backoff interval = 0 OR pMAC not ready (*pTx = 0*)
  b. Red (not ok) paths when random-backoff interval > 0 and preemptable packets (*pTX = 1*) are in queue
- Impact of "red" path transitions : Transmission of preemptable fragment occurs immediately after min-ipg (*ipg_timer_done*) while eMAC is in collision/backoff period *(!eTx).*
  1. Latency delay of express packet increases due to preemptable packet transmission
  2. Random backoff algorithm intent violated and leads to increased probability of subsequent collisions especially when far-end also has both *eTx & pTX = 1*
- At collision event, if PLS.SIGNAL.indication received from RS is sent as-is to both pMAC & eMAC, then both MACs will register a collision and back-off. This should be prevented and only the MAC which was transmitting (selected by MM) at time of collision should get the collision event



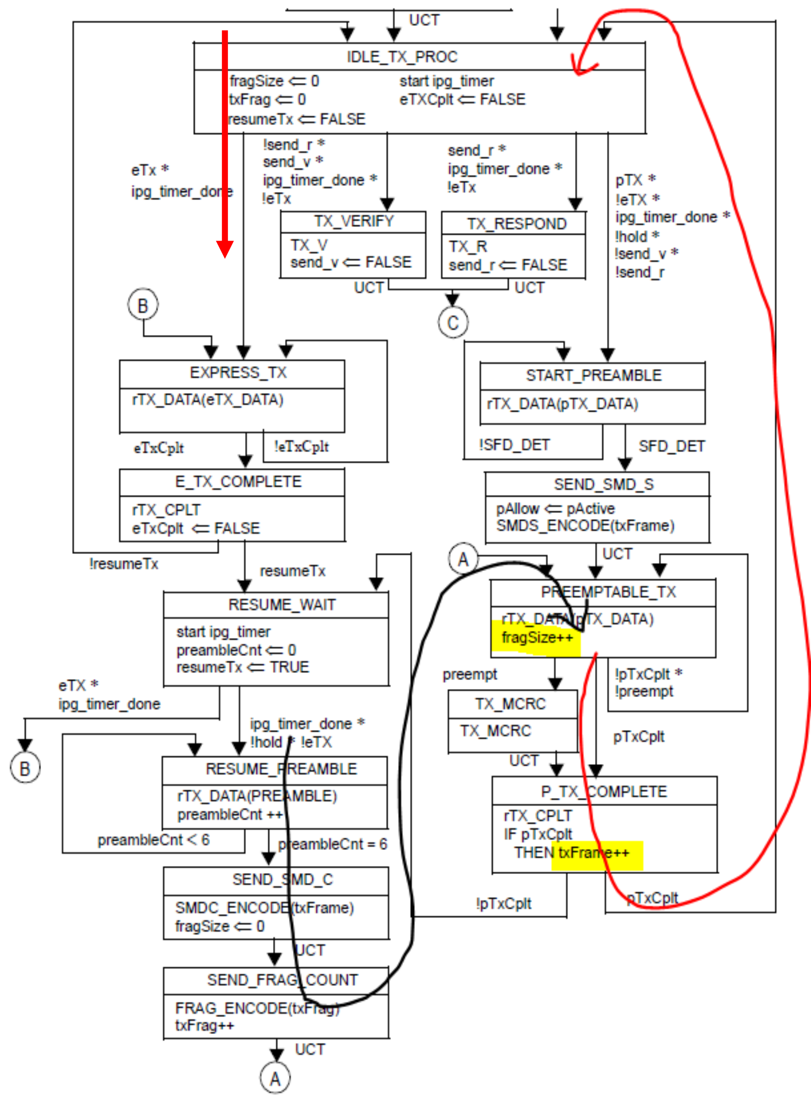
Collision Scenarios in MACMERGE - 2

- Collision during Preemptable SMD-S (first fragment) frame transmission in PREEMPTABLE_TX state
- On collision, pMAC will make *pTxCplt* = 1 & *preempt* will be 0
- State will move to IDLE_TX_PROC because "pre-empt = 0 & pTxCplt =1" after collision and wait for IPG (96 bits) period to be over; After that
  - a. Blue (okay) paths only when preemptable packets gets retransmitted when eTX = 0 after random back-off; Side effect is that txFrame++, fragSize++ even though the frame/fragment transmission was not successful.
  - b. Red path when eTx = 1 during random backoff of pMAC
- Impact of "red" path transition : Transmission of Express packet can occur immediately (after min-ipg) while pMAC is in random-backoff
  1. Random backoff algorithm intent violated and leads to increased probability of subsequent collisions especially when far-end also has both eTx & pTX = 1. However alternative view is that eMAC is pre-empting the pMAC which is the objective of MM & hence it should be okay



Collision Scenarios in MACMERGE - 3

- During Preemptable SMD-C (not first fragment) frame transmission in PREEMPTABLE_TX state
- State will move to IDLE_TX_PROC because “pre-empt = 0 & pTxCplt =1” after collision; This has major impact
    1. pMAC will declare late collision and abort frame transmission because collision occurred after minFrame size (i.e not in first fragment)
- If eTx = 1 after ipg_timer_done, state moves to EXPRESS_TX
    2. Random backoff algorithm intent violated and leads to increased probability of subsequent collisions especially when far-end has both eTx & pTX = 1



Collision Scenarios in MACMERGE Receive

rRX_DV -> 0 prematurely due to collision and backoff by far-end transmitter

1. During Express packet reception (Path E)
   - Okay with no impact
2. During SMD-S Fragment reception (Path S)
   - State machine moves to FRAME_COMPLETE -> IDLE_RX_PROC
   - Hence state machine expects SMD-S again which is okay since far-end will retransmit SMD-S
3. During SMD-C Fragment reception (Path C)
   - State machine moves to FRAME_COMPLETE -> IDLE_RX_PROC
   - If other end retransmits SMD-C, it will discard (red path) because it is now looking for a new SMD-S

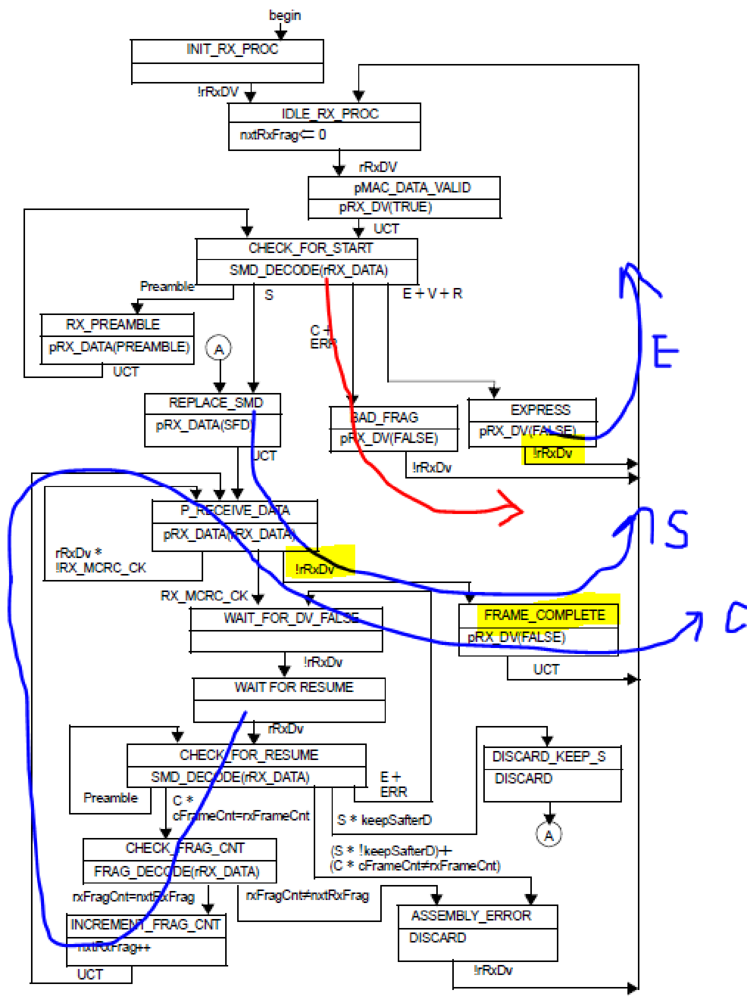


Figure 99–6—Receive Processing state diagram

Summary of Collision Issues

1. If collisions occur when both eMAC and pMAC have packets in queue and are ready to transmit, random backoff interval is violated as the other MAC's frame will get transmitted after min IPG of 96 bit-times, while the MAC whose packet transmission had collision is in random backoff

2. If collisions occur when pMAC is transmitting a continuation fragment, it will result in "late collision" event in pMAC and likely packet abort or retransmit of complete packet from SMD-S (wastage of bandwidth) depending on implementation

3. If a station re-transmits a continuation fragment after collision, then it will be discarded by the link-partner's receiver as it is expecting a new SMD (first segment)

4. If PLS_SIGNAL.indication is sent as-is (directly w/o qualification) to both pMAC/eMAC on reception of COL as specified in P802.3de D2.1, then both pMAC/eMAC will abort transmission and back-off; This must be avoided

   - **Solution :** PLS_SIGNAL.indication = SIGNAL_ERROR is sent only to the specific eMAC or pMAC which is actively transmitting and not to the other.

Suggestions to fix

Prevent "late" collisions during transmission of continuation fragment by doing carrier extension & avoid IPG before start of continuation fragment transfer.

- MM transmits Carrier Extension symbol (COMMIT symbol?) in RESUME_WAIT state so that the line is blocked (carrier extension during IPG) before resumption of continuation fragment. This avoids collisions in continuation fragment transfer totally
- Split RESUME_WAIT state into 2 states –
  - RESUME_WAIT_CE (MM transmits carrier extension) :
    - Entry from E_TX_COMPLETE only if express packet transfer completed w/o collision & *resumeTx = 1*
    - Same exit transitions as now
  - RESUME_WAIT_NO_CE (w/o carrier extension) :
    - Entry from P_TX_COMPLETE *(!pTxCplt)* or from E_TX_COMPLETE if express packet transfer resulted in collision
    - Exit only to EXPRESS_TX (B) on *eTx = 1* after random backoff; **This also blocks pMAC from resuming transmission while eMAC is in random backoff**
- This results in IDLE/IPG before transmission of every Express packet which allows **eMACs on both sides an equal opportunity to transmit a packet** after a fragment transfer.
- This also limits the burst on line to a maximum size/delay of 1 express packet + 1 continuation fragment transfer.
- It still allows the local eMAC to pre-empt the pMAC multiple times as desired without further increase of burst size

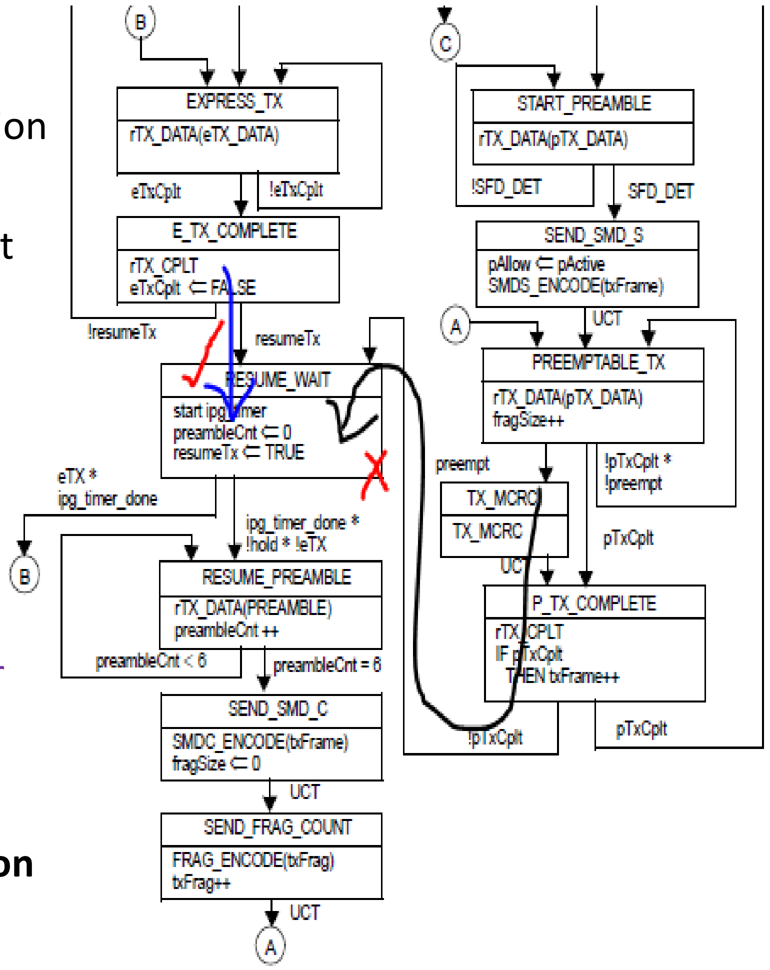
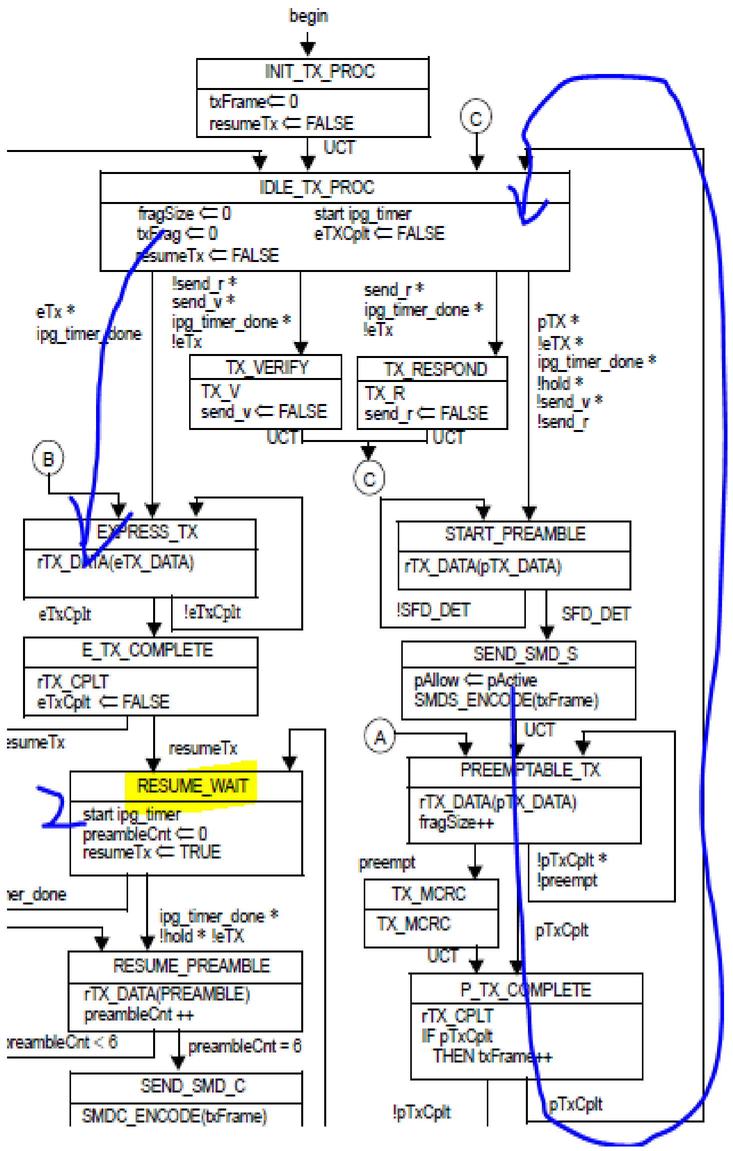


Figure 99–5—Transmit Processing state diagram

Suggestions to fix

- When Collision occur during SMD-S (first) fragment transfer, state machine moves to IDLE_TX_PROC when pMAC is in random backoff
- If *eTx=1* after ipg_timer_done, the express packet transfer starts violating the random back-off. But this can be justified because eMAC has higher priority
- If another collision occurs during that express packet transfer, then state moves to RESUME_WAIT_NO_CE as explained in previous slide which ensures that pMAC will not get any further opportunity to transfer until the express packet transfer is completed successfully
- This ensures that during collision backoff at most 1 MAC from either side of the link is participating as intended, after a max of 2 consecutive collisions
- Only caveat is that eMAC1 and pMAC2 (or vice versa) can arbitrate and pMAC can still win the line. This is a inherent nature of half-duplex and cant be avoided.



Loose ends

- Fragment/Frame counters of preemptable packet

- As per current Transmit Processing S/M, the *txFrame* counter gets incremented even on packet abort after collision in P_TX_COMPLETE state
  - This should not result in any misbehaviour in remote receiver because it decodes the *rxFrameCnt* count value from each SMD-S fragment and compares it against the *cFrameCnt* of each continuation fragment

- Fragment counter are used only in continuation fragments
  - SMD-C (Cont Frag) will not have any collisions after suggested fixes of carrier extension. Hence this the Fragment count in TX and other side’s RX will be in-sync.

Any questions or suggestions ?