

Control Multiplexer, Idle Deletion, Data Detector, and GearBox for Upstream CNU Burst Transmission.

Control Multiplexer

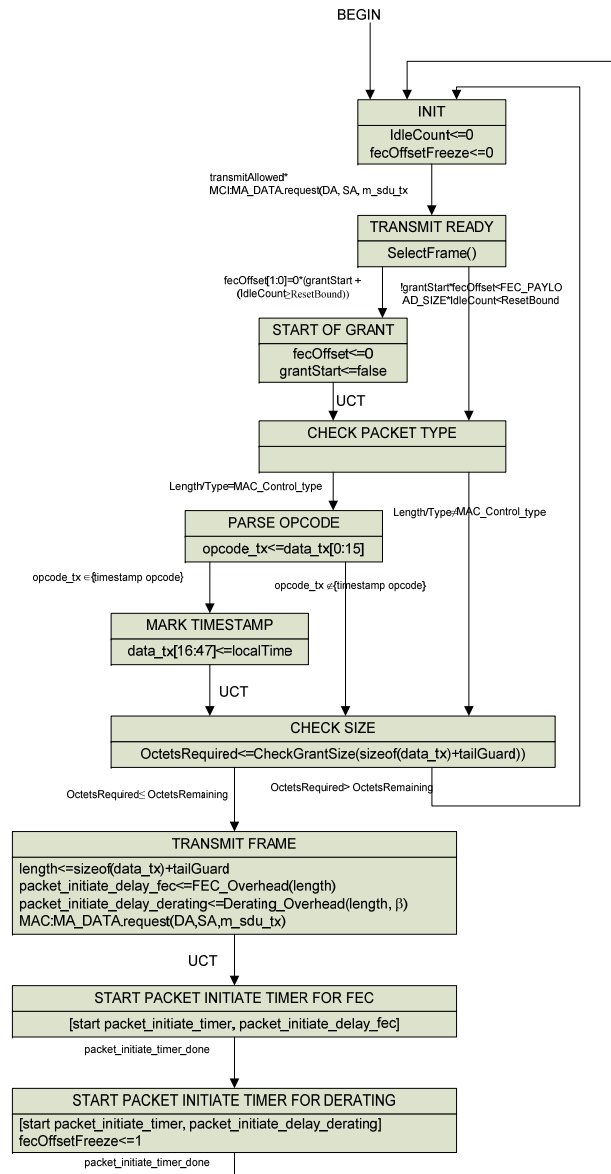
Jin Zhang, Xiaolin Che

Marvell

Functions of CNU Control Multiplexer

- Common for CLT and CNU control multiplexer
 - Select which data unit to send
 - For example, always control unit before data unit.
 - Implementation dependent
 - Force idle period after each data unit for:
 - Insertion of FEC parity
 - Insertion of blank period to adapt to PMD rate.
- CNU only:
 - Identify gap of idles and decide formation of bursts
 - Check if the remaining time is enough for the incoming MAC frame

CNU Control Multiplexer



Key parameters and functions:

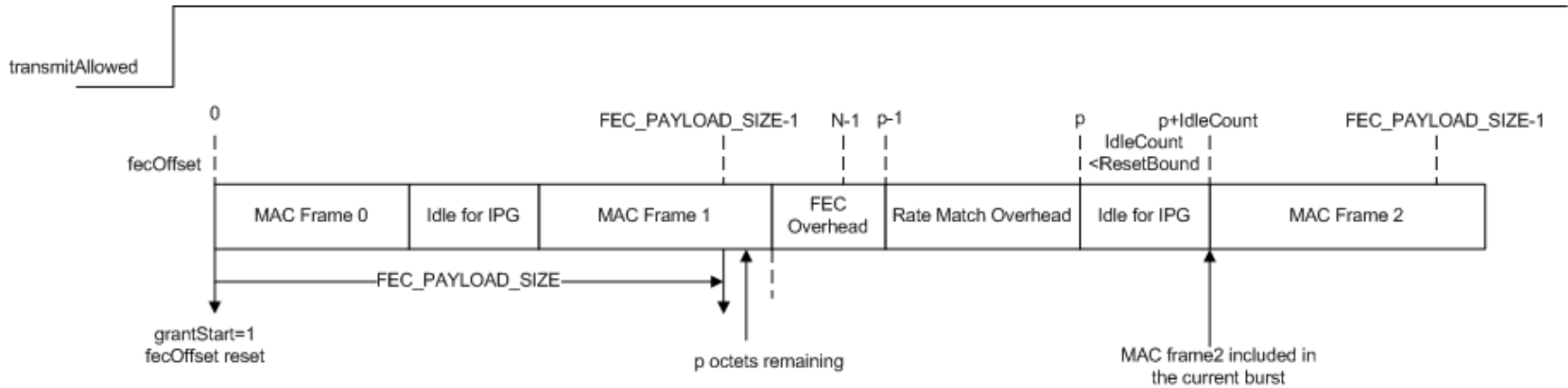
1. ResetBound
2. CheckGrantSize
3. FEC_Overhead and Derating Overhead

ResetBound

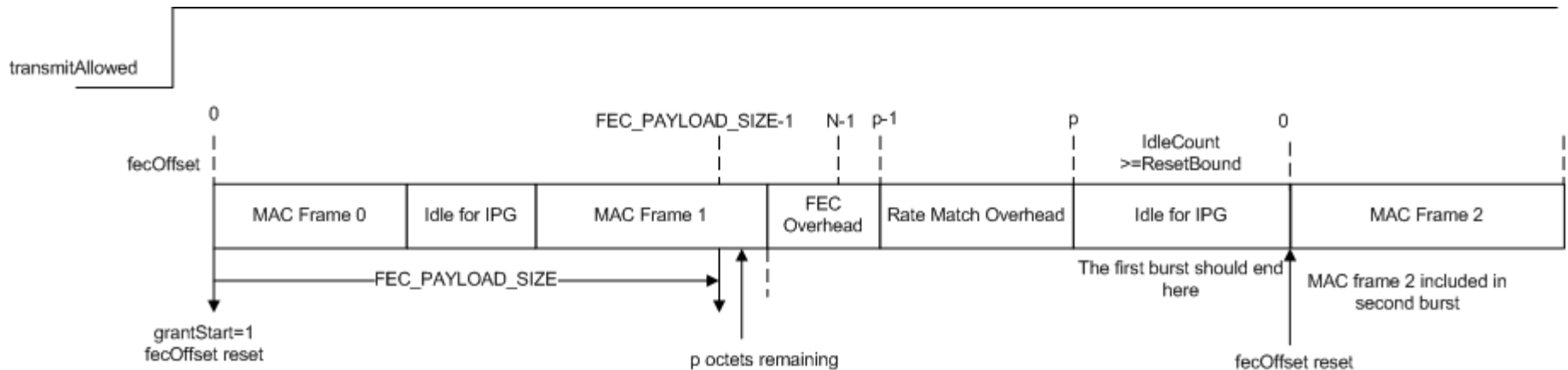
- “IdleCount” counts the natural idles between MAC frames
- Case 1: If $\text{IdleCount} < \text{ResetBound}$, it is more efficient to include the next frame in the current burst.
- Case 2: If $\text{IdleCount} \geq \text{ResetBound}$, terminate the current burst, because this is more efficient.

Function of ResetBound

Case 1:



Case 2:

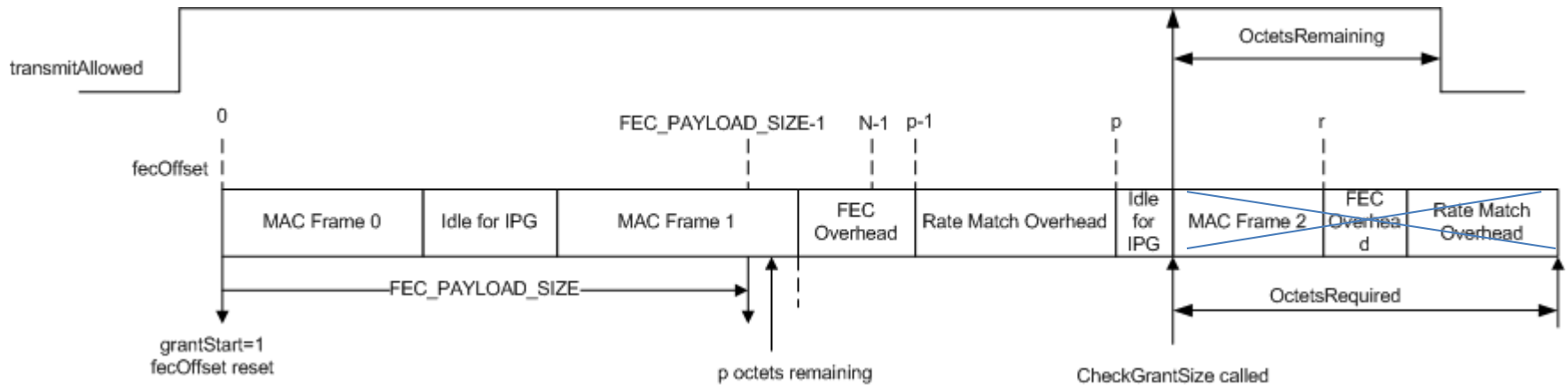


How to Determine ResetBound?

- Part 1: Cover the time used to transmit the remaining p octets from previous MAC frame
 - $p1 = f(p, \beta)$; //similar to CheckGrantSize function
 - Should first decide how to transmit the remaining p octets, where $0 \leq p < \text{FEC_PAYLOAD_SIZE}$
- Part 2: an upper bound of the time used to transmit end marker and start marker
 - Worst case: $p2 = 10 \text{ bits} * (\text{Number of REs for start and end markers}) * \beta/8$
- $\text{ResetBound} \geq p1 + p2$,
 - For efficiency, we can set ResetBound as function of p
 - For simplicity, we can set ResetBound for the worst case of p .

CheckGrantSize Function

- Purpose
 - whenever a new MAC frame is about to transmit, check if the current grant has enough time to allow for it



Calculate CheckGrantSize(length, β)

1. $[n0, n1, n2] = \text{FEC_LUT}(\text{fecOffset} + \text{length});$
2. $\text{Psize} = n0 * \text{FEC_PARITY_SIZE0} +$
 $n1 * \text{FEC_PARITY_SIZE1} +$
 $n2 * \text{FEC_PARITY_SIZE2};$
3. $\text{OctetsRequired} = \text{length} + \text{Psize} * \beta + (\beta - 1) * (\text{fecOffset} + \text{length});$
4. Return OctetsRequired;

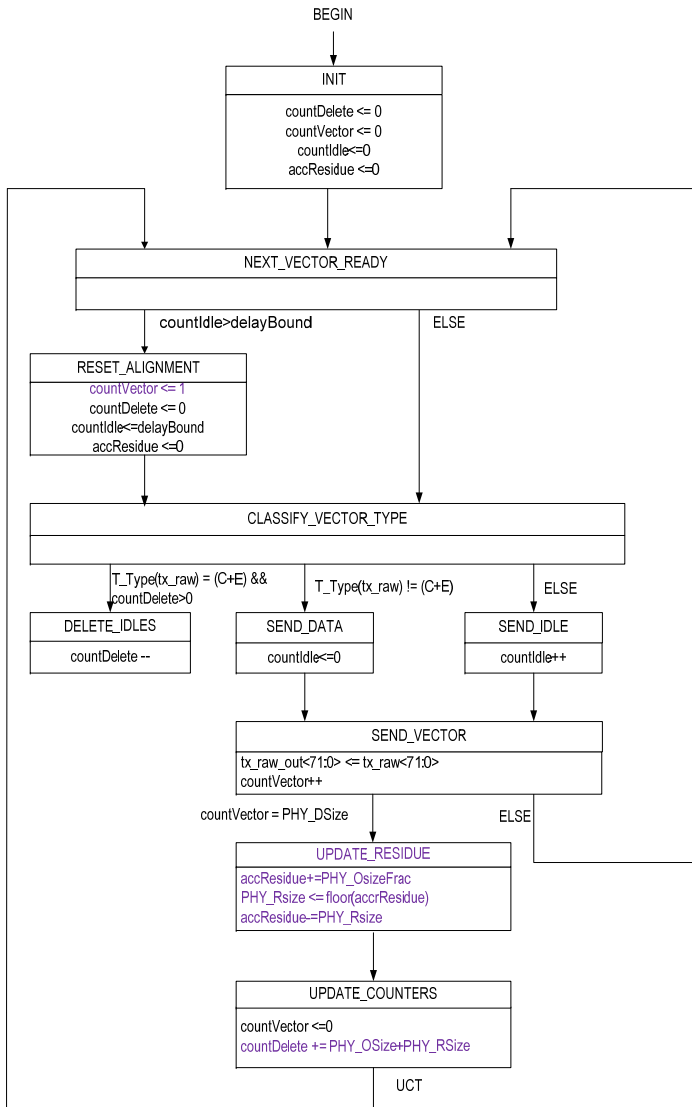
CNU Idle Deletion

- Only delete 72-bit idle vectors inserted from control multiplexer.
 - Delete idle vectors for FEC overhead
 - Delete idle vectors for rate adaptation overhead
- Should not delete idle vectors for inter-packet gap.
- In the long run, should supply exact amount of vectors for PMD rate.
- CNU Only: identify the end of burst, and reset to a transparent mode

Modification of CNU Idle Deletion

- Combine deletion of idles for FEC overhead and rate adaptation overhead into a unified diagram.
- Compensate for the residue of PHY_Osize.
- In the transparent mode, reset the FIFO size.

Modified CNU Idle Deletion



- accResidue: a fractional number to accumulate the residue.

- PHY_RSize: binary number, 1 or 0

- PHY_DSize: integer 220

- FEC_OSize: integer 29

- PHY_OSize:

$$PHY_OSize = \left\lfloor \frac{XGMII_Rate - PCS_Rate}{PCS_Rate} \times (PHY_DSize + FEC_OSize) \right\rfloor + FEC_OSize$$

- PHY_OSizeFrac: fractional part of the actual PHY_OSize. Its precision is implementation dependent.

$$PHY_OSizeFrac = \frac{XGMII_Rate - PCS_Rate}{PCS_Rate} \times (PHY_DSize + FEC_OSize) + FEC_OSize - PHY_OSize$$

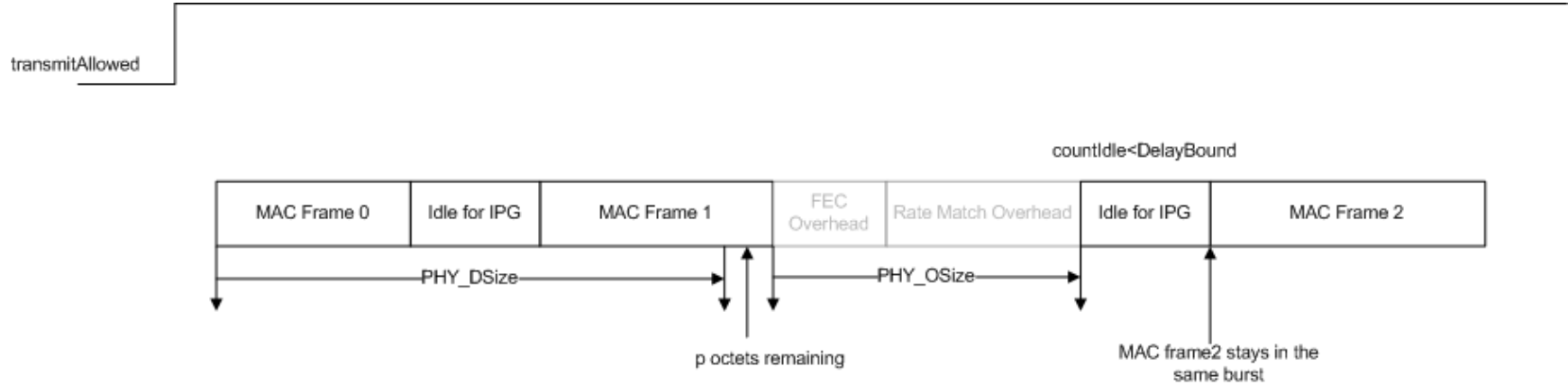
$$PCS_Rate = PMD_Rate \times \frac{64}{65}$$

$$PMD_Rate = \frac{PLCTotalBits}{PLCTotalCycles} \times OFDM_SampleFreq$$

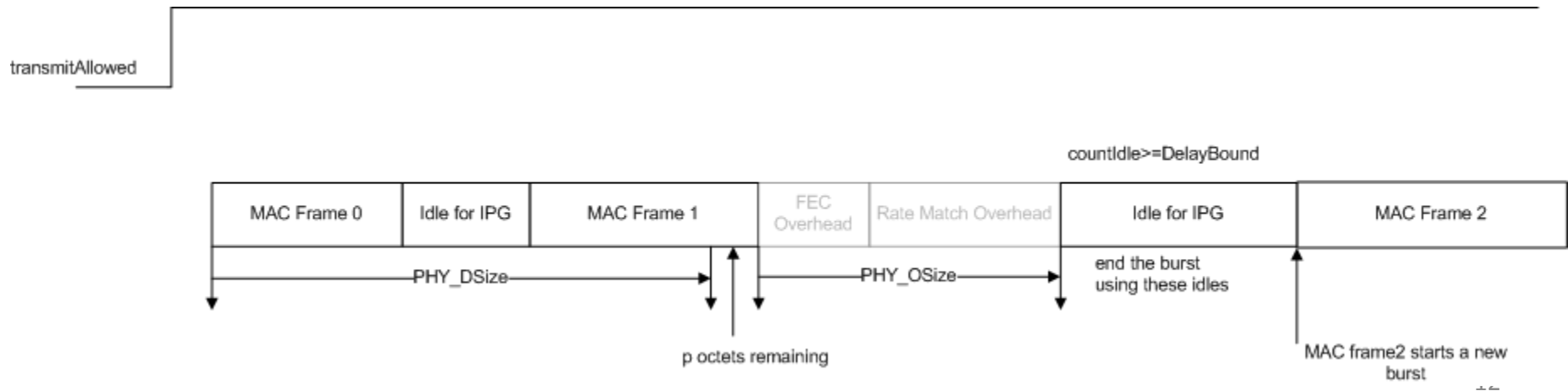
- DelayBound = ResetBound/8

Example

Case 1:



Case 2:



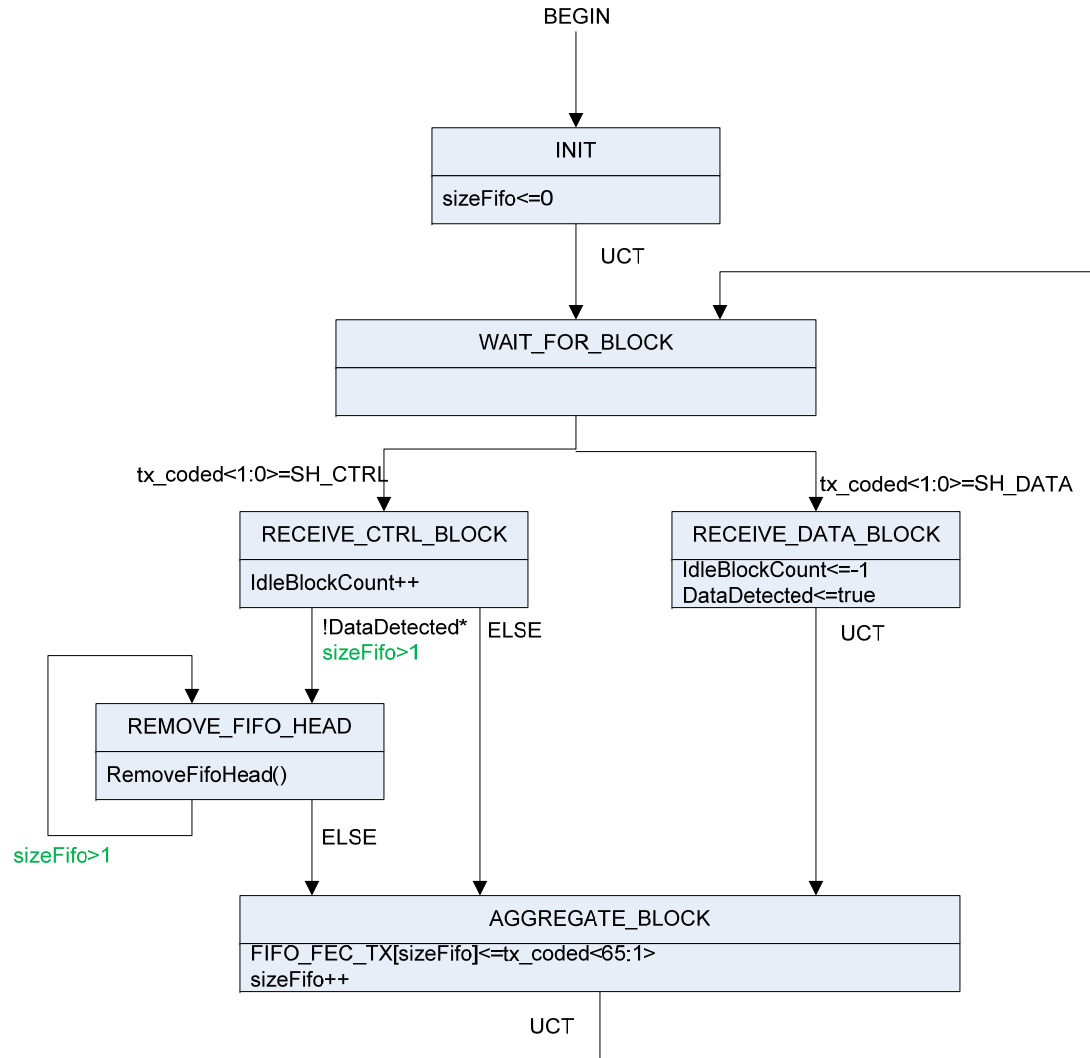
CNU Data Detector

- Control FEC encoder and insert parity.
- Formation of burst
 - Act on arrival of data vector and trigger the burst start.
 - Identify the arrival of idles indicating end of burst.
 - Upon terminating a burst, process the remaining data octets and insert FEC parities and rate adaptation overhead.

CNU Data Detector Input Process

- When no burst data is present, idle deletion is in a “transparent mode”, i.e. passing down all idle vectors.
- CNU data detector shall block all idles in “transparent mode”, except for one idle vector reserved for special use.
- CNU data detector starts to buffer data in FIFO when data vector is detected.

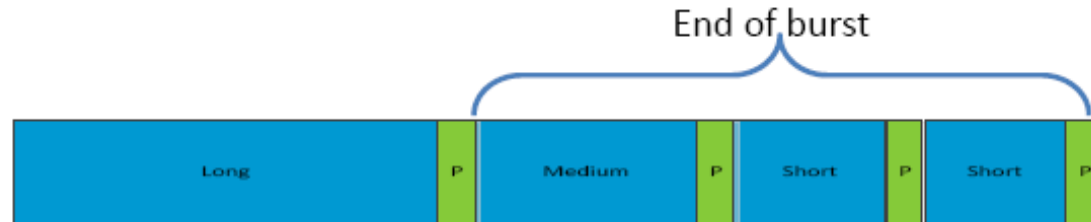
Proposed CNU Data Detector Input Process



CNU Data Detector Output Process

- Insert FEC parities
 - 3 encoders to consider
 - Two options were on debate.[ref: kliger_3bn_02_0314.pdf]
 - Two options have similar TX memory. (not critical)
 - Option 1 has more TX latency: 1.6~6.4us
 - Option 2 has higher implementation complexity, mainly on the memory scheduling and sharing at the receiver.
 - The decision is basically between certain TX latency and possible receiver complexity.
 - In this contribution we show how data detector based on option 1 works
 - Disclaimer: This contribution does not preclude any future possibility of simplification of coding scheme, for example, using 2 instead of 3 FEC codes. In case that TX latency turns out to be a bottleneck during further investigation, we do not preclude other options.

Two options for Data Transmission



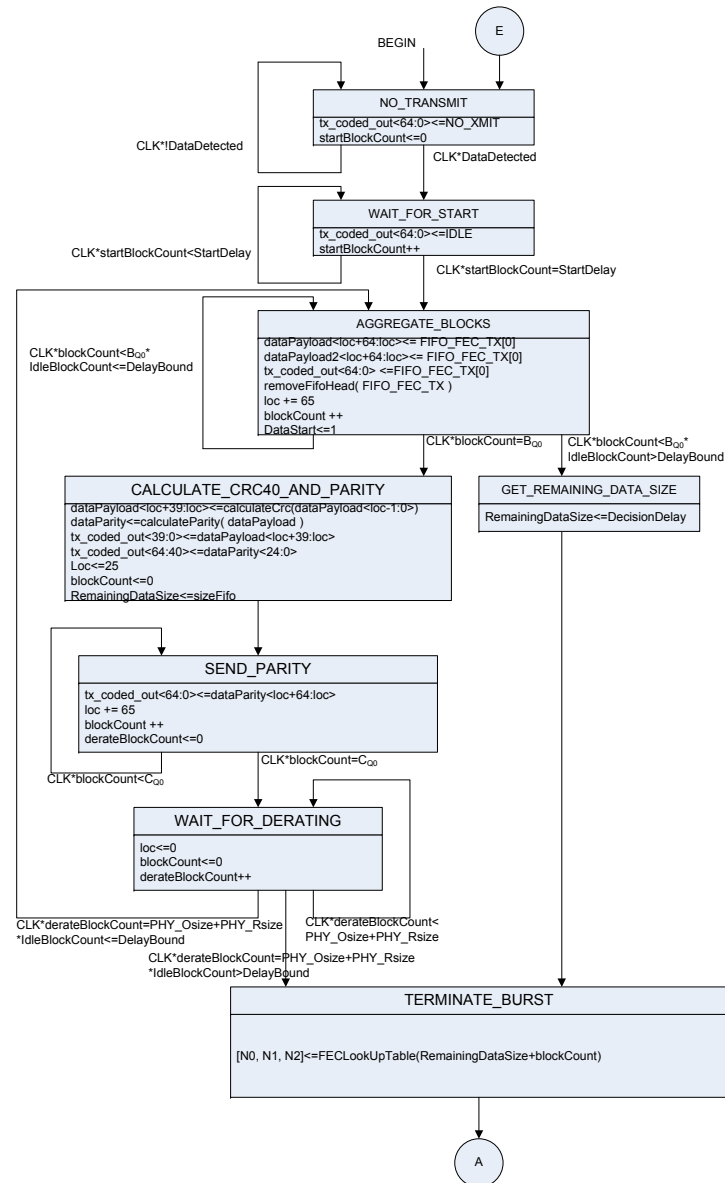
Option 1 – conventional streaming of codewords Info follow by Parity (and CRC)



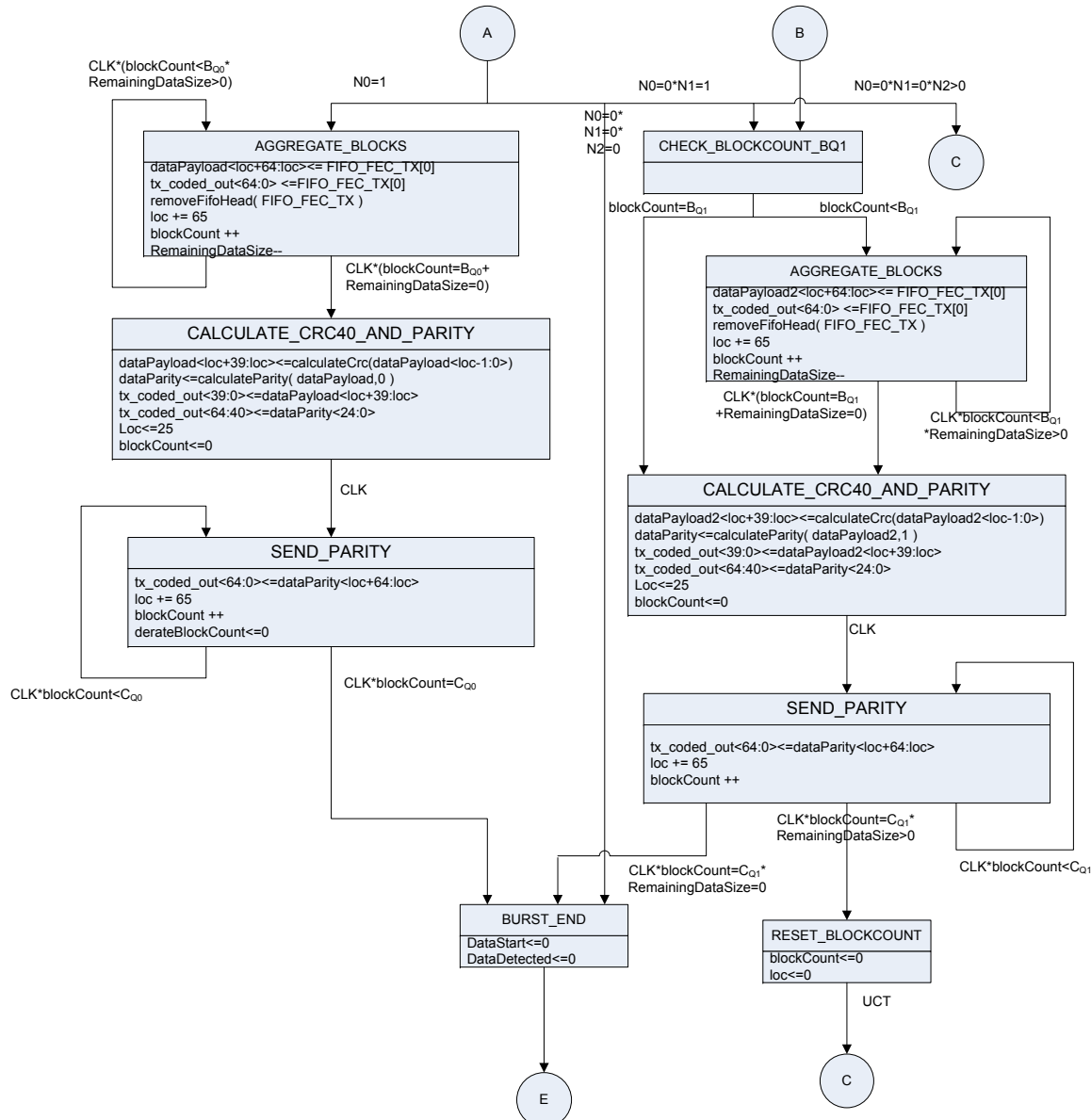
Option 2 – Parity follows info except of last 2-3 codewords
Info of all codewords transmitted following Parity and CRC of all codewords

- Option 1 – Conventional streaming of CWs info followed by parity, CW by CW
- Option 2 – Same as #1, info followed by parity except for last 2-3 CWs , all info is transmitted first, followed by parity of all three CWs

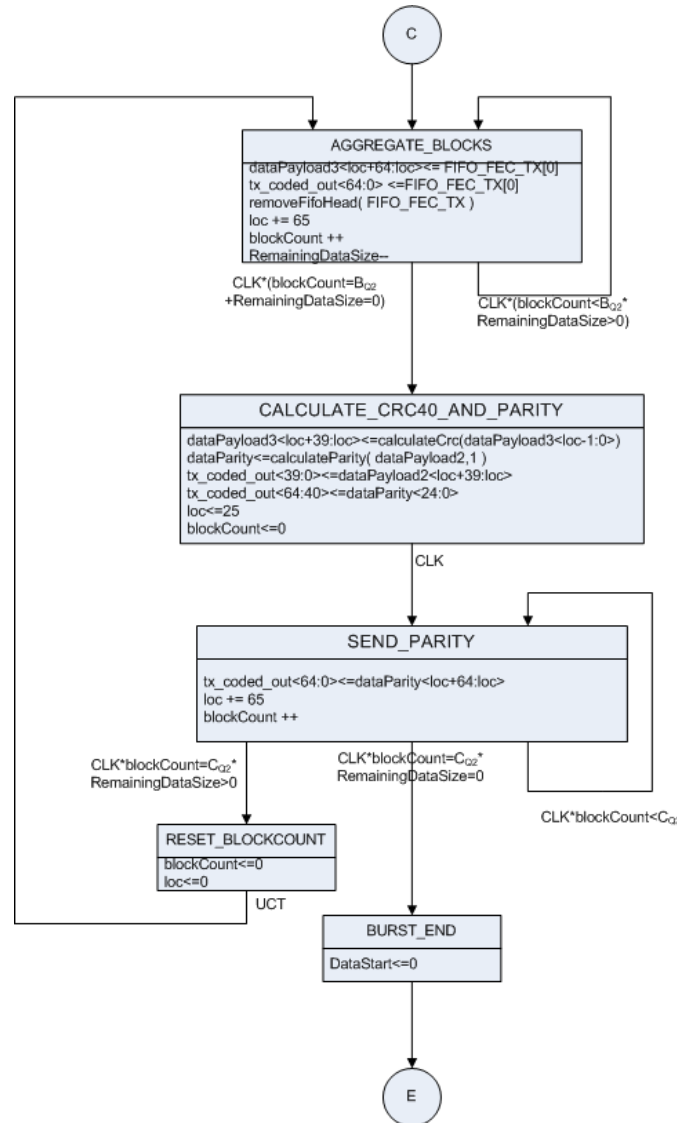
CNU Data Detector Diagram – I



CNU Data Detector Diagram – II



CNU Data Detector Diagram – III

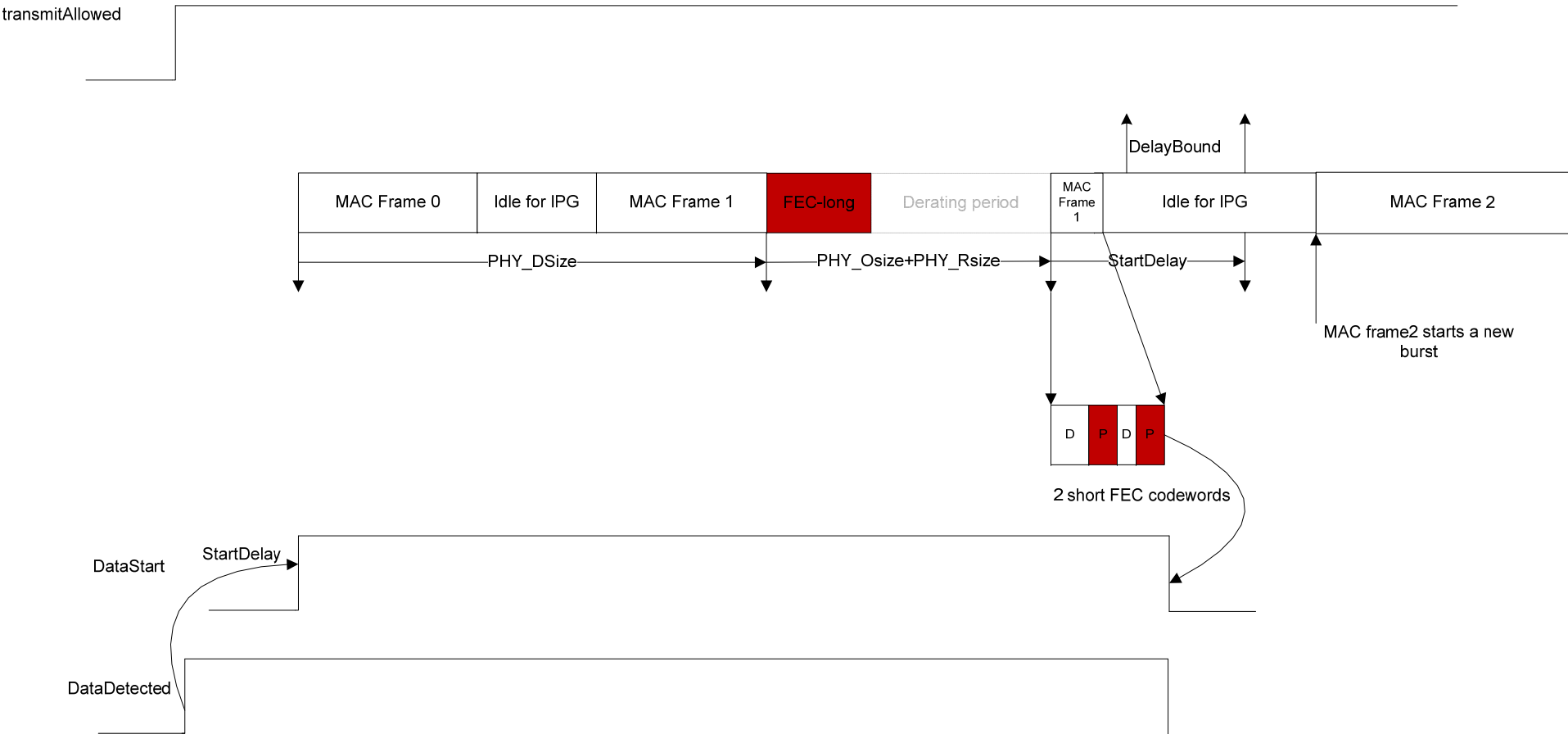


Parameters and Functions

- StartDelay: Initial delay to get FIFO filled.
StartDelay=DecisionDelay+DelayBound
- DecisionDelay: Additional delay for FEC encoder
- DelayBound: the threshold of number of consecutive idle vectors for terminating a burst.
- dataPayload: encoder memory buffer for long FEC code
- dataPayload2: encoder memory buffer for medium FEC code
- dataPayload3: encoder memory buffer for small FEC code.
- [N0, N1, N2]<=FECLookUpTable(size): return the number of long, medium, short codewords given the size of remaining data.
- PHY_Osize and PHY_Rsize: same as in idle deletion
- DataStart: set to 1 when data detector output vectors during a burst

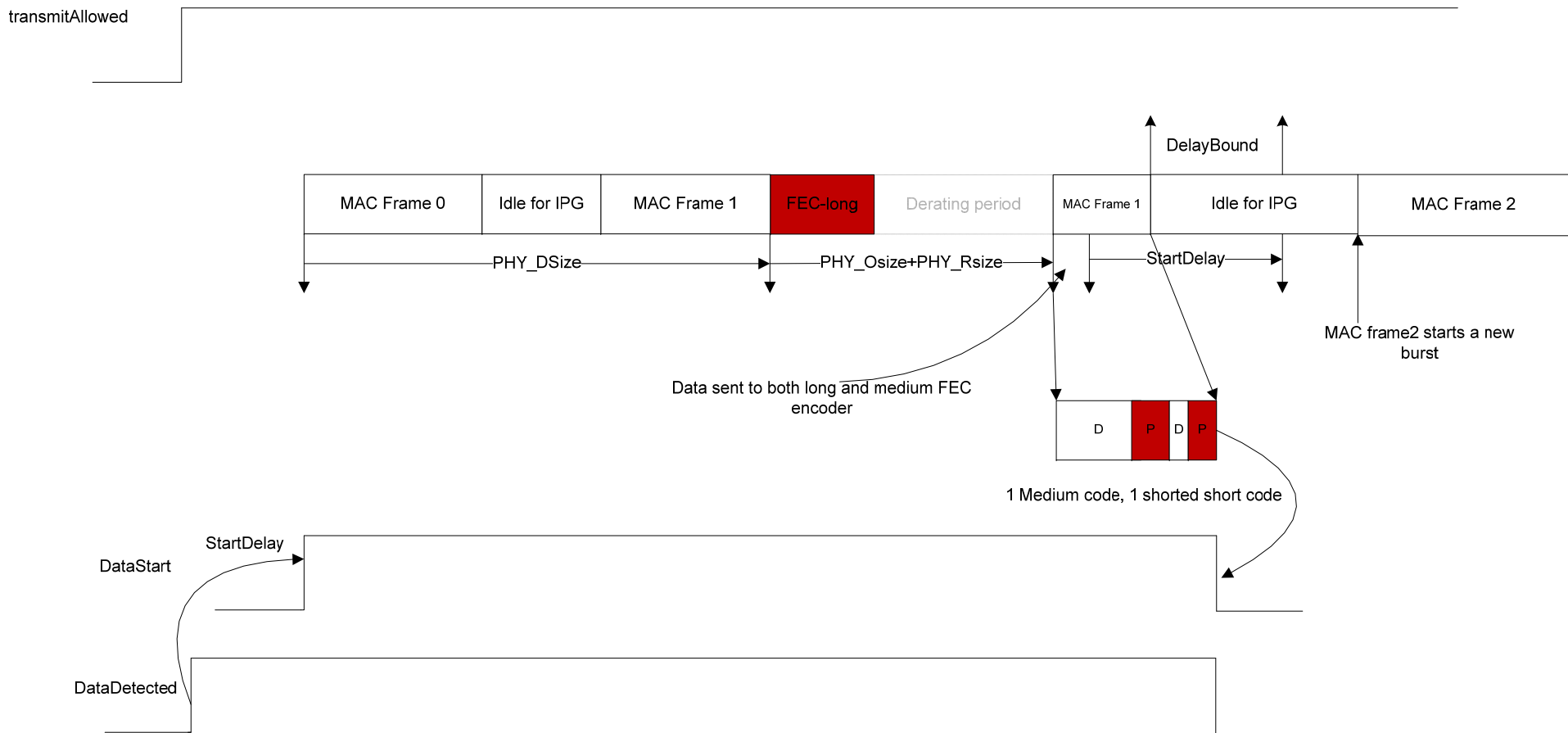
Example(1) (updated)

Case 1: remaining data vectors less than DecisionDelay



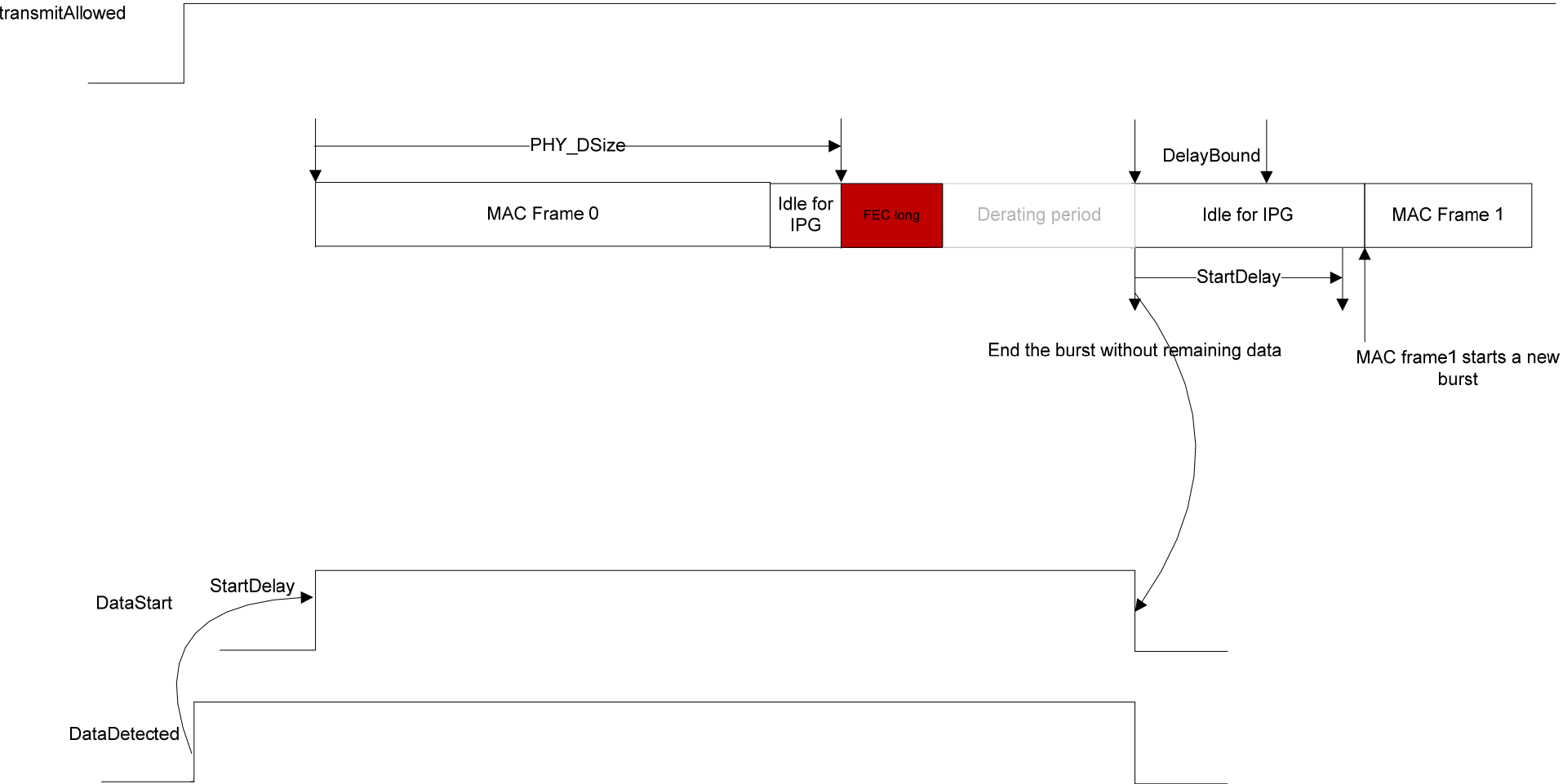
Example (2)(updated)

Case 2: remaining data vectors greater than DecisionDelay



Example (3)(updated)

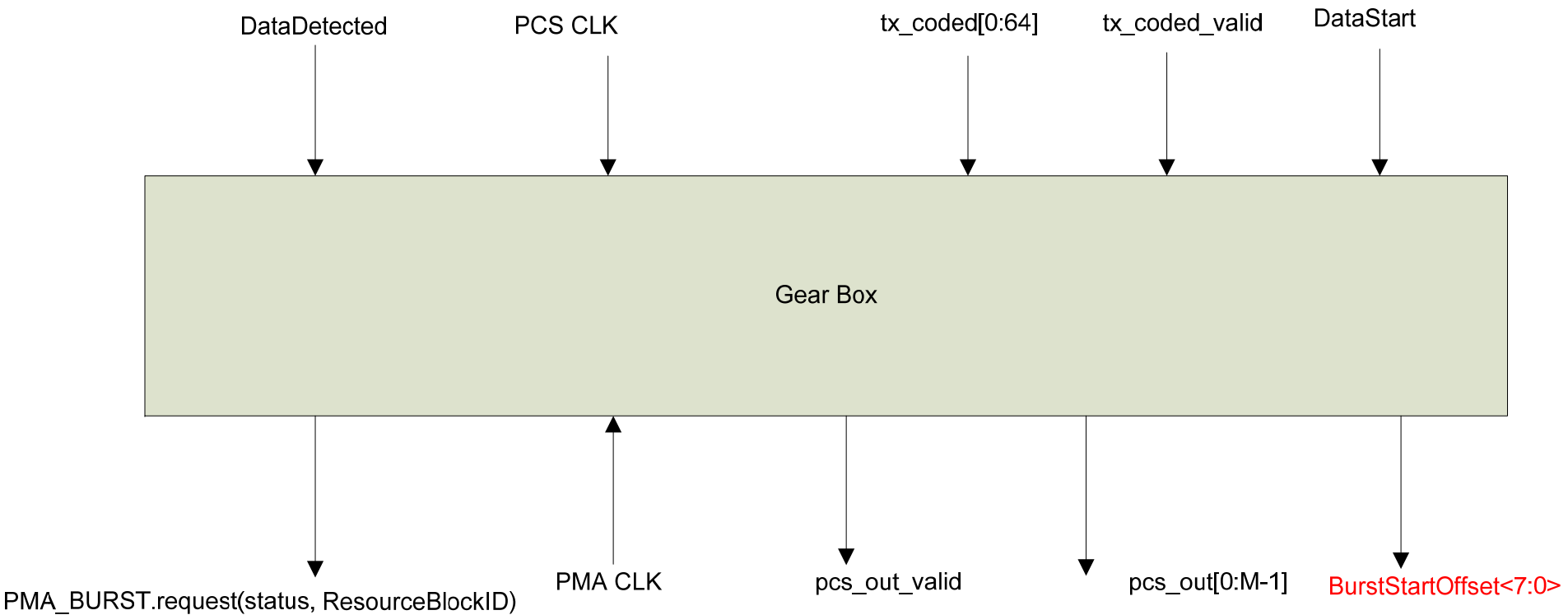
Case 3: No remaining data. The last long codeword just finishes the burst



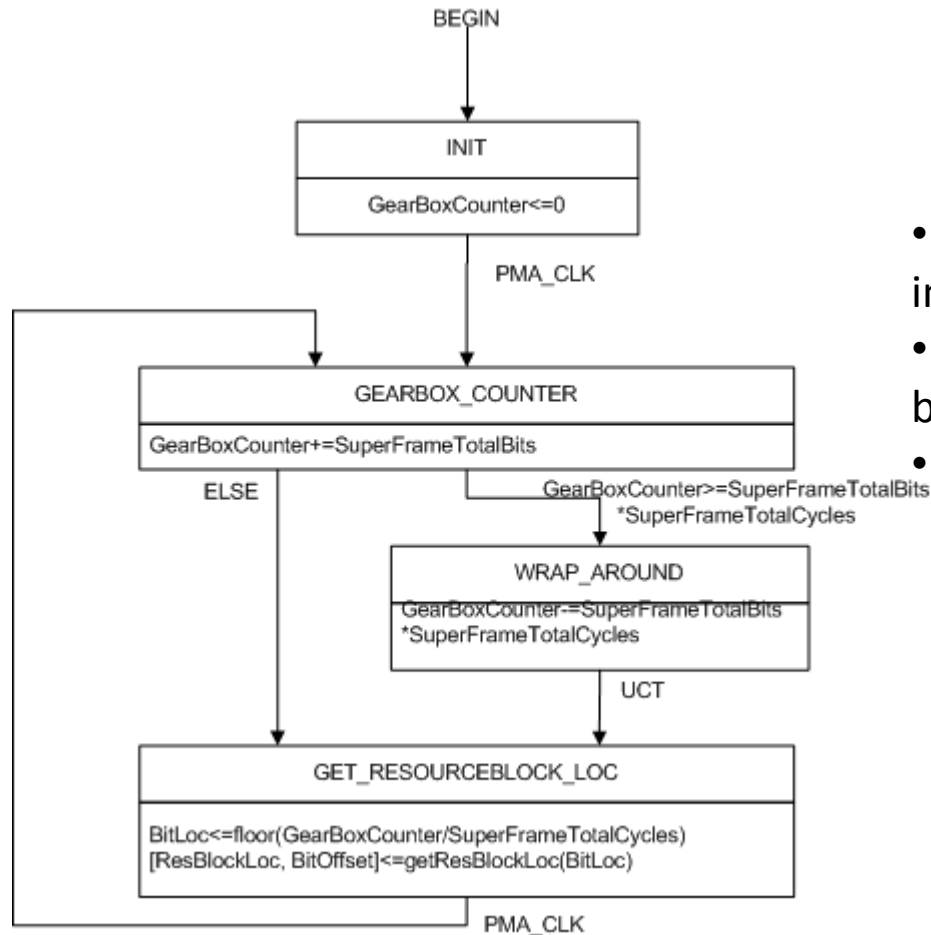
CNU Gear Box

- Unlike CLT, the CNU gearbox plays an important role in defining the 2-D burst.
- In all CNUs, the gearbox has a common counter that sweeps through a superframe at a constant bit rate, i.e. sweeps through `superFrameTotalBits` within `SuperFrameTotalCycles`.
- Upon receiving `DataDetected` from data detector, the gearbox locks in a snapshot of the common counter and passes down to PMA a primitive to indicate the first RB position of a burst
- Upon receiving `DataStart`, the gearbox input process starts buffering data detector output.
- Upon receiving zero for `DataStart`, the gearbox shall empty its buffer and notify PMA burst end using a primitive

CNU Gearbox Input/Output

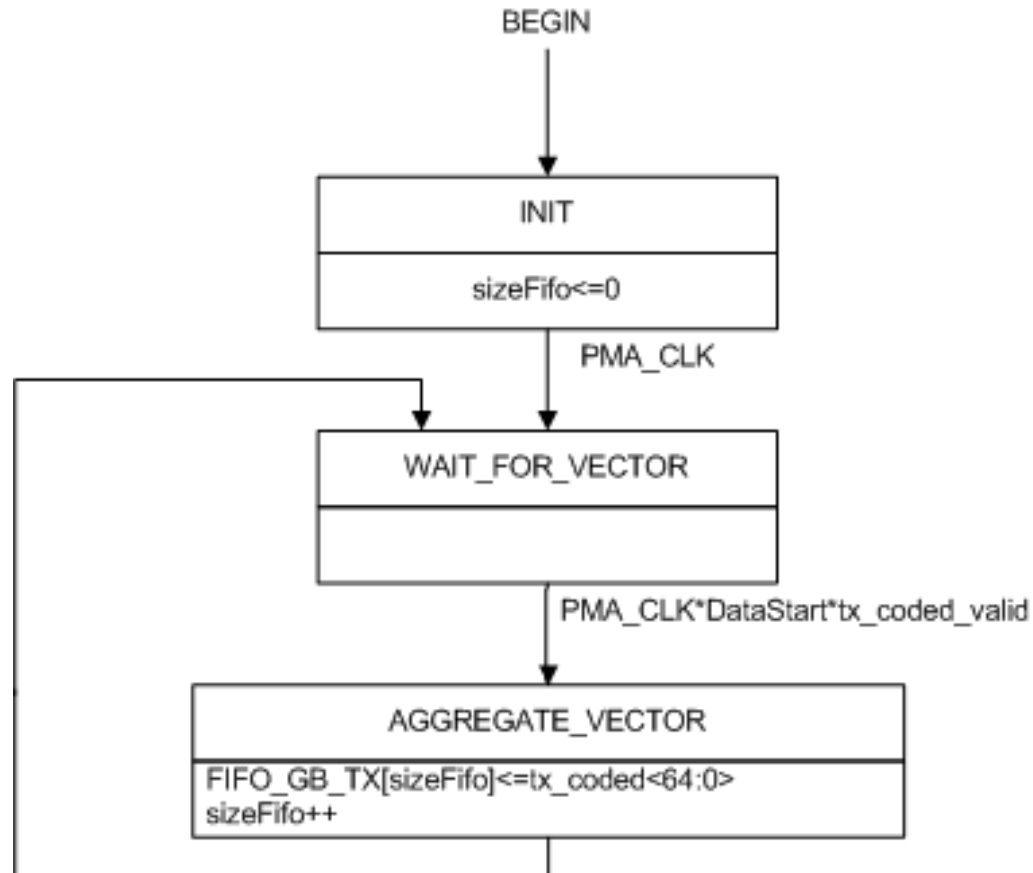


Gearbox Counter

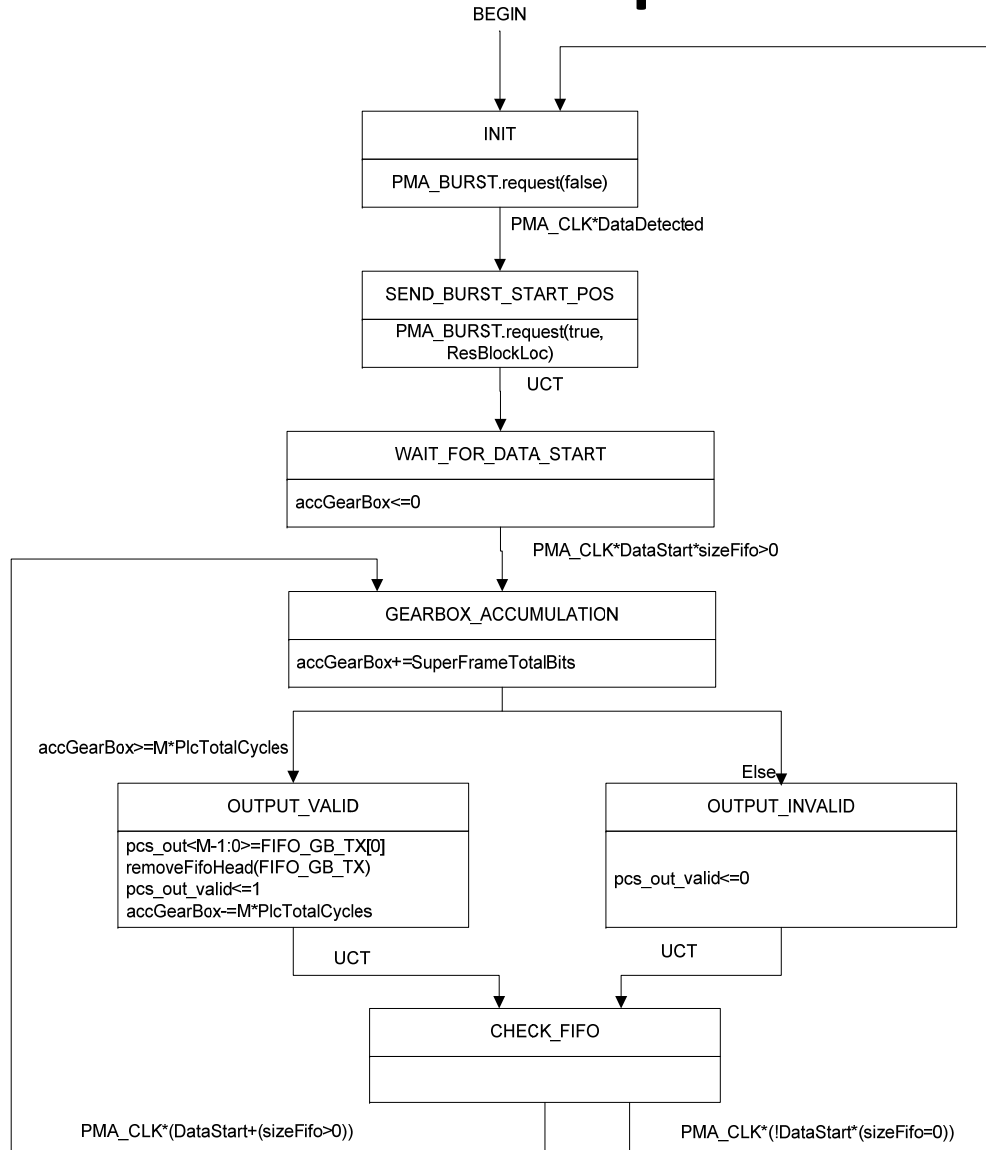


- ResBlockLoc: location of resource block in one superframe
- BitOffset: bit offset within a resource block
- BitLoc: location of the current bit.

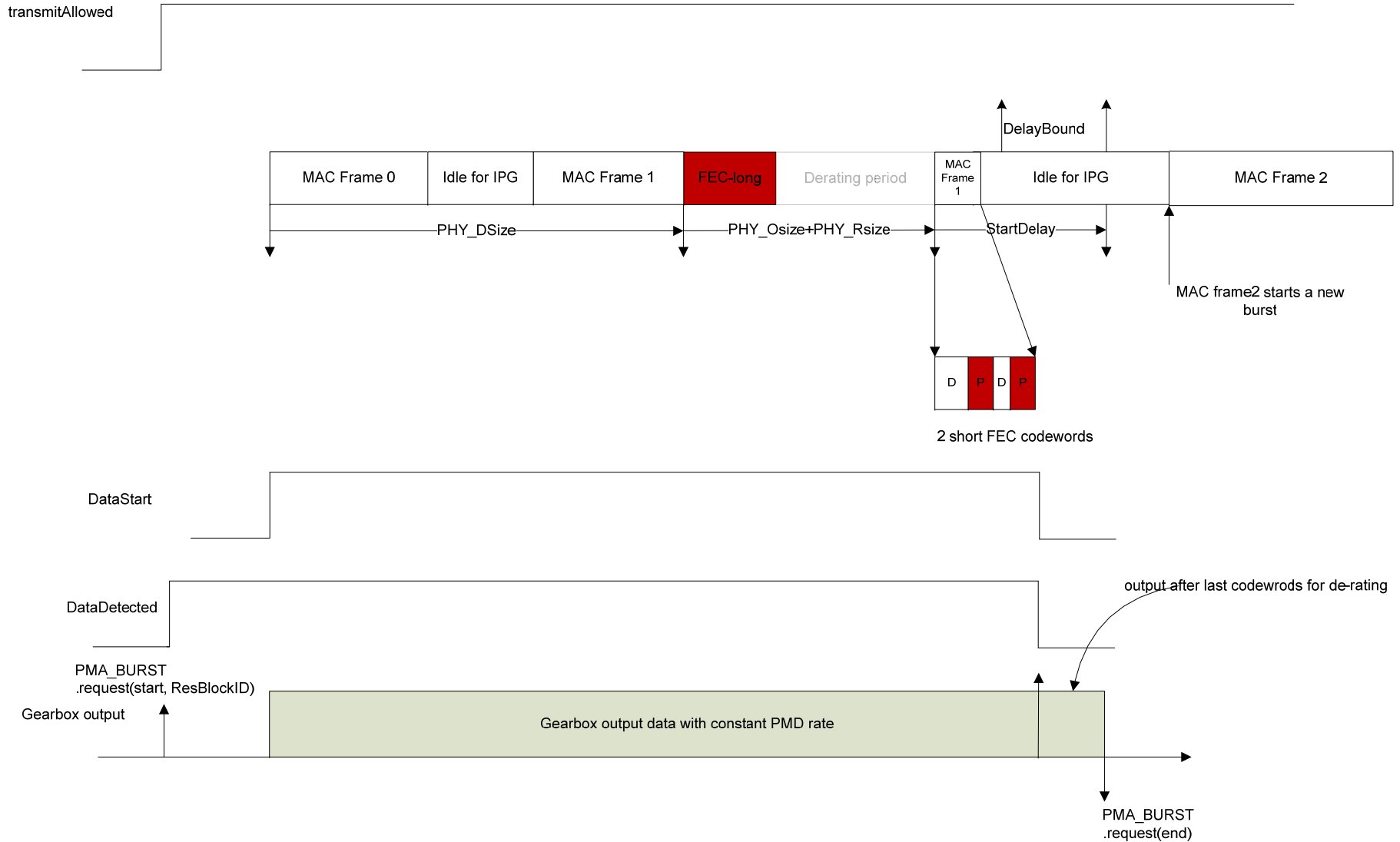
Gearbox Input Process



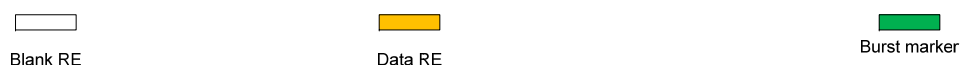
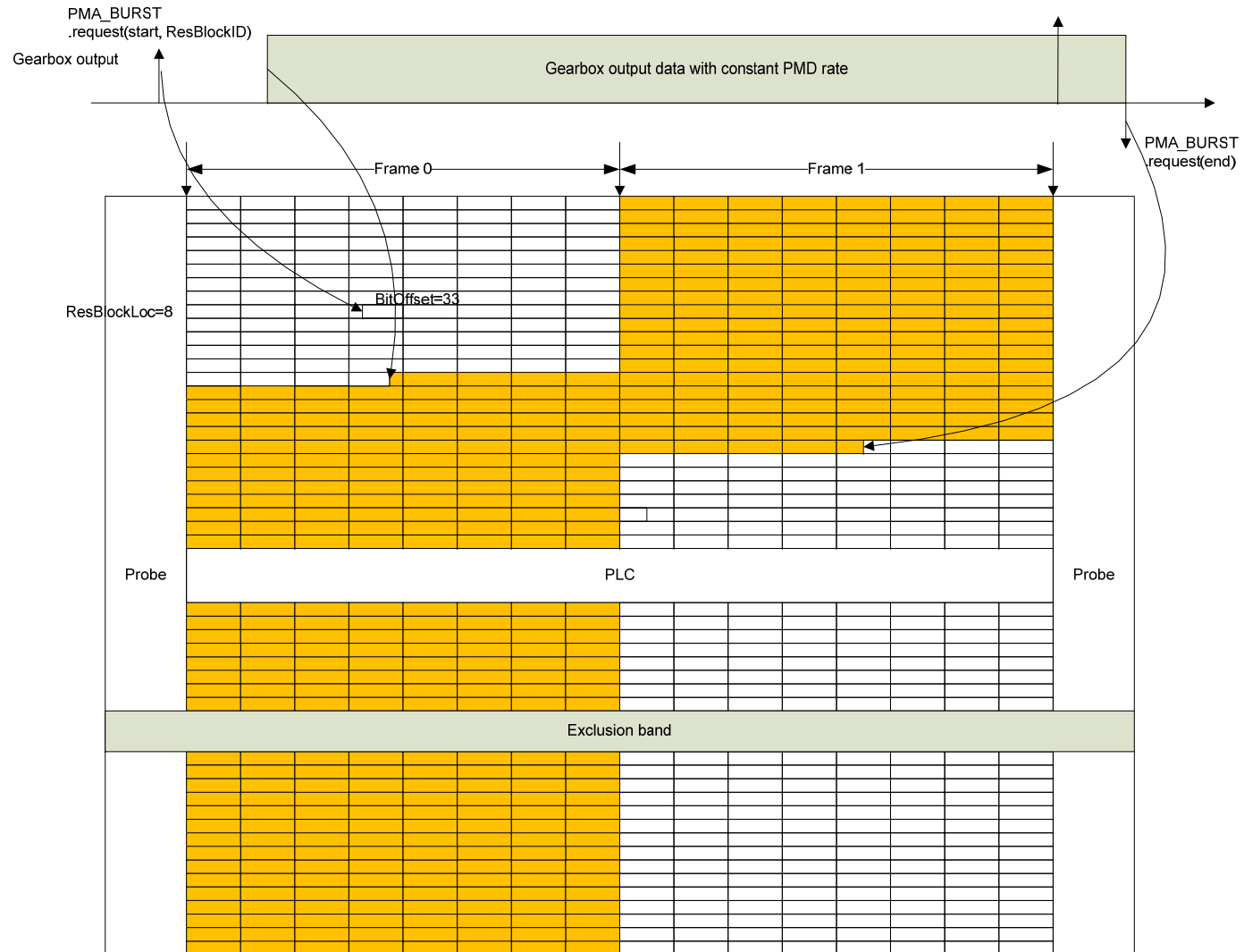
Gearbox Output Process



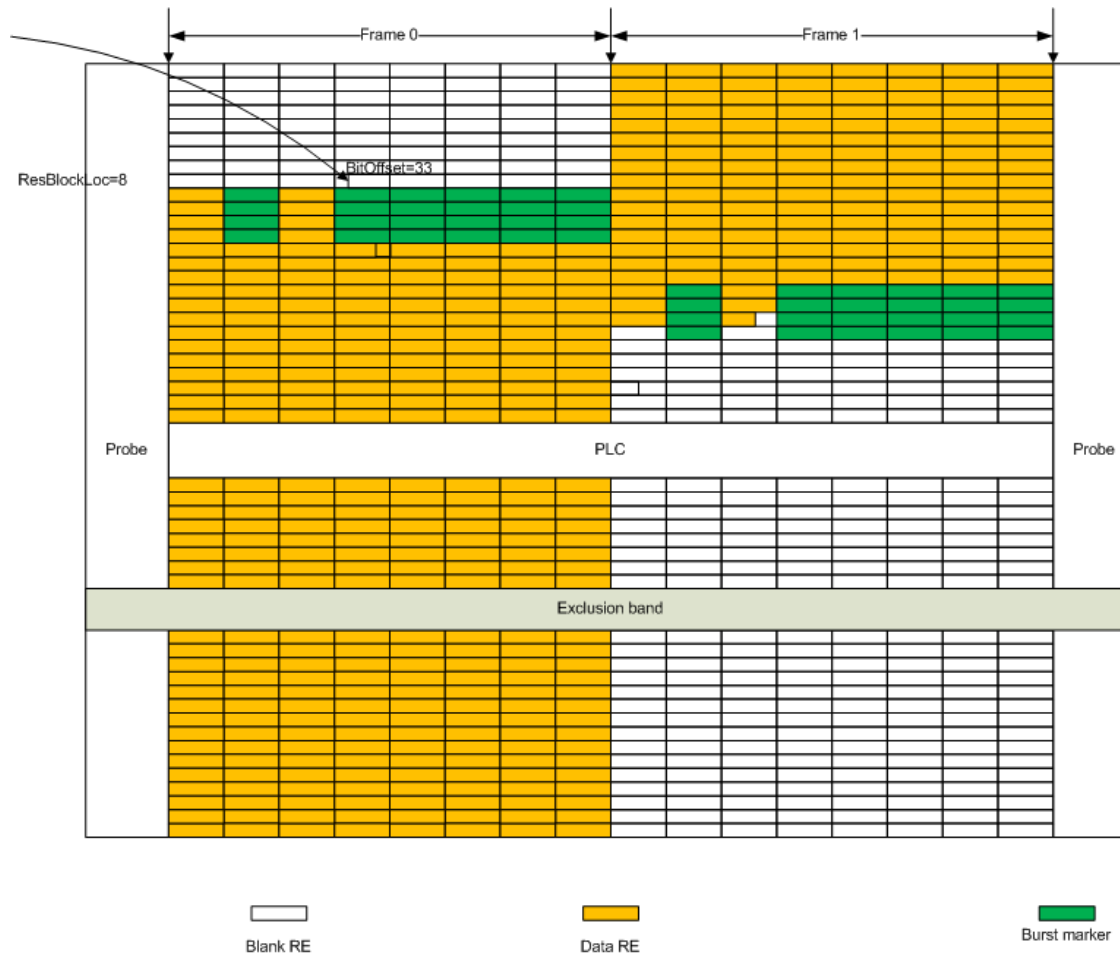
Gearbox Example



How Does PMA Map Data? (Update)



After Reposition at PMA Modulator



PMA Mapping Process

- Upon receiving PMA_BURST.request(start, ResBlockLoc), locate the first resource block, fill in start marker.
- Upon receiving burst data, place (after possible interleaving) the data at the first available position.
- Upon receiving PMA_BURST.request(end), find out the position of end marker with minimum burst end offset.

Thank you
Q&A