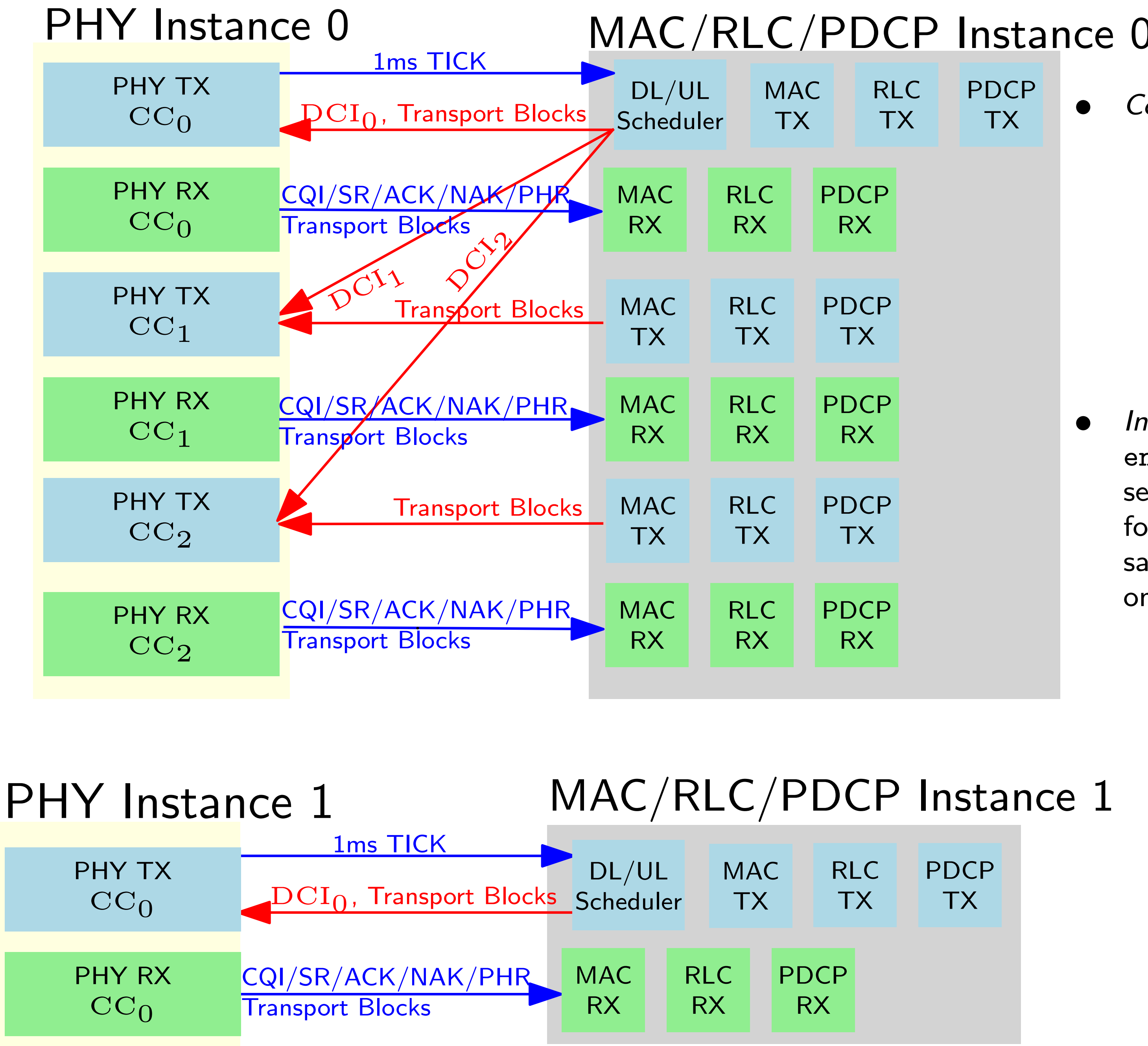
Some Preliminaries

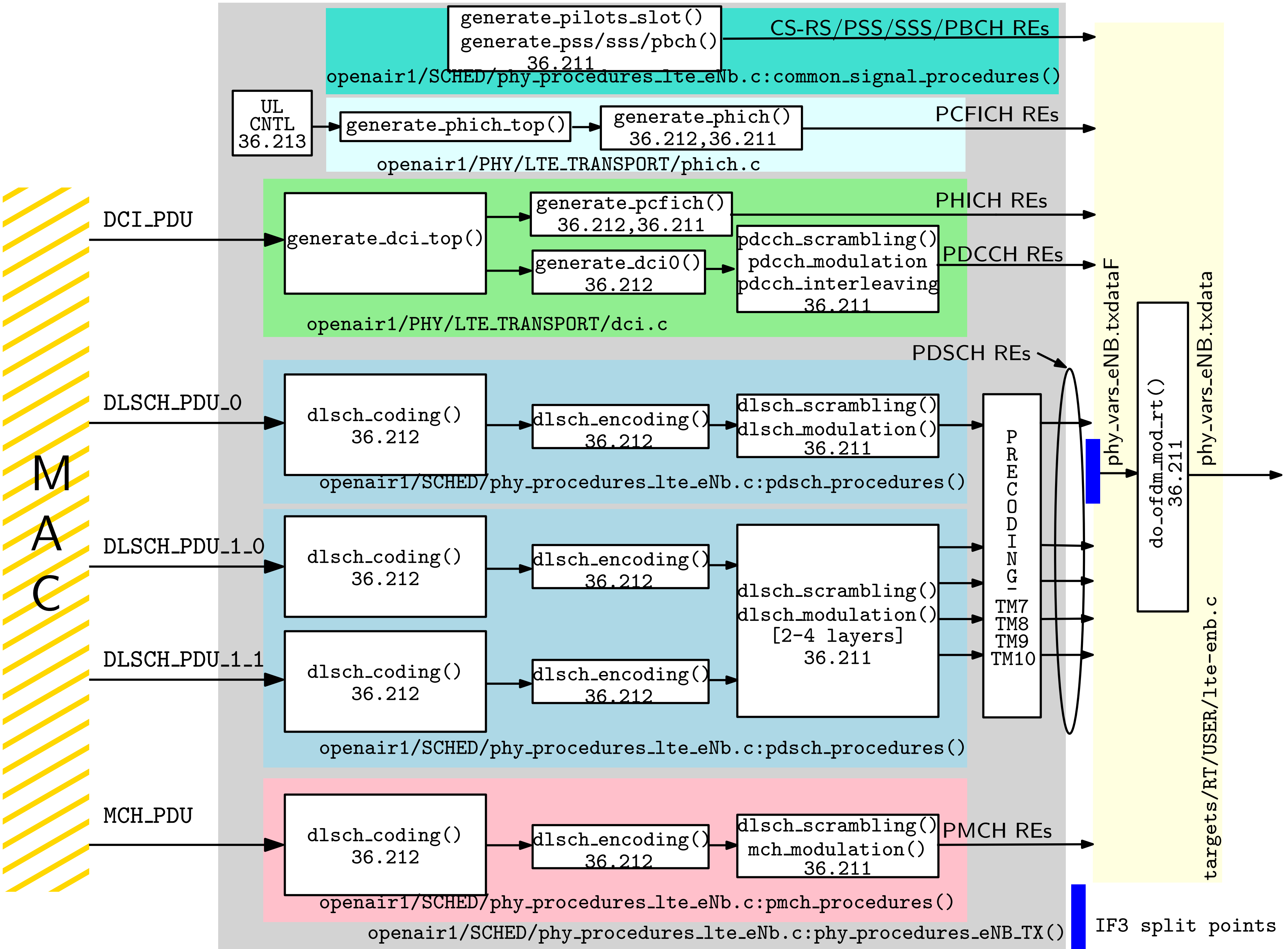
- This description corresponds to the enhancement-10-harmony branch
- **purpose 1**: clean up interfaces within L1. L1/L2 are being done by others in the community (e.g. Bell Labs)
- **purpose 2**: harmonize HW plaforms (hence the name)
- **purpose 3**: make the interfaces flexible for functional splits between RU/SU and local radio gateways
- **purpose 4**: harmonize lte-softmodem (lte-ue/lte-enb) and oaisim

eNodeB Threads, Instances and Component Carriers

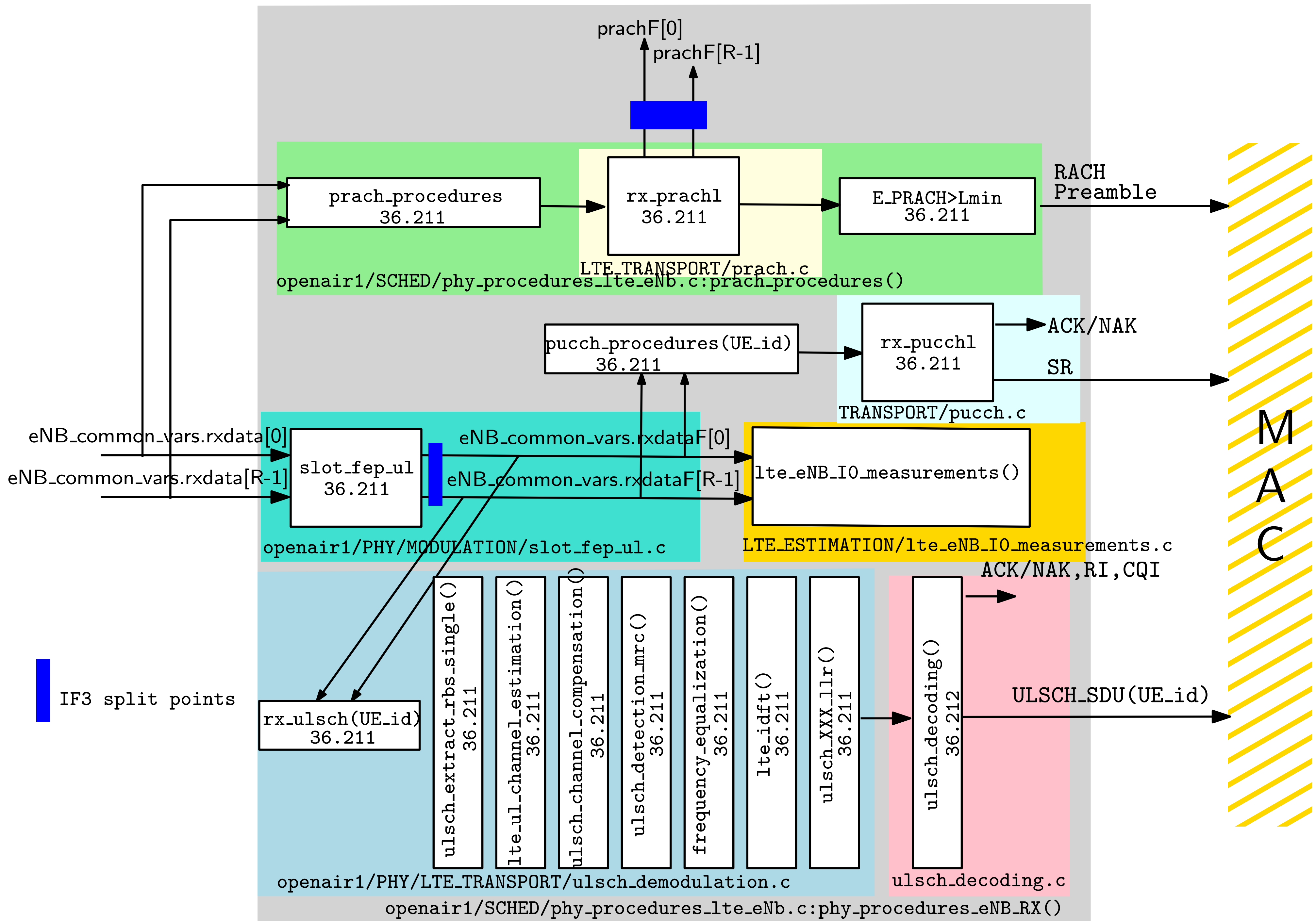


- *Component Carrier* is
  - Rel10+ component carrier
  - distributed antenna (DAS) on common carrier to others (indoor, handover-less deployment, CoMP, etc.)
- *Instance* (`Mod_id`, or `enb_mod_id`) is a separate set of threads and contexts for a full eNodeB in the same Linux process. It is only used by `oaisim` now.

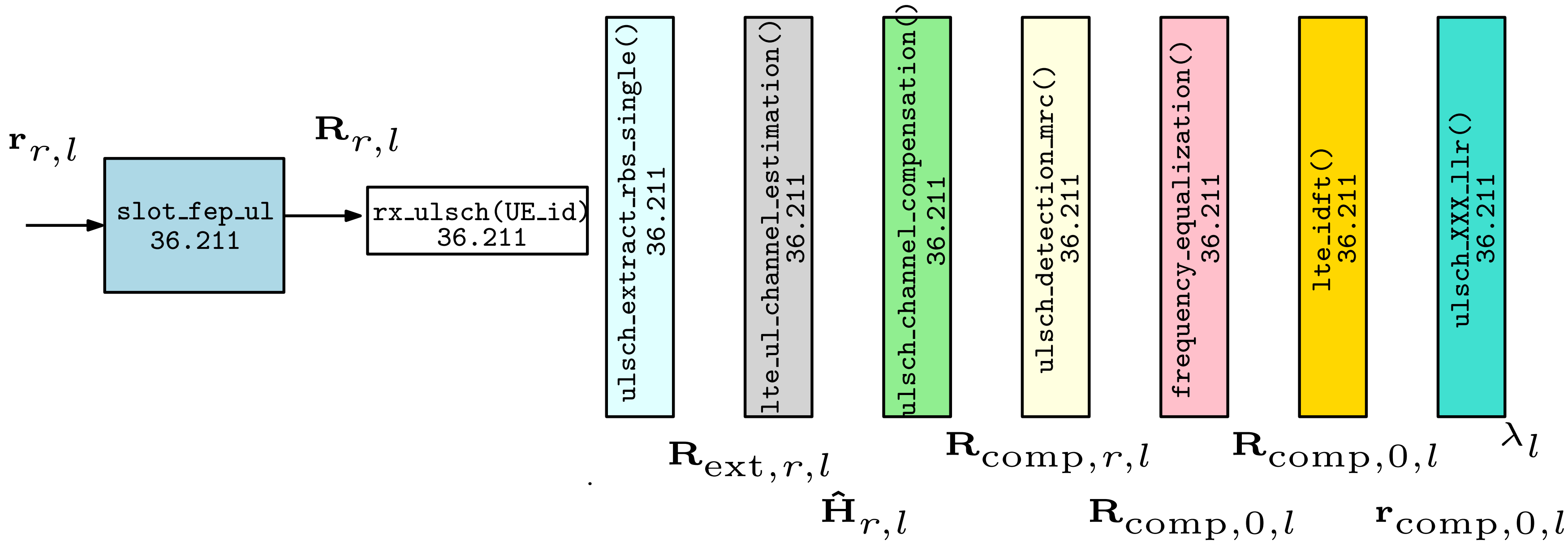
eNodeB PHY TX Procedures



eNodeB PHY RX Procedures



eNodeB ULSCH Demodulation



$\mathbf{R}_{r,l} = \mathrm{DFT}_{N_{\mathrm{fft}}}(\mathbf{r}_{r,l} \odot \mathbf{F}_{7.5}), r = 0, 1, \cdots, R-1, l = 0, 1, \cdots, N_{\mathrm{symb}} - 1$ **(eNB_common_vars→rxdataF[][])**

$R_{\mathrm{ext},r,l}(n) = R_{r,l}(12\mathrm{firstPRB} + n), n = 0, 1, \cdots, 12N_{\mathrm{PRB}} - 1$ **(eNB_pusch_vars→ulsch_rxdataF_ext[][])**

$\hat{\mathbf{H}}_{r,l} = \mathbf{R}_{\mathrm{ext},r,l} \odot \mathbf{DRS}_l^*(\mathrm{cyclicShift}, n_{\mathrm{DMRS}(2)}, n_{\mathrm{PRS}})$, **(eNB_pusch_vars→drs_ch_estimates[])**

$\mathbf{R}_{\mathrm{comp},r,l} = \hat{\mathbf{H}}_r \odot \mathbf{R}_{\mathrm{ext},r,l} 2^{-\log_2|H_{\max}|}, \hat{\mathbf{H}}_r = \frac{1}{2}(\hat{\mathbf{H}}_{r,3} + \hat{\mathbf{H}}_{r,10})$
(eNB_pusch_vars→ulsch_rxdataF_comp)

$\mathbf{R}_{\mathrm{comp},0,l} = \frac{1}{R}\sum_{r=0}^{R-1} \mathbf{R}_{\mathrm{comp},r,l}$

$R_{\mathrm{comp},0,l}(n) = R_{\mathrm{comp},0,l}(n)\dot{Q}_8\left(\frac{1}{|\hat{H}(n)|^2 + I_0}\right), \hat{H}(n) = \sum_{r=0}^{R-1} \hat{H}_r(n)$

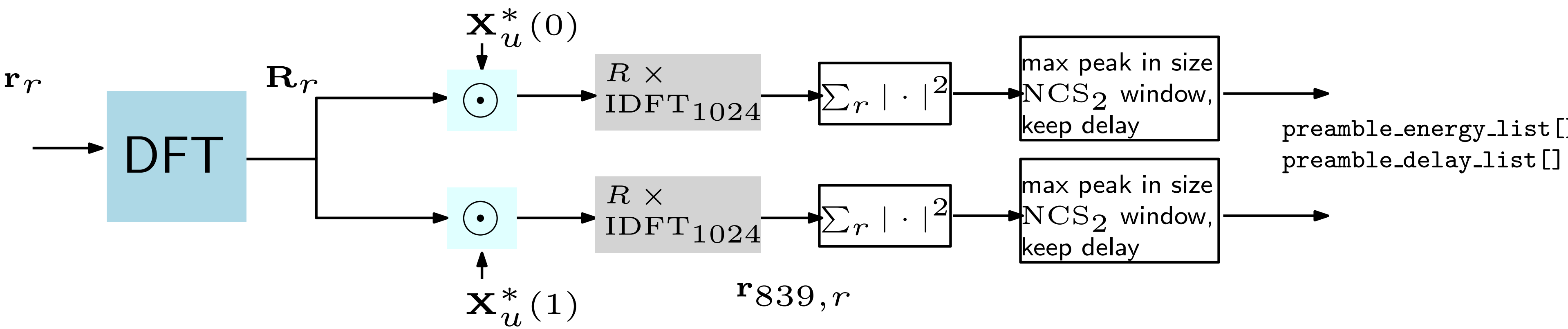
$\mathbf{r}_{\mathrm{comp},0,l} = \mathrm{DFT}_{12N_{\mathrm{PRB}}}(\mathbf{R}_{\mathrm{comp},0,l})$

QPSK : $\lambda_l(2n) = \mathrm{Re}(r_{\mathrm{comp},0,l}(n)), \lambda_l(2n+1) = \mathrm{Im}(r_{\mathrm{comp},0,l}(n))$ **(eNB_pusch_vars→ulsch_llr)**

16QAM : $\lambda_l(4n) = \mathrm{Re}(r_{\mathrm{comp},0,l}(n)), \lambda_l(4n+2) = \mathrm{Im}(r_{\mathrm{comp},0,l}(n))$

$\lambda_l(4n+1) = |\mathrm{Re}(r_{\mathrm{comp},0,l}(n))| - 2\overline{|h(n)|}, \lambda_l(4n+3) = |\mathrm{Im}(r_{\mathrm{comp},0,l}(n))| - 2\overline{|h(n)|}$

eNodeB PRACH Detection



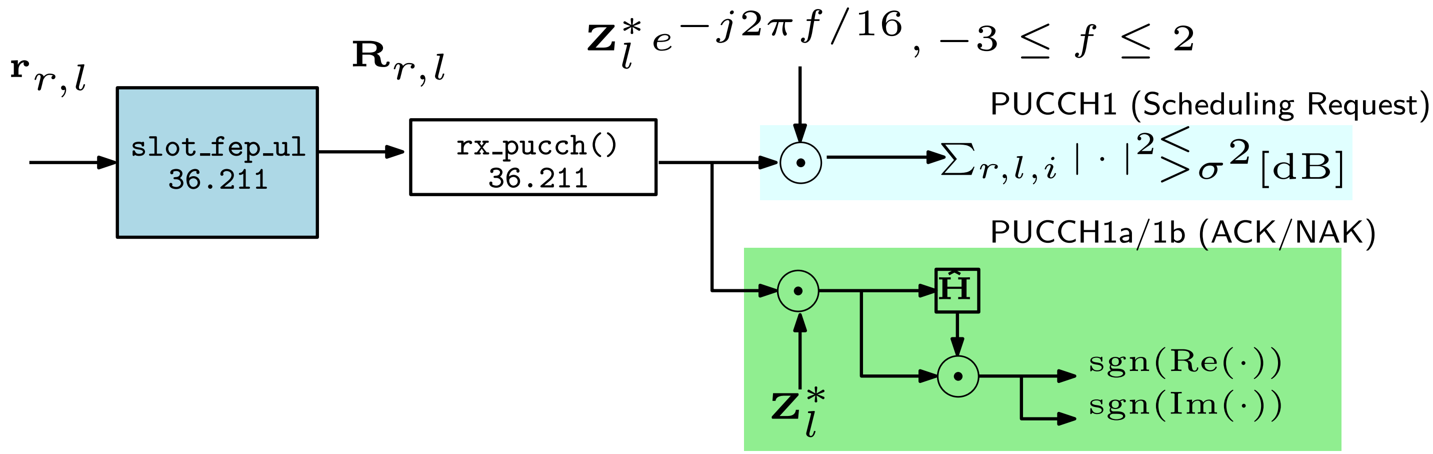
$\mathbf{R}_r = \mathrm{DFT}_{N_{\mathrm{PRACH}}}(\mathbf{r}_r),\ r = 0, 1, \cdots, R-1$ **(lte_eNB_prach_vars→rxsigF[])**

$\mathbf{R}_{\mathrm{comp},r} = \mathbf{R}_r \odot \mathbf{X}_u^*[i],\ r = 0, 1, \cdots, R-1$ **(lte_eNB_prach_vars→prachF[])**

$\mathbf{r}_{839,r} = \mathrm{IDFT}_{1024}\left(\mathbf{R}_{\mathrm{comp},r}\right),\ r = 0, 1, \cdots, R-1$ **(lte_eNB_prach_vars→prach_ifft[])**

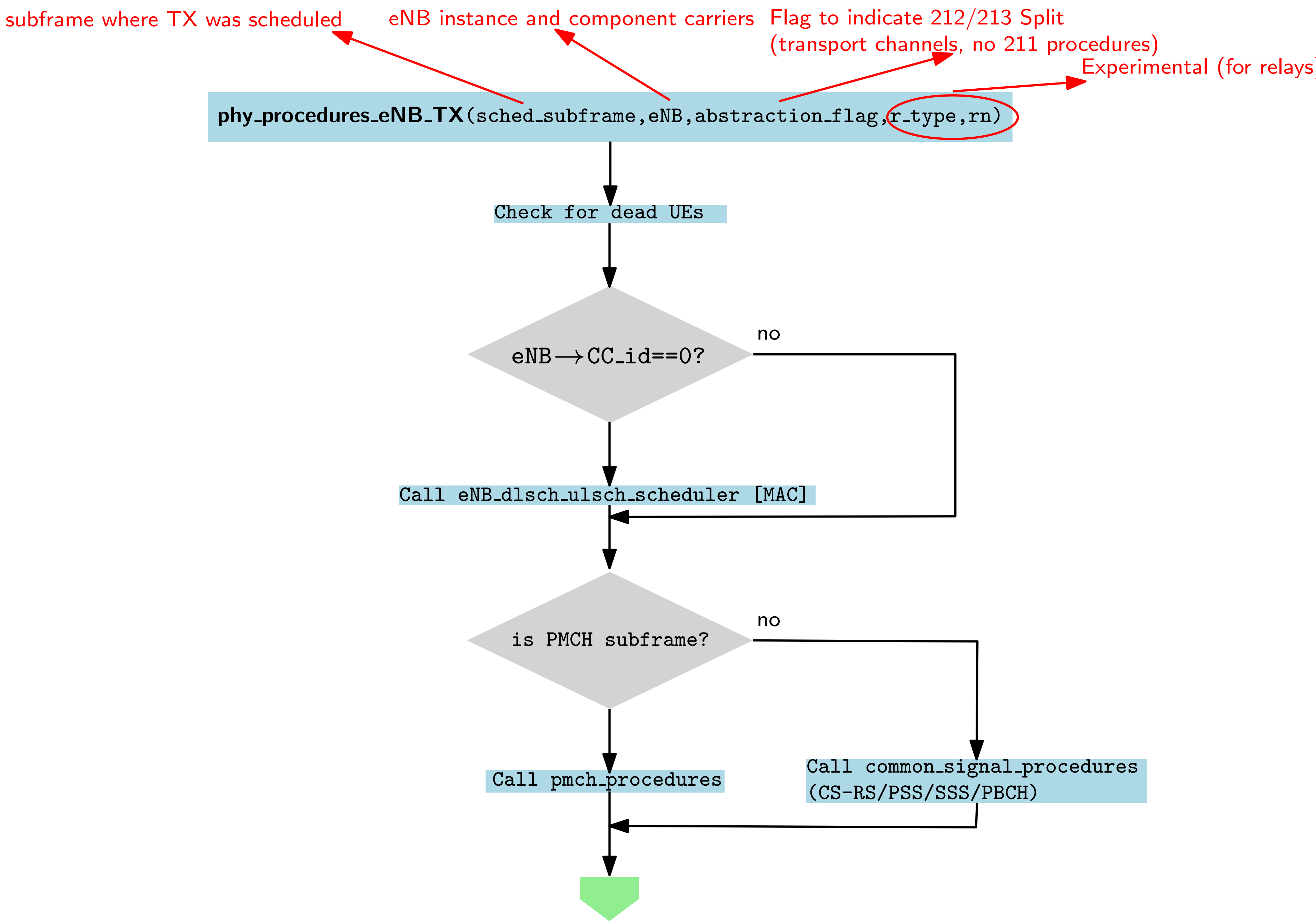
- PRACH detection is a quasi-optimal non-coherent receiver for vector observations (multiple antennas)
- correlation is done in the frequency-domain, number of correlations (in the example above 2) depends on *zeroCorrelationConfig* configuration parameter
- peak-detection (for delay estimation) is performed in each NCS time-window

eNodeB PUCCH Detection

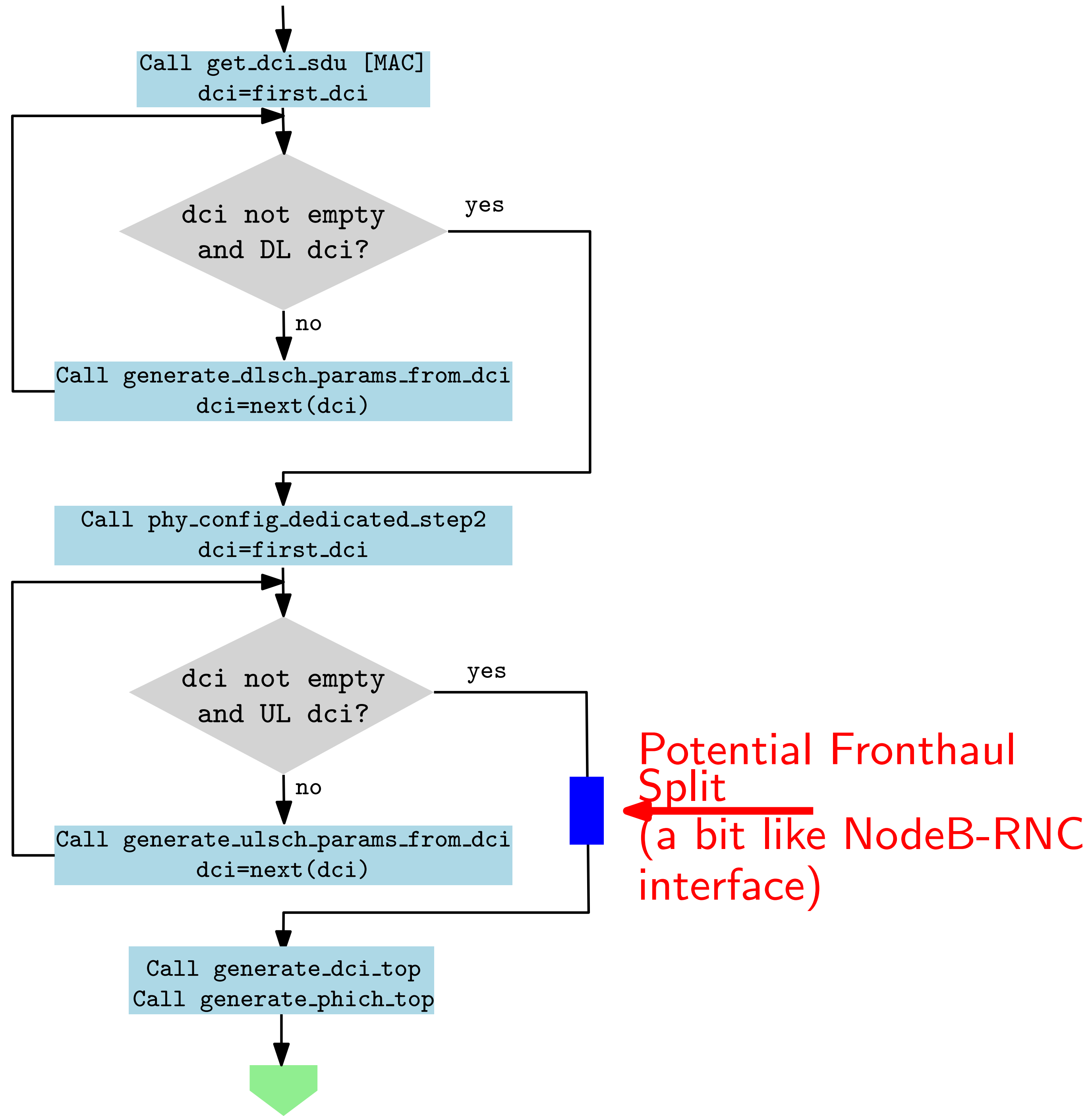


- PUCCH1 detection is a quasi-optimal non-coherent receiver (energy detector) for vector observations (multiple antennas) for scheduling request. Care is taken to handle residual frequency-offset.
- PUCCH1A/1B detection is quasi-coherent based on a rough channel estimate obtained on the 3 symbols without data modulation.
- In both cases, correlation is done in the frequency-domain

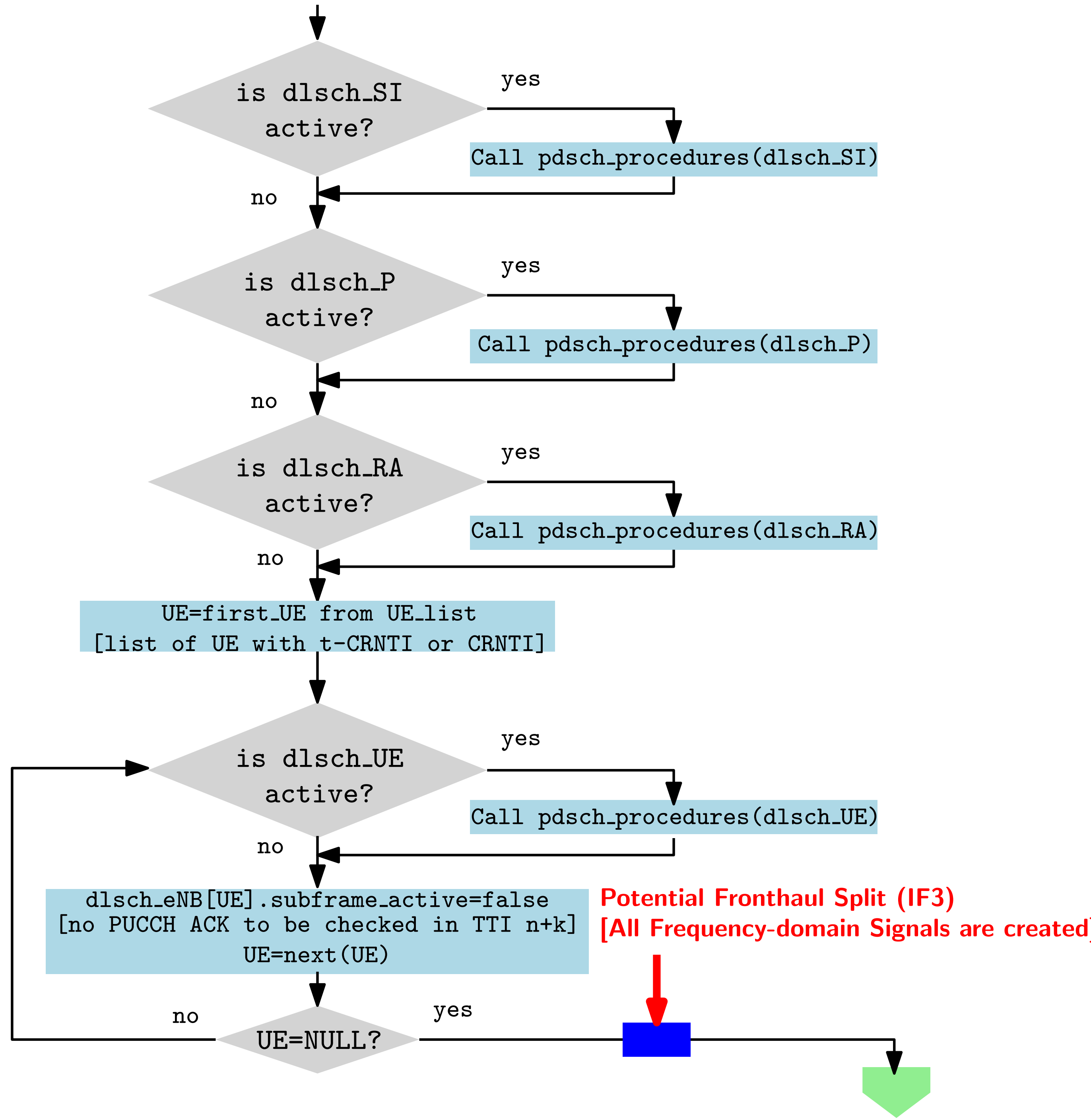
eNodeB TX Flowchart



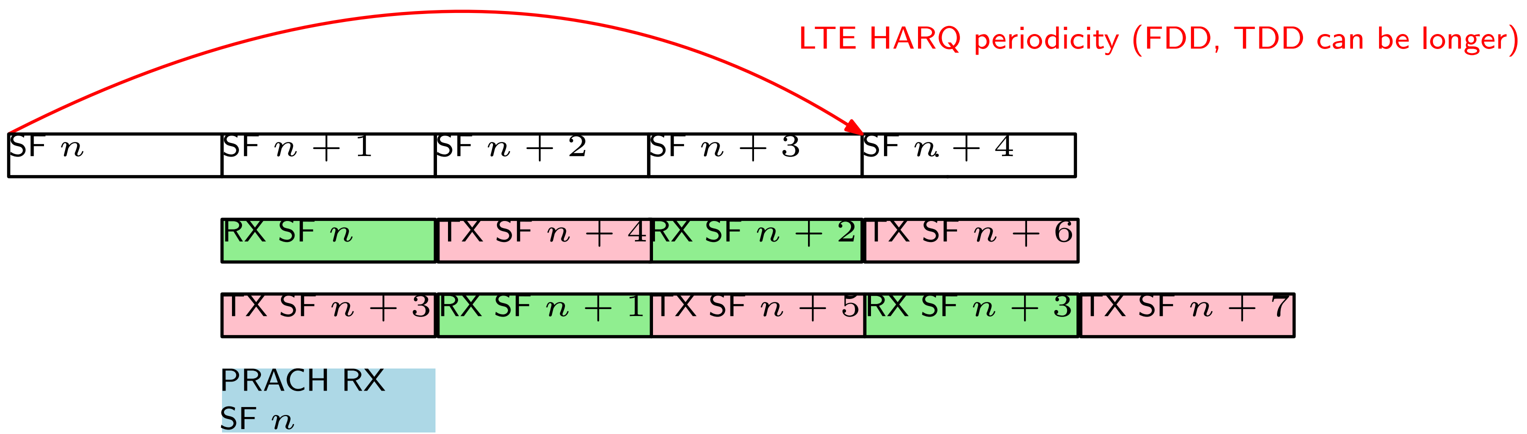
eNodeB TX Flowchart



eNodeB TX Flowchart



eNodeB Timing



- The current scheduling / timing requires approximately 1ms processing limit (basically 1 core RX, 1core TX). The current architecture will work on a single core if the RX and TX procedures can fit in the time of a single subframe.
- three threads, TX, RX and PRACH. RX for n processes the subframe and that wakes TX for $n+4$ in sequence (data dependency) and then waits for signal in $n+1$. If a PRACH process is to be launched in n the RX thread wakes the PRACH thread.
- TX for $n+4$ will exit the PC somewhere in first-half of SF n+3

eNodeB Timing

RX processing subframe n (eNB_thread_rx)

- trx_read(1 ms)
- slot_fep for slots $2n$ and $2n+1$ [phy_procedures_eNB_common_rx()]
- launch PRACH RX thread (subframe n)
  - prach_procedures
- phy_procedures_eNB_rx() (subframe n)

TX processing subframe $n+4$ (eNB_thread_tx)

- phy_procedures_eNB_tx
- ofdm_mod
- trx_write(1 ms)