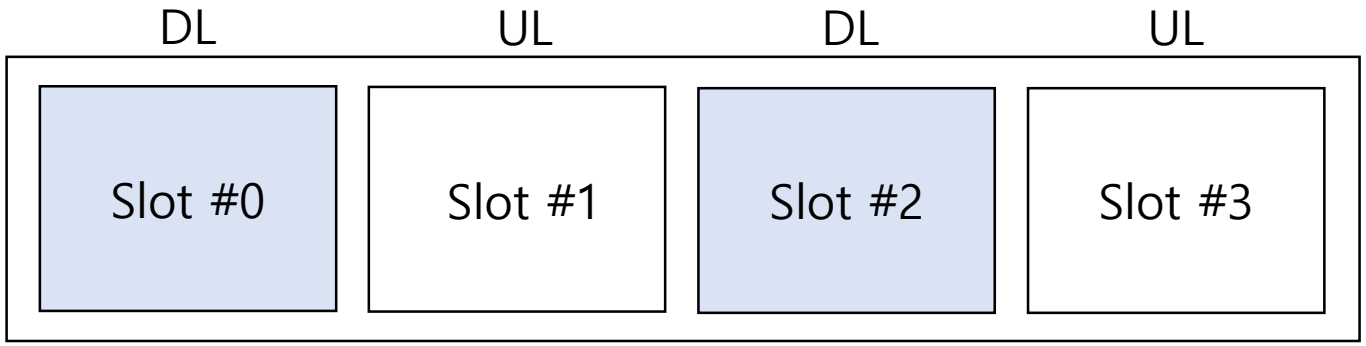
STLC 송수신을 위한
전체 framework 및 송수신 블록 설계

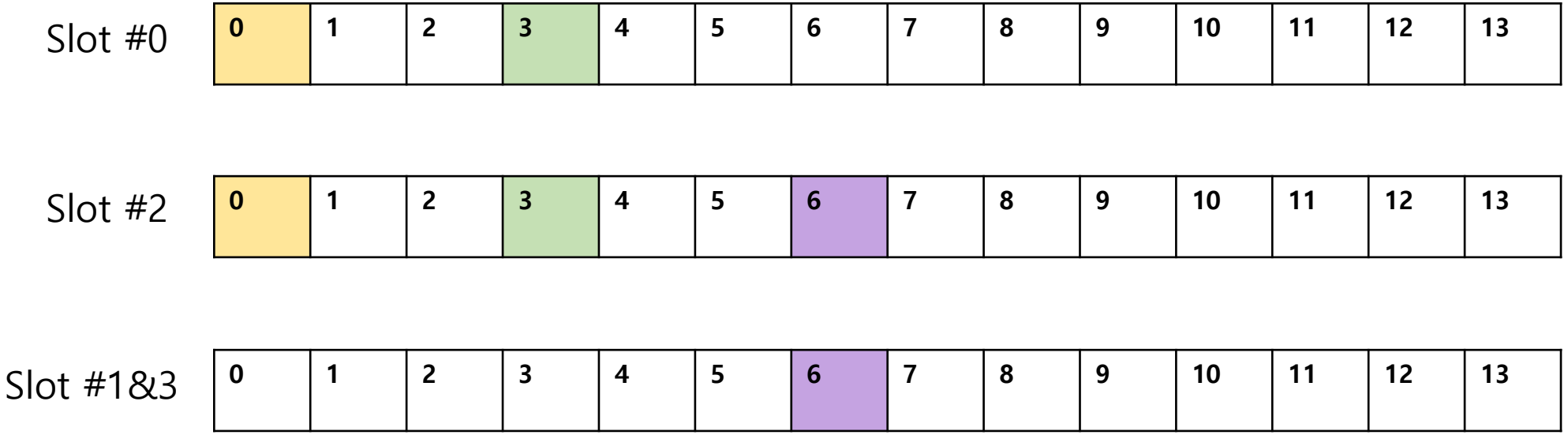
발표자 : 유제인

CONTENTS

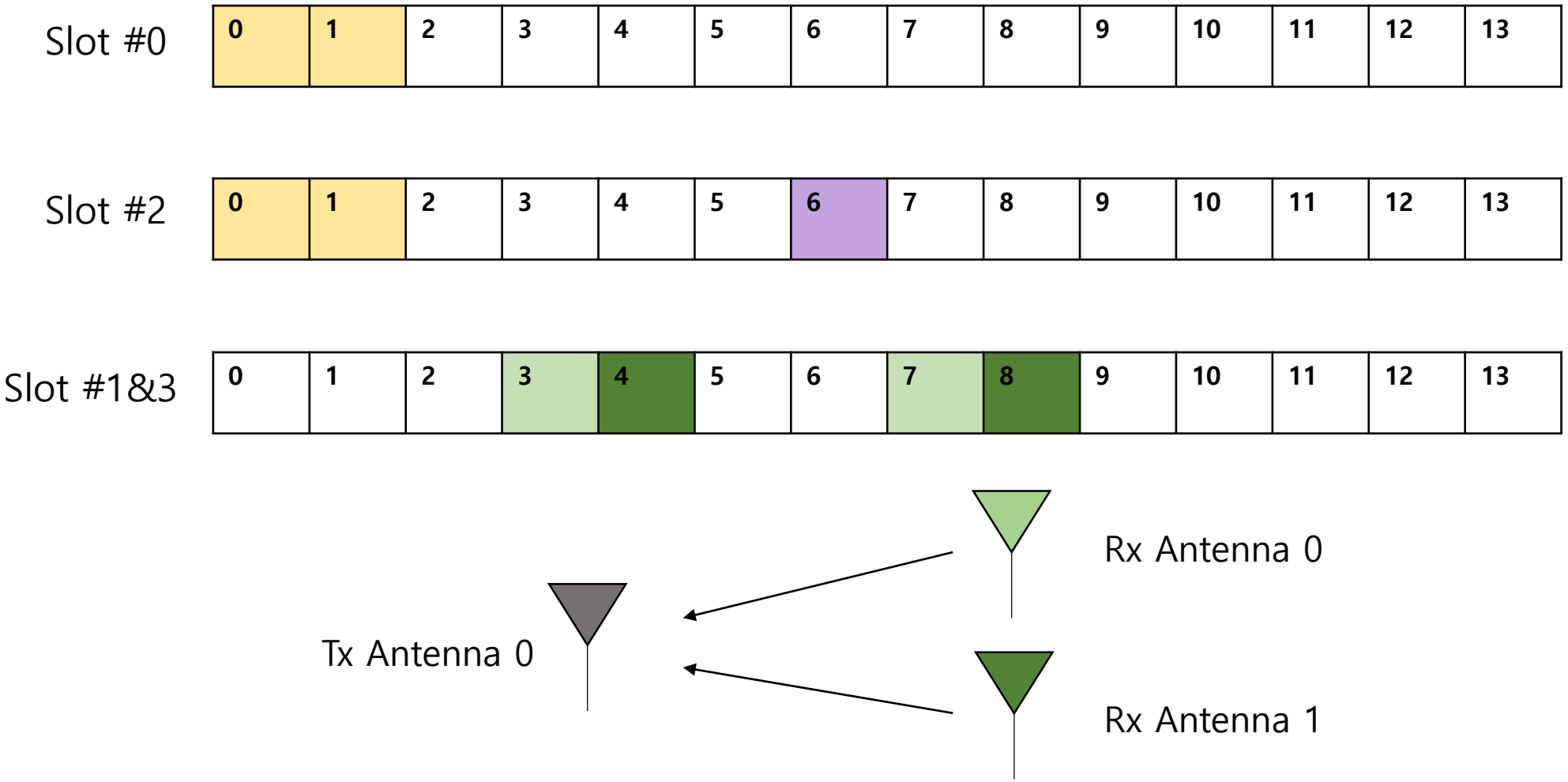
1. Frame Structure



Yellow : Data
Green : Dmrs
Purple : Preamble

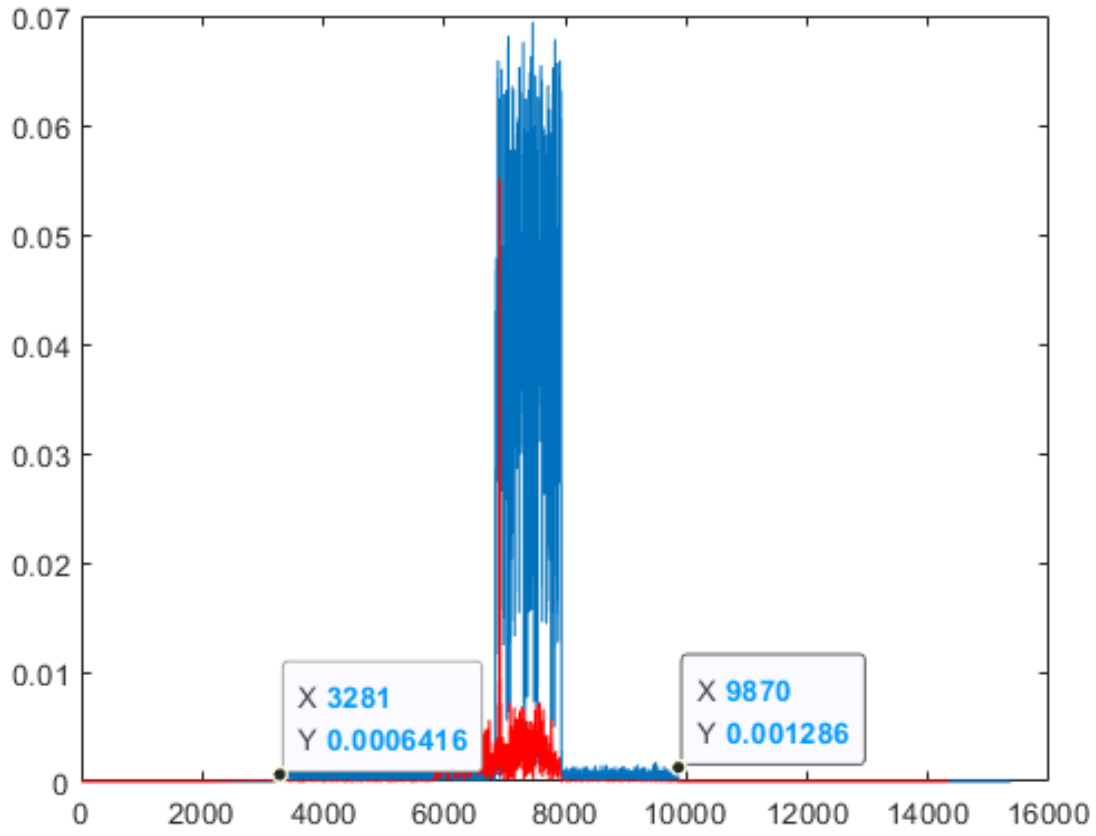


1. Frame Structure



2. Frame Structure Issue

Uplink



FFTSize = 1024, Cp0 = 16, Cp = 72

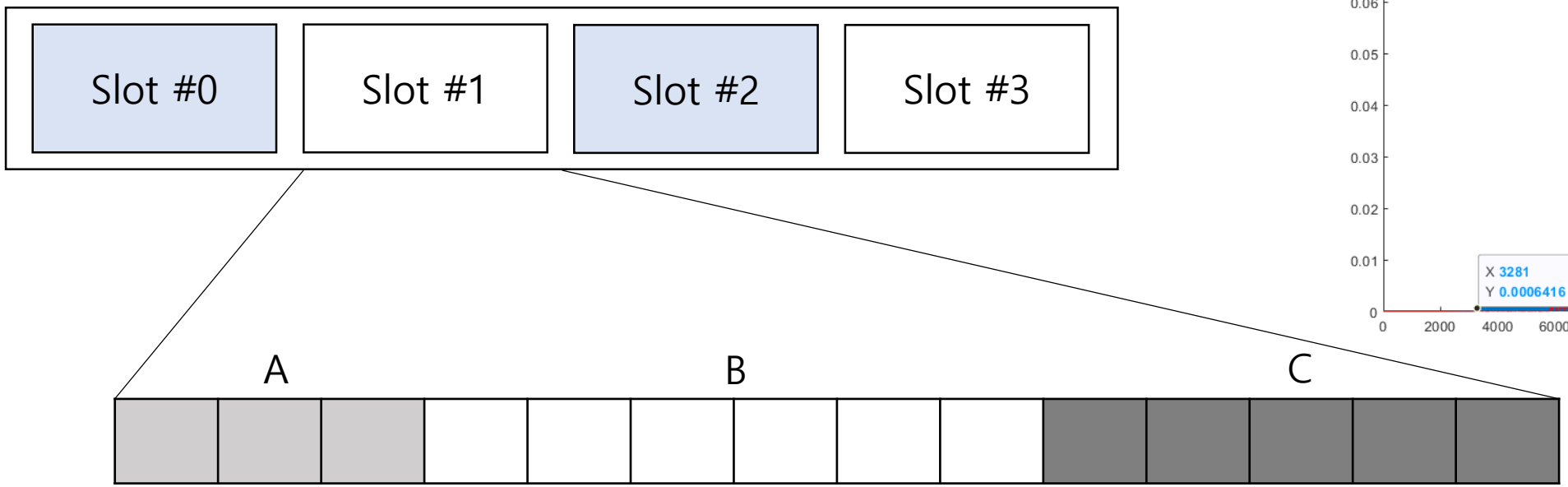
Issue 1)
Starting index of Symbol #3
= (FFTSize+Cp)*3 + Cp0
= 3304

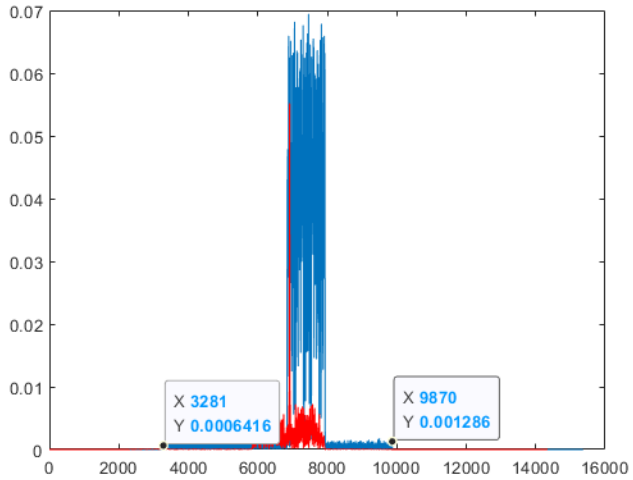
=> **3304 – 3281 = 23**

Issue 2)
Ending index of Symbol #8
= (FFTSize+Cp)*9 + Cp0
= 9880

=> **9880 > 9870**

2. RF state handler



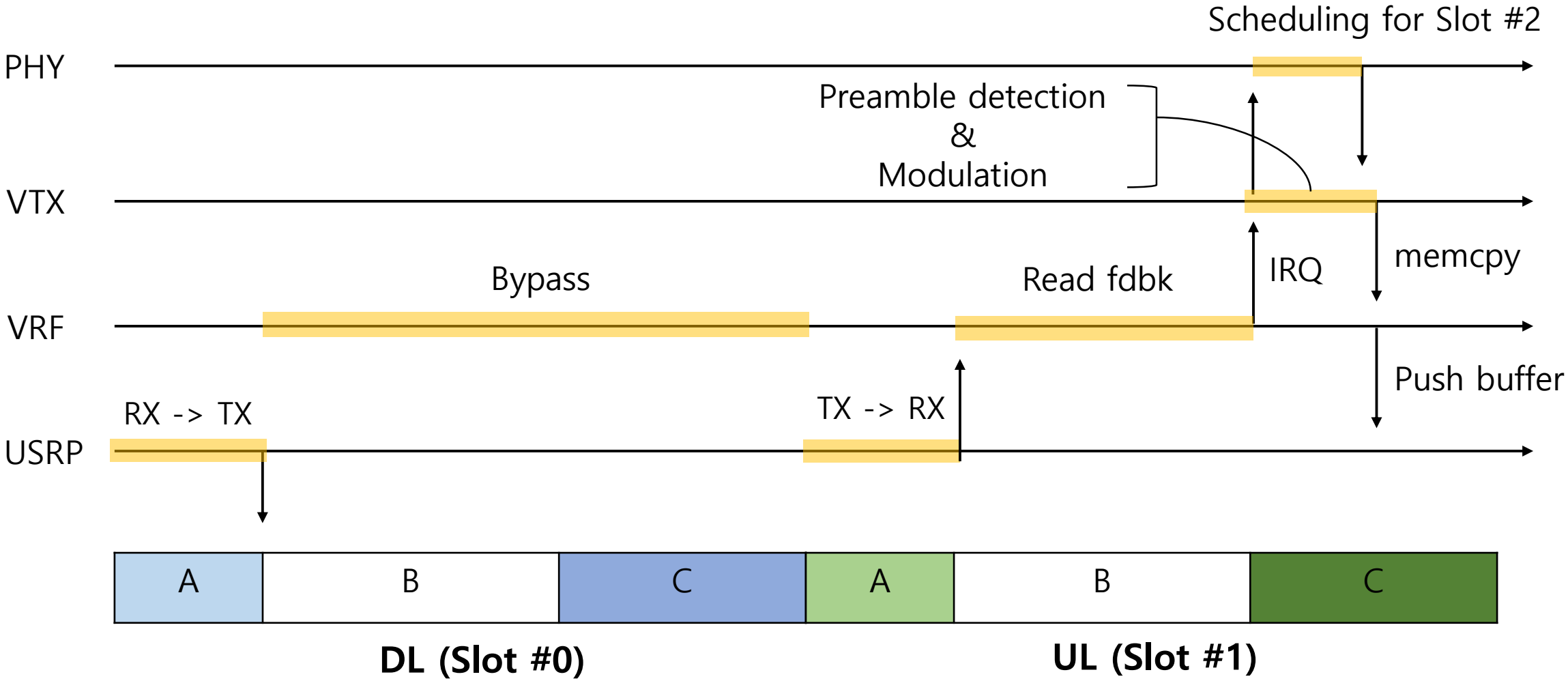


A : Guard Time ; for changing HW state => under 0.21429ms (1ms * 3/14)

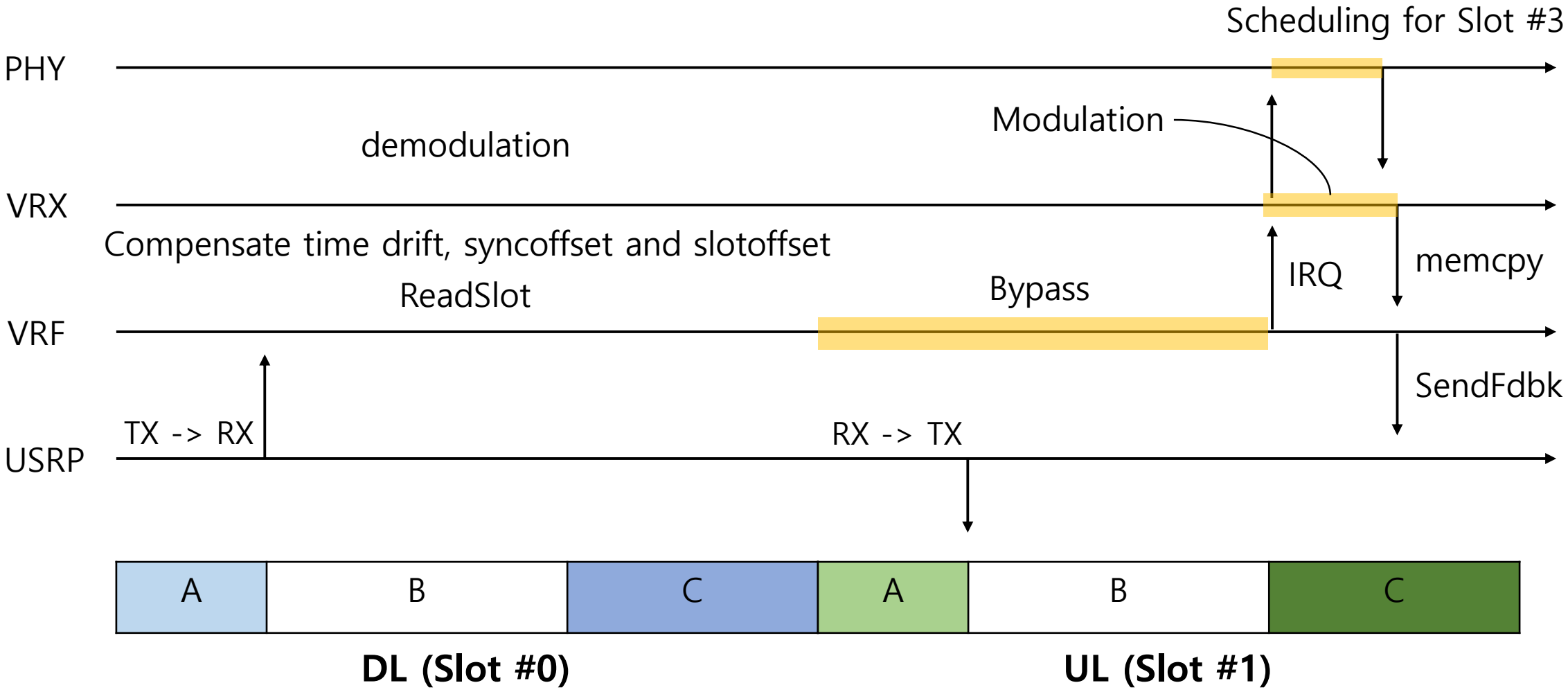
B : Read Slot => under 0.42857ms (1ms * 6/14)

C : SchedulTime + IQsendMargin ; Estimation of Channel H, modulation, Scheduling and Iq send margin
=> under 0.35714ms (1ms * 5/14)

3. Timing Diagram : TX



3. Timing Diagram : RX



4. Initial SetUp

1-1) vi runTx.sh

```
#!/bin/bash

sudo ./cmake_targets/ran_build/build/nr-uesoftmodem -C 3300000000 --TM 2 --BM 1 --node-type 1 --preamble-scale 2 --data-scale 2
~
```

1-2) vi runRx.sh

```
#!/bin/bash

  sudo ./cmake_targets/ran_build/build/nr-uesoftmodem -C 3300000000 --node-type 0 --TM 2 --BM 1 --capture-rx-data 1 --nb-rx-data 1000
```

```
#define CONFIG_HLP_DLF          "Set the downlink frequency for all component carriers\n"
#define CONFIG_HLP_TRANSMITMODE "Transmit mode (0 - SISO, 1 - 2x1 STBC, 2 - 1x2 STLC)"
#define CONFIG_HLP_BWMODE       "Bandwidth mode (0 - 5MHz, 1 - 10MHz, 2 - 20MHz, SCS : 15kHz (common))"
#define CONFIG_HLP_NODE_TYPE    "type of node (0 - rx, 1 - tx)\n"
#define CONFIG_HLP_SCALE_PRE    "Preamble scaledown level (power of 2)\n"
#define CONFIG_HLP_SCALE_DATA   "data scaledown level (power of 2)\n"
#define CONFIG_HLP_SCALE_PILOT  "pilot scaledown level (power of 2)\n"
#define CONFIG_HLP_CAPTURE      "data capture option\n"
#define CONFIG_HLP_NBRXDATA     "number of data to transmit\n"
```

	TM	BM
0	SISO	5MHz
1	STBC	10MHz
2	STLC	20MHz

2) CMakeList

```
#########################
# PHY options
#########################
add_boolean_option(MOD_STLC                 True "build option for STLC codes")
add_boolean_option(MOD_STBC                 True "build option for STBC codes")
add_boolean_option(DNF_PWROPTRF             True "optimizing RF power consumption")
if (CMAKE_SYSTEM_NAME STREQUAL "Darwin")
add_boolean_option(BUILDOPT_MACOS           True "building in macOS")
endif()
```

	MOD_STBC	MOD_STLC
SISO	True	False
STBC	True	False
STLC	True	True

※ If you changed the CMakeList, you must run cleanUE.sh

3) IQ Send Margin

```
//TX related parameters
#ifdef MOD_STBC
#define VRF_IQSENDMARGIN          250
#else
#define VRF_IQSENDMARGIN          250
#endif
#define VRF_MINSCHEDTIME          100
```

Change VRF_IQSENDMARGIN

STBC : 300
SISO & STLC : 250

5. USRP

TX/RX 모두 가능한 노드에 안테나 연결

TX

STLC 동작 시 주황색 불빛

RX

6. Parameter for Debugging


```
//#define VRX_DBGPRINT
```

TX Log

```
[PHY] [VRX] WARNING ::: too large frequency offset : 1135, needs to be compensated immediately
[PHY] detection success (time offset : 6840)
[PHY] [VRX] WARNING ::: too large frequency offset : 1103, needs to be compensated immediately
[PHY] detection success (time offset : 6840)
```

```
#define VRX_DBGPRINT
```

UL

```
[PHY] [VRF] 1 slot processing.....
[PHY] [VRF] schedule IRQ for slot 2 (fdbk), rest:3280   <- number of offset sub carrier
[VHW]  >>>> 3/4 rx :   : 697 us
[PHY] [vTX] slot IRQ from RF (2)
[VHW] >>>> signal to L1 : 0 us
[VHW] >>>> L1 schedule indication : 0 us
[VHW] >>>>>> VTX start preamble detection : 1 us
[PHY] preamble detection start :::::: start : 6400, end : 7200
[PHY] [VRX] WARNING ::: too large frequency offset : 1077, needs to be compensated immediately
[VHW] >>>>>> VTX end preamble detection : 65 us   <- preamble detection에 걸린 시간!!!
[PHY] detection success (time offset : 6844)
[VHW] >>>> VTX start modulation : 1 us
[VHW] >>>> VTX end modulation : 9 us
[PHY] [vTX] waiting scheduling IRQ
[VHW] >>>>>> return back to origin : 11 us
```

SchedTime

```
[VHW]  >>>> sendUSRP end :   : 126 us
[VHW]  >>>> receiveUSRP end :   : 206 us
```

IQSendMargin

```
[VHW] --> slot interval : 1035 us    <- 1ms에 근접
```

DL

```
[PHY] [VRF] 2 slot processing.....
[PHY] [VRF] schedule IRQ for slot 3 (normal), rest:3840, bitmap:5
[VHW]  >>>> 3/4 rx :   : 660 us   <- A+B 구간에서 걸린 시간
[VHW]  >>>> receiveUSRP end :   : 297 us
[VHW] --> slot interval : 961 us    <- 1ms 이내
```

7. RF Algorithm : Frame structure & RF handler

1>

```
#ifdef MOD_STLC
    if (txMode == 2)
    {
        txrxMode = nhal_trxMode_1x2STLC;
        antType = nhal_trxAnt_common;
        fdbk_pattern = 0xA; //bitmap of slots for feedback channel
        fdbk_width = nhal_fdbkWidth_3_6_5;
    }
#endif
```

← Slot pattern (0xA = 1010)

← Block design

2>

```
switch(fdbk_width)
{
    case nhal_fdbkWidth_1_2_1:
        nhal_rfic_ix->woReg.fdbkOffset = samples_per_slot/4;
        nhal_rfic_ix->woReg.fdbkDuration = samples_per_slot/2;
        break;
    case nhal_fdbkWidth_3_5_6:
        nhal_rfic_ix->woReg.fdbkOffset = samples_per_slot*3/14;
        nhal_rfic_ix->woReg.fdbkDuration = samples_per_slot*5/14;
        break;
    case nhal_fdbkWidth_3_6_5:
        nhal_rfic_ix->woReg.fdbkOffset = samples_per_slot*3/14;
        nhal_rfic_ix->woReg.fdbkDuration = samples_per_slot*6/14;
        break;
    default:
        LOG_E(PHY, "[ERROR] configuration fail : invalid feedback width (%i)\n", fdbk_width);
        return -1;
        break;
}
```

fdbkOffset	3291
fdbkDuration	6583
SchedTimeSample	1536
IQSendMargin	3840

← Interval A ; change HW state

← Interval B ; read Buffer

3>

```
vrf_IQSendMargin = (uint32_t)(VRF_IQSENDMARGIN*(vrf_rfDevcfg[0].sample_rate/1000000.0));
vrf_SchedTimeSample = (uint32_t)(VRF_MINSCHEDTIME*(vrf_rfDevcfg[0].sample_rate/1000000.0));
vrf_SchedTime = VRF_MINSCHEDTIME;
```

7. RF Algorithm : read slot

1> vrf_readFdbk ; TX에서 UL slot을 읽어올 때 사용

```
#ifdef MOD_STLC
void vrf_readFdbk(openair0_timestamp *timestamp, uint8_t slot_nb)
{
    void *rxp[HW_NB_RXANT];


    // 1. bypass
    for (int i=0; i<vrf_activatedAnt; i++)
    {
        memset(&vrf_rxdata[slot_nb][i][0], 0, vrf_fdbkConfig.offset*sizeof(int));
        rxp[i] = ((void *)&dummy_rx[i][0]);
    }


    AssertFatal(vrf_fdbkConfig.offset ==
                vrf_rfdevice.trx_read_func(&vrf_rfdevice,
                                           timestamp,
                                           rxp,
                                           vrf_fdbkConfig.offset,
                                           vrf_activatedAnt),
                "");



    //2. read fdbk
    for (int i=0; i<vrf_activatedAnt; i++)
        rxp[i] = ((void *)&vrf_rxdata[slot_nb][i][vrf_fdbkConfig.offset]);

    AssertFatal(vrf_fdbkConfig.duration ==                          // Add RXANT_ID
                vrf_rfdevice.trx_read_func(&vrf_rfdevice,
                                           timestamp,
                                           rxp,
                                           vrf_fdbkConfig.duration,
                                           vrf_activatedAnt),
                "");
}
#endif
```

A 구간 subcarrier bypass

B 구간 subcarrier 개수 만큼 읽어옴

2> vrf_readSlot ; RX에서 DL slot을 읽어올 때 사용

```
void vrf_readSlot(openair0_timestamp *timestamp, uint8_t slot_nb, int deltaDrift)
{
    void *rxp[HW_NB_RXANT];

    for (int i=0; i<vrf_activatedAnt; i++)
        rxp[i] = ((void *)&vrf_rxdata[slot_nb][i][0]);

    AssertFatal((vrf_samples_in_slot+deltaDrift )==
                vrf_rfdevice.trx_read_func(&vrf_rfdevice,
                                           timestamp,
                                           rxp,
                                           vrf_samples_in_slot+deltaDrift,
                                           vrf_activatedAnt),
                "");
}
```

Slot 전체를 읽어옴

7. RF Algorithm : send slot

1> vrf_sendSlot ; TX에서 DL slot을 전송할 때 사용

```
void vrf_sendSlot(openair0_timestamp *timestamp, uint8_t slot_nr, uint8_t slot_offset, int txrxOffset)
{
    void *txp[HW_NB_TXANT];
    int writeBlockSize = vrf_samples_in_slot;

    for (int i=0; i<vrf_activatedAnt; i++)
        txp[i] = (void *)&vrf_txdata[slot_nr][i][0];

    AssertFatal( writeBlockSize ==
                 vrf_rfdevice.trx_write_func(&vrf_rfdevice,
                                             *timestamp  + txrxOffset +
                                             slot_offset*vrf_samples_in_slot,
                                             txp,
                                             writeBlockSize,
                                             vrf_activatedAnt,
                                             4),"");
}
```

← 전체 subcarrier 전송

← A 구간이 지난 다음 USRP 전송 시작
ex) slot #1에서 processing -> slot #2에서 전송

2> vrf_sendFdbk ; RX에서 UL slot을 전송할 때 사용

```
#ifdef MOD_STLC
void vrf_sendFdbk(openair0_timestamp *timestamp, uint8_t slot_nr)
{
    void *txp[HW_NB_TXANT];
    int writeBlockSize = vrf_fdbkConfig.duration;

    for (int i=0; i<vrf_activatedAnt; i++)
        txp[i] = (void *)&vrf_txdata[slot_nr][i][vrf_fdbkConfig.offset];

    AssertFatal( writeBlockSize ==
                 vrf_rfdevice.trx_write_func(&vrf_rfdevice,
                                             *timestamp + vrf_fdbkConfig.offset +
                                             2*vrf_samples_in_slot,
                                             txp,
                                             writeBlockSize,
                                             vrf_activatedAnt,
                                             4),"");
}
```

← B 구간 만큼의 subcarrier만 전송

← 2 Slot + A 구간이 지난 다음 USRP 전송 시작
ex) slot #1에서 processing -> slot #3에서 전송

THANK YOU!