Please contact **Radha Krishna Ganti (Email:** rganti@ee.iitm.ac.in **) f*or further details***

Please contact **Radha Krishna Ganti (Email:** rganti@ee.iitm.ac.in **) f*or further details***

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and respect to my Guide, Prof. Radha Krishna Ganti who gave me this opportunity to work on upcoming 5G tech. I would like to thank my seniors Rajat and Aniruddh who have advised me when needed, throughout this project. I would also like to thank all my course teachers for providing the knowledge needed for the project. I am extremely grateful to my family for making me what I am today. Finally, I thank all my friends for helping me out when needed.

Contents

1. INITIAL ACCESS | PSS and SSS detection

5G NR

5G is specific new radio access technology being developed to meet needs of future mobile communication. The deployment of 5G can be classified in two ways: Non-Standalone (NSA) and Standalone (SA). NSA devices support 5G on 4G infrastructure and SA devices run on newly developed 5G basis.

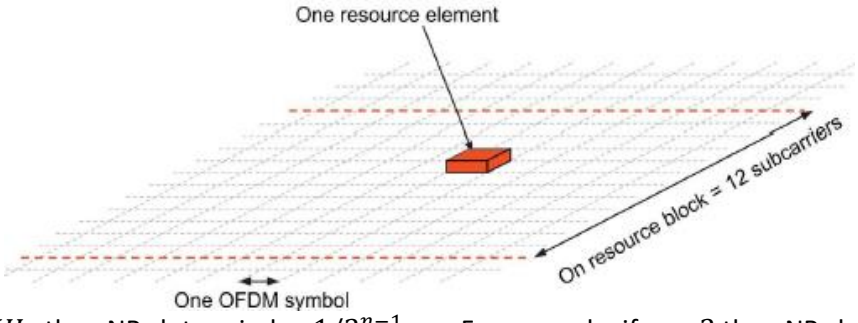
The different classes of 5G include: i) eMBB ,ii) mMTC and iii) URLLC. This classification is based on application requirements. First, enhanced Mobile Broad Band will support high data rates, enhanced user experience etc., to mobile broadbrand services. Second, Massive Machine Type Communication includes devices which generate low volumes of data, costs low and have high efficiencies for energy consumption. This type of application is used in IoT. The last one Ultra Reliable Low Latency Communication provide usage for networks which require very low latency and very high reliability for connectivity. For example this type usage can be seen in self-driving automated cars etc.

5G NR operates in sub-6 GHz which extends 4G LTE, operating from below 6GHz and other frequency range operates at a much higher 24250MHz to 52600MHz which is commonly referred to as mm Wave spectrum.

This new technology requires development of mm-Wave technologies, Massive MIMO, Hybrid beam forming, inclusion of small cells which complement macro cells, new modulation and coding (LDPC) schemes and physical layer algorithms. 5G is not all that different from 4G-LTE but sometime an improvement and sometime it's an introduction.

OFDM symbol

5G OFDM symbol construction and terminology is somewhat different than LTEs'. One slot in 5G NR consist of 14 symbols and slot duration varies with sub-carrier spacing (SCS). One Resource element is one symbol duration in time domain and one SCS in frequency domain. One Resource block corresponds to 12 SCS in frequency domain only and time boundaries aren't provided to have flexibility over different transmissions.



If $SCS = n * 15KHz$ then NR slot period $= 1/2^{n-1}\, ms$. For example, if $n = 2$ then NR slot period for 30KHz SCS is $0.5ms$

1) 5G PHY layer includes CRC encoding, LDPC (low density parity check coding), Rate-Matching & Hybrid ARQ, Scrambling, Modulation, Layer Mapping, Multi-Antenna pre-coding and Resource mapping. Data from MAC layer is fed to PHY layer in segments called transport blocks which is modified and transmitted after PHY layer operations.
2) Initial Access is most important and primary part for UE to access the 5G network by time and frequency synchronization through certain information contained in symbols called SSBs' and RACHs' for Downlink and Uplink synchronization respectively.

contents

Initial Access – Downlink Synchronization

Initial access is the first step performed by UE to access 5G NR. When an UE is turned on or lost its connectivity or didn't find a correct cell with required criterion then it has no idea of frame timing or carrier frequency of new cell which it would camp on. Initial access is performed by UE to find a new cell which meets cell criterion. Initial access includes two procedures 1) Cell Search which starts with frequency search 2) Random access which starts with PRACH (Physical Random Access Channel) for requesting resources by the UE to connected cell in first procedure. I here, discuss only Cell search part.

Cell Search

Cell Search is carried out when a device is entering the coverage area of a system. To enable mobility, cell search is also continuously carried out by devices moving within the system, both when the device is connected to the network and when in idle/inactive state. This process is based on certain symbols transmitted by NR cell periodically. Those set of symbols is called SSB (Signal Synchronization Block). SSB consists of 4 symbols namely:

1) PSS (Primary Synchronization Signal)
2) SSS (Secondary Synchronization Signal)
3) PBCH (data) and
4) PBCH (DMRS).

SSB in Frequency domain:

SSB transmission is based on OFDM, that is, they are transmitted on set of time and frequency resource elements. One SSB takes 4 symbols in time domain and 240 SCS in frequency domain. In LTE, SSB is always located at the centre of carrier frequency but in 5G NR a different method is adopted. SSBs' location in frequency domain is restricted to certain limited possible locations in each frequency band called synchronization raster which enables the device to search sparser synchronization raster making cell search faster. Scanning doesn't give exact location of SSB and device is informed about other required information by PBCH and remaining broadcast system information.

SSB in Time domain:

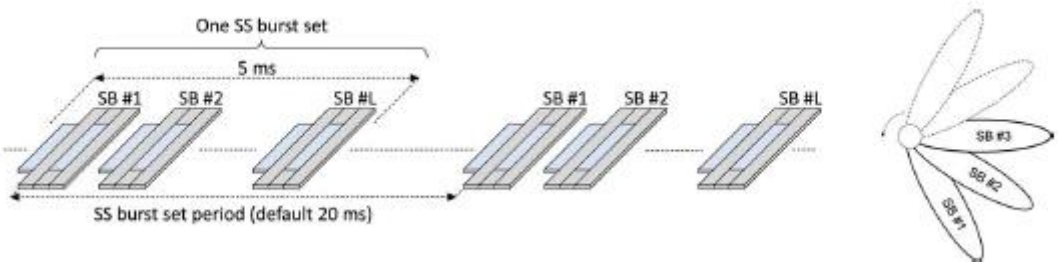
SSBs' are transmitted as a set called SSBurst. Each SSB has a period of transmission in time domain which can vary from 5ms to 160ms. But devices can assume 20ms periodicity. In LTE, SS Block periodicity is 5ms and in 5G NR larger periodicity is used to enhance NR network energy performance and Ultra-lean design (The principle which minimizes transmissions of always-on signals). Device waits for periodicity of time on each frequency in raster and moves on to next if not found. This method makes device to wait for longer periods but sparser raster compensates it. A shorter SSB periodicity (<20ms) is used for faster cell search and longer SSB periodicity is used by devices in connected mode for secondary carrier in carrier aggregation scenario.

In SSBurst, number of SSBs' is decided based on frequency used. Let L be SSBs' in SSBurst then:

L	**Frequency used(f)**
4	f < 3 GHz (FR1)
8	3 GHz < f <6 GHz (FR1)
64	mm-Wave range (FR2)

contents

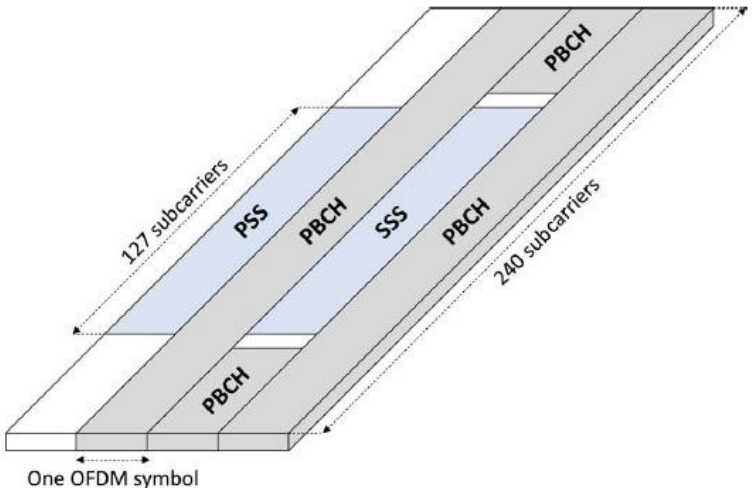
Each SSB corresponds to each beam generated by transmitter in a specific direction. By applying beam-forming for the SS block, the coverage of a single SS Block transmission is increased.



So, each SSB's periodicity is period of SSBurst set. Using 64 SSBs' for high frequency usage can be justified as using large number of SSBs' in one SSBurst takes more slots so more time in lower frequency range but with high frequency usage ,we use larger SCS, taking even less time per NR slot.

Now, SS Block locations are configured based on which SCS is used. Even though SSBurst set period can go up to 160ms, SSB set is confined to 5ms, that is, either to first half or second half of 10ms frame. PBCH includes a "time index" that explicitly provides the relative location of the SS block within the sequence of possible SS Block locations.

SSB Structure



Before going into structure of SSB, Physical Cell ID (PCI) is determined by two components $N_{ID}^{(1)}$ and $N_{ID}^{(2)}$. 5G NR provides us with 1008 unique cell IDs. And ID-1 is contained by SSS and ID-2 is contained by PSS.

PCI = $3 * \mathrm{ID1} + \mathrm{ID2}$ where ID1 = {0, 1, 2…335} and ID2 = {0, 1, 2}.

PSS:

PSS (Primary Synchronization Signal) is 1st symbol in SSB. It is generated by M-sequence consisting of 127 elements, that is, 127 SCS. It contains cell_ID2 and if device detects PSS, it has found periodicity up to PSS. It can then also use transmissions from the network as a reference for its internal frequency generation, thereby to a large extent eliminating any frequency deviation between the device and the network. As there are three different PSS sequences device has to search for all of them.

contents

SSS:

SSS (Secondary Synchronization Signal) is 3rd symbol which is suffixed and prefixed by PBCH. SSS is also constructed from 2 basic M-sequences consisting of 127 elements, that is, 127 SCS. PBCH uses 48 subcarriers on each side of SSS. So it leaves 8 and 9 sub-carriers empty on sides of SSS sequence. As device now know the PSS timing so it does of SSS and PSS made easy for device to detect 336 different SSSs' to find PCI.

PBCH:

PBCH (Physical Broadcast Channel) is over which channel coded information is transmitted whereas PSS and SSS are physical signals with specific structures. PBCH contains information about MIB (Master Information Block) which has information to obtain remaining broadcast information, that is, SIBs' (System Information Block). Each SSB's time index and half-frame bit (The half-frame bit indicates if the SS block is located in the first or second 5ms part of a 10ms frame.) present in PBCH is used to determine frame boundary. There are other important things to be checked by device from PBCH like whether cell is barred or not, SIB1 numerology (to know which SCS is used) etc,. PBCH contains following:

Information Carried Within the PBCH

Information	Number of Bits
SS-block time index	0 (FR1)/3 (FR2)
CellBarred flag	2
1st PDSCH DMRS position	1
SIB1 numerology	1
SIB1 configuration	8
CRB grid offset	5 (FR1)/4 (FR2)
Half-frame bit	1
System frame number (SFN)	10
Cyclic redundancy check (CRC)	24

Demodulation reference signals (DM-RS) for PDSCH are intended for channel estimation at the device as part of coherent demodulation. They are present only in the resource blocks used for PDSCH transmission. Similarly, the DM-RS for PUSCH allows the gNB to coherently demodulate the PUSCH.

After checking PBCH, device looks for SIB1 which contain scheduling information for Random Access from UE to cell selected. The **remaining SIBs**, not including SIB1, consist of the system information that a device does not need to know before accessing the system. These SIBs can also be periodically broadcast similar to SIB1. Alternatively, these SIBs can be transmitted on demand from device, that is, only transmitted when explicitly requested by the connected device. This implies that the network can avoid periodic broadcast of these SIBs in cells where no device is currently camping, thereby allowing for enhanced network energy performance.

contents

Fading and Doppler effects

Fading is variation of attenuation of a signal due of variables in channel. Fading may occur due to multipath propagation or shadowing from obstacles in path or weather. There are different types of fading to be discussed below. Before discussing them, some important channel characteristics should be known. (T_c) **Coherence time** is duration of time over which channel response is almost constant, that is, channel needs to be measured every T_c. (B_c) **Coherence bandwidth** is frequency range over which channel's frequency response is almost flat.(t_{max}) **Max delay spread** of channel is time value up to which impulse response of channel falls below certain threshold level. But for comparisons rms of delay spread (**ɾ̃**) is used. Here t_{max} and **ɾ̃** are not related.

B_c and RMS delay spread **ɾ̃** are related as $B_c = 1/(2 * \text{ɾ̃})$.

Flat fading: Let bandwidth of channel be B_s, then for this type of fading which occurs in frequency domain where $B_c > B_s$. **Frequency-selective fading:** In frequency domain $B_c < B_s$ and (t_{max} > symbol duration) so there will be Inter symbol Interference in time domain.

Slow fading: In time domain, T_c >> symbol duration of signal, that is, channel imposed amplitude and phase change is roughly constant over period of use. Channel can be estimated in time < T_c. It comes with low Doppler spread. **Fast fading:** This is exact opposite of slow fading. Channel estimation time is greater than T_c so channel can't be estimated. It comes with high Doppler spread as channel varies fast.

Block fading: Block fading is where the fading process is approximately constant for a number of symbol intervals.

Multipath fading is where signal is received from various possible paths available between transmitter and receiver. **Rayleigh fading** and **Richian fading** are one of these multipath fading where Richian has one LOS component and Rayleigh doesn't. **Deep fading:** Point where signal at receiver experiences strong destructive interference.

Doppler Shift is deviation introduced due of relative motion between transmitter and receiver. The transmitted carrier frequency (f_c) is deviated by Doppler shift (f_d). So received frequency $f_{rx} = f_c + f_d$. Doppler shift is given by $f_d = \left(\frac{v}{c}\right) * \cos\Theta * f_c$ where **v** is velocity's magnitude of receiver w.r.t transmitter, **Θ** is angle between v and line joining transmitter and receiver and **c** is speed of light in vaccum. (f_{ds}) Doppler Spread is twice of maximum Doppler shift (f_{dm}). $\mathbf{f_{dm}}$ can be related to T_c as follows:

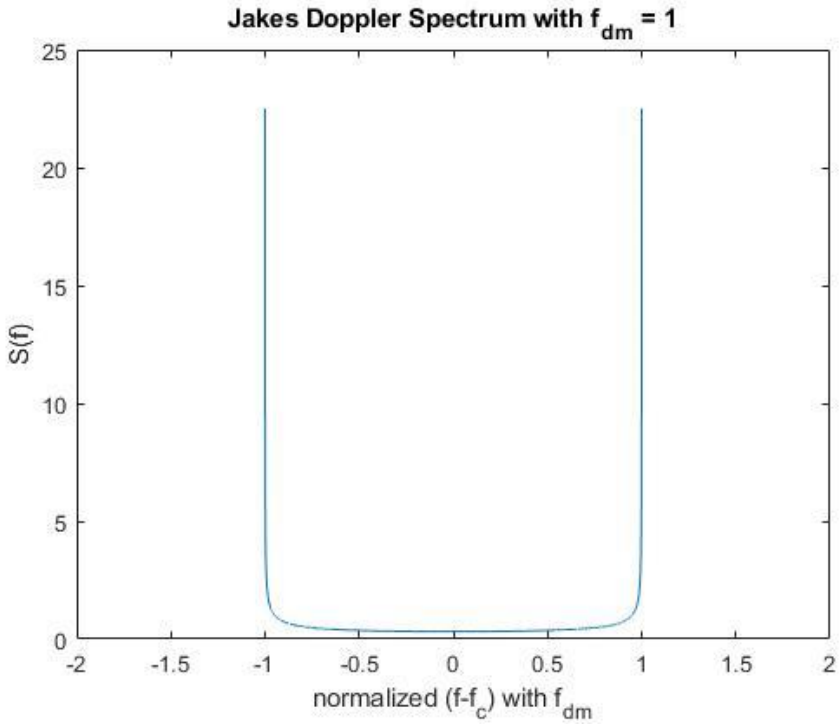
$$T_c = \frac{1}{4 * f_{dm}}.$$

This can be justified as if receiver is moving fast which changes conditions of channel quickly, so this decreases and increases T_c , f_{dm} respectively with v. Doppler spectrum describes the channel's power spectrum across the transmitted carrier frequency. Channels in simulation use Jakes/Classic Doppler spectrum. Jakes Doppler spectrum is given by equation (here f=0 refers to f_c frequency):

$$S(f) = \frac{1}{\pi * f_{dm} * \sqrt{1-(\frac{f}{f_{dm}})^2}} \text{ where } |f| < f_{dm}.$$

Above equation can be derived by taking Doppler shift equation making **Θ** subject and differentiating to get S(f) form. This when plotted gives bell shaped curve in frequency domain as shown below:

contents



Channels

The channel model to be used for simulation is a discrete wide sense stationary uncorrelated scattering (WSSUS) channel model for which the received signal is represented by the sum of delayed replicas of the input signal weighted by independent zero-mean complex Gaussian time variant processes. Specifically, if $z(t)$, $w(t)$ denote the complex low pass representations of the channel input and output, respectively, then:

$$w(t) = \sum_{n=1}^{N} \sqrt{p_n}\ g_n(t)\ z(t - \tau_n)$$

where p_n is the strength of the nth weight, and $g_n(t)$ is the complex Gaussian process weighting the nth replica.

The power spectrum of $g_n(t)$, called the Doppler spectrum of the nth path, controls the rate of fading due to the nth path. To completely define this channel model requires only a specification of the Doppler spectra of the tap weights $\{P_n(\upsilon); n = 1, \ldots, N\}$, the tap delays $\{\tau_n; n = 1, \ldots, N\}$, and the tap weight strengths $\{p_n; n = 1, \ldots, N\}$.

A type is classified to be low delay spread type channel and **B** type is classified to be large delay spread channels. In this area fast moving terminals (vehicles) should probably be connected to the macro cells to reduce the handoff rate (number of hand-offs per minute) and slow moving terminals (pedestrians) should be connected to the micro cells to achieve high capacity.

Channels used for simulation are **PED-A** and **VECH-A**. Tap strengths are given in dB and tap delays are given in ns in following table (relative delay corresponds to "all delays w.r.t to 1st tap"):

contents

Table A.2.6.2: ITU Channel Model for Outdoor to Indoor and Pedestrian Test Environment

Tap	Channel A		Channel B		Doppler spectrum
	Relative delay (ns)	Average power (dB)	Relative delay (ns)	Average power (dB)	
1	0	0	0	0	Classic
2	110	–9.7	200	–0.9	Classic
3	190	–19.2	800	–4.9	Classic
4	410	–22.8	1 200	–8.0	Classic
5	–	–	2 300	–7.8	Classic
6	–	–	3 700	–23.9	Classic

Table A.2.6.3: ITU Channel Model for Vehicular Test Environment

Tap	Channel A		Channel B		Doppler spectrum
	Relative delay (ns)	Average power (dB)	Relative delay (ns)	Average power (dB)	
1	0	0.0	0	–2.5	Classic
2	310	–1.0	300	0	Classic
3	710	–9.0	8.900	–12.8	Classic
4	1 090	–10.0	12 900	–10.0	Classic
5	1 730	–15.0	17 100	–25.2	Classic
6	2 510	–20.0	20 000	–16.0	Classic

Taking f_c = 1 GHz, v=1.5m/s (5.4kmph) and c is speed light gives max Doppler shift as 5Hz for Pedestrian Channel. Similarly for Vehicular channel with same f_c and taking $v \cong 90kmph$ we get max Doppler shift near 80Hz. For simulating Channels mentioned above in Matlab, I used **comm.RayleighChannel**() function available in Matlab. Matlab code for channels is given below:

PEDA:

```
function fd_frame_ch = PEDA(fd_frame,sampling_rate,SNR)
    H_doppler = comm.RayleighChannel('SampleRate',sampling_rate,'PathDelays',...
        [0 110 190 410]*1e-9,'AveragePathGains',[0.0 -9.7 -19.2 -22.8],...
    'MaximumDopplerShift',5,'DopplerSpectrum',doppler('Jakes'));
    fd_frame_ch = awgn(H_doppler(fd_frame),SNR,'measured');
end
```

VECHA:

```
function fd_frame_ch = VECHA(fd_frame,sampling_rate,SNR)
    H_doppler = comm.RayleighChannel('SampleRate',sampling_rate,'PathDelays',...
        [0 310 710 1090 1730 2510]*1e-9,'AveragePathGains',[0.0 -1.0 -9.0 -10.0
-15.0 -20.0],...
        'MaximumDopplerShift',80,'DopplerSpectrum',doppler('Jakes'));
    fd_frame_ch = awgn(H_doppler(fd_frame),SNR,'measured');
end
```

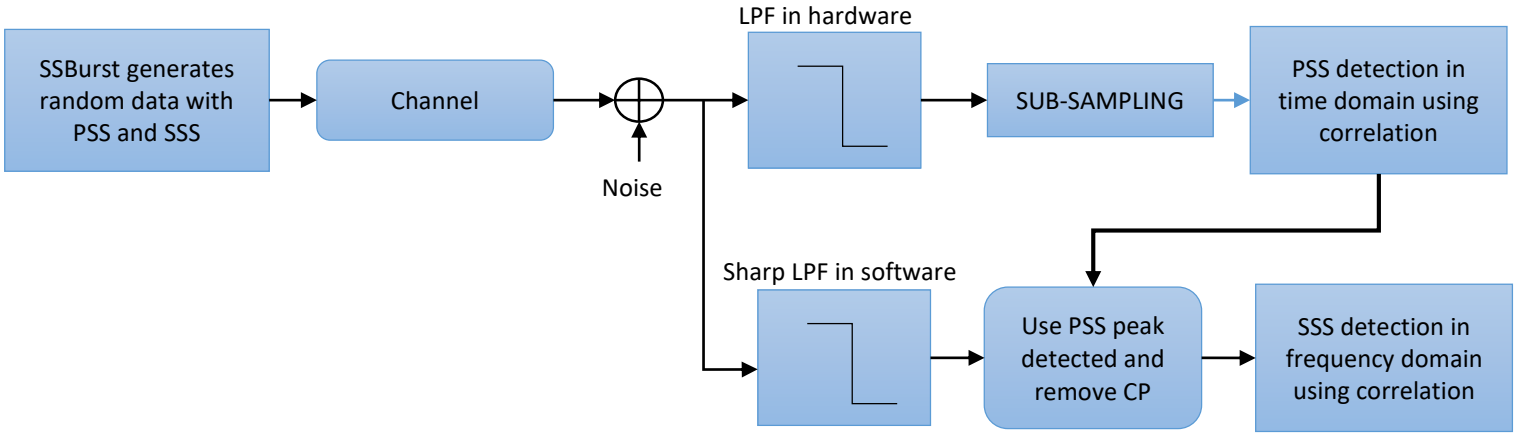
This completes construction of two channels considered for simulations. As discussed above for PSS, SSS and SSBurst, the first two can be generated from nrPSS(cell_id), nrSSS(cell_id) functions available in Matlab which take physical cell ID as input to generate 127 elements of ones and negative ones. **SSBurst** is written as function to create symbols of 4096 elements with random BPSK data and inserting PSS and SSS at specific indices adhering to structure of **SSB**.

contents

Matlab code for SSBurst function

```
%gives SS burst in time domain
function ssburst = SSBurst(slots_per_burst,L,cell_id,gap)
FRAME_SIZE = 4096;
ssburst = zeros(FRAME_SIZE*14*slots_per_burst,1);
for i=1:slots_per_burst*14
    frame = randi([0 1],FRAME_SIZE,1);
    frame = (2*frame-1);
    if (L==4) && (i<28)
        if rem(i-3,14)==0
            %frame = 0*frame;
            frame(round(FRAME_SIZE/2-
length(nrPSS(cell_id))/2):round(FRAME_SIZE/2+length(nrPSS(cell_id))/2)-1) =
nrPSS(cell_id);
            frame(round(FRAME_SIZE/2-length(nrPSS(cell_id))/2)-
gap:round(FRAME_SIZE/2-length(nrPSS(cell_id))/2)-1) = zeros(gap,1);

frame(round(FRAME_SIZE/2+length(nrPSS(cell_id))/2)+1:round(FRAME_SIZE/2+length(n
rPSS(cell_id))/2)+gap) = zeros(gap,1);

        elseif rem(i-5,14)==0
            %frame = 0*frame;
            frame(round(FRAME_SIZE/2-
length(nrSSS(cell_id))/2):round(FRAME_SIZE/2+length(nrSSS(cell_id))/2)-1) =
nrSSS(cell_id);
            frame(round(FRAME_SIZE/2-length(nrSSS(cell_id))/2)-
gap:round(FRAME_SIZE/2-length(nrSSS(cell_id))/2)-1) = zeros(gap,1);

frame(round(FRAME_SIZE/2+length(nrSSS(cell_id))/2)+1:round(FRAME_SIZE/2+length(n
rSSS(cell_id))/2)+gap) = zeros(gap,1);

        elseif rem(i+5,14)==0
            %frame = 0*frame;
            frame(round(FRAME_SIZE/2-
length(nrPSS(cell_id))/2):round(FRAME_SIZE/2+length(nrPSS(cell_id))/2)-1) =
nrPSS(cell_id);
            frame(round(FRAME_SIZE/2-length(nrPSS(cell_id))/2)-
gap:round(FRAME_SIZE/2-length(nrPSS(cell_id))/2)-1) = zeros(gap,1);

frame(round(FRAME_SIZE/2+length(nrPSS(cell_id))/2)+1:round(FRAME_SIZE/2+length(n
rPSS(cell_id))/2)+gap) = zeros(gap,1);

        elseif rem(i+3,14)==0
            %frame = 0*frame;
            frame(round(FRAME_SIZE/2-
length(nrSSS(cell_id))/2):round(FRAME_SIZE/2+length(nrSSS(cell_id))/2)-1) =
nrSSS(cell_id);
            frame(round(FRAME_SIZE/2-length(nrSSS(cell_id))/2)-
gap:round(FRAME_SIZE/2-length(nrSSS(cell_id))/2)-1) = zeros(gap,1);

frame(round(FRAME_SIZE/2+length(nrSSS(cell_id))/2)+1:round(FRAME_SIZE/2+length(n
rSSS(cell_id))/2)+gap) = zeros(gap,1);
        end
    end
    if (L==8) && (i>28 && i<56)
        %use above condition's code
    end
    ssburst((i-1)*FRAME_SIZE+1:i*FRAME_SIZE) = ifft(ifftshift(frame));
end
end
```

Layout of PSS and SSS detection



Before detecting PSS, Frequency search is carried on, **assumption**: Carrier frequency is found for simulation. SSBurst generates time domain data, that is, it performs **ifft**( ) operation on each 4096 symbol and returns an array containing time domain symbols. In PSS detection file shown below, I have added CP to each 4096 time domain symbol and carried channel operations on whole time domain signal. SCS used is 30KHz which implies NR_slot duration = 0.5ms and operations performed throughout are in baseband. Channels could be chosen from CHANNEL part by providing values {0,1,2} to variable **Channel** in code given below.

Matlab code for PSS detection

```
%one slot=0.5ms i.e 30KHz SCS and 14 symbols per slot each with 4096 fft bins
%so sample period is at max 0.5*1.744*1e-8= 8.7193ns
%in freq domain symbol size=4096 & SCS with 30KHz
%so we have 4096*30000KHz = 122880KHz BW in baseband
%1frame=1symbol here
%2ss bursts taken, 1SS burst = 20slots,1slot=14symbols
%Let beams produced be L=4 for fc<3GHz

FRAME_SIZE = 4096;
gap=57;
CFO = 0;%Hz
TX_FREQ = 1e+9;%Hz, in passband
RX_FREQ = 1e+9; %Hz, in passband
SS_Bursts=2;
CP = 288;
REF_FRAME_SIZE = 4096;%time domain symbol size of tx
L=4;
cell_id = 17;
pss = nrPSS(cell_id);
slots_per_burst=5;
pss_locs = ([[3 9 17 23],70+[3 9 17 23]]-1)*(FRAME_SIZE+CP)+CP+1;%time domain
sss_locs = ([[3 9 17 23],70+[3 9 17 23]]+1)*FRAME_SIZE+1;%freq domain
NUM_FRAMES = SS_Bursts*slots_per_burst*14;
a = zeros(NUM_FRAMES*(FRAME_SIZE+CP),1);
b = zeros(NUM_FRAMES*(FRAME_SIZE),1);

                    %Next comes the transmitter part using SSBurst
```

contents

```
%-----------------------------TRANSMITTER---------------------------------
fd_frame =complex(a,a);
tx_frame_f = complex(b,b);
for j=1:SS_Bursts
    ssburst = SSBurst(slots_per_burst,L,cell_id,gap);
    for i=1:14*slots_per_burst
        ifft_frame = ssburst((i-1)*FRAME_SIZE+1:i*FRAME_SIZE);
        complete_frame = [(ifft_frame(end-CP+1:end));ifft_frame];
        fd_frame(((j-1)*14*slots_per_burst*(FRAME_SIZE+CP))+(i-
1)*(FRAME_SIZE+CP)+1:((j-
1)*14*slots_per_burst*(FRAME_SIZE+CP))+i*(FRAME_SIZE+CP)) = complete_frame;
        %tx_frame_f(((j-1)*14*slots_per_burst*(FRAME_SIZE))+(i-
1)*FRAME_SIZE+1:((j-1)*14*slots_per_burst*(FRAME_SIZE))+i*FRAME_SIZE) =
fftshift(fft(ifft_frame));
    end
end
%-------------------------------CHANNEL-----------------------------------
%PEDA-->SNR=-8dB | VECHA-->SNR=0dB
Channel=2;
sampling_rate = 122.88*1e+6;
if Channel==1
    disp('Channel PED-A ON')
    SNRdB=-8;
    fd_frame_ch = PEDA(fd_frame,sampling_rate,SNRdB);
elseif Channel==2
    disp('Channel VECH-A ON')
    SNRdB=0;
    fd_frame_ch = VECHA(fd_frame,sampling_rate,SNRdB);
else
    disp('Channel off')
    fd_frame_ch = fd_frame;
end
%here fd_frame is corrupted with channel and noise
%-------------------------------RECEIVER----------------------------------
%CFO effect
fd_frame_rx = fd_frame_ch*exp(-1j*2*pi*CFO);
fd_frame_sss = fd_frame_rx;%for sss detection
%LOW PASS FILTER (very imp)
cutoff1 = (round(length(pss)/2)+1)/(FRAME_SIZE/2);
cutoff2 = cutoff1+1e-4;
lpf = firpm(95,[0 cutoff1 cutoff2 1],[1 1 0 0],[10 1]);
fd_frame_rx = conv(fd_frame_rx,lpf','same');

%--------------------sub_sampling-----------------
SUB_SAMPLE = 32;%should divide 4096 into an integer
REF_FRAME_SIZE = REF_FRAME_SIZE/SUB_SAMPLE;
fd_frame_sub = decimate(fd_frame_rx,SUB_SAMPLE);
%resample(fd_frame_rx,1,SUB_SAMPLE); %fd_frame_rx(1:SUB_SAMPLE:end);
ref_pss_frame = zeros(REF_FRAME_SIZE,1);
ref_pss_frame(round(REF_FRAME_SIZE/2-
length(pss)/2):round(REF_FRAME_SIZE/2+length(pss)/2)-1) = pss;
ref_pss_frame = ifft(ifftshift(ref_pss_frame));
%----------------cross correlation-------------------
%mag_ref_pss = sqrt(sum(abs(ref_pss_frame).^2));
pss_corr = zeros(length(fd_frame_sub),1);
for i=1:(length(fd_frame_sub)-length(ref_pss_frame)+1)
    pss_corr(i) = sum(fd_frame_sub(i:i+length(ref_pss_frame)-
1).*conj(ref_pss_frame));
    nm_pss = sqrt(sum(abs(fd_frame_sub(i:i+length(ref_pss_frame)-1)).^2));
    pss_corr(i) = pss_corr(i)/nm_pss;
end
```

```
figure(1)
plot(1:length(pss_corr),(pss_corr));
title(['pss correlation with PED_A channel with SNR = ',num2str(SNRdB),'
dB']);

%----------------Peak Detection-----------------
if SUB_SAMPLE==1
    N = 65;
elseif SUB_SAMPLE==32
    N=1;% 5 9 17 33
end
val = zeros(length(pss_corr),1);
for i=1:length(pss_corr)-N+1
    avg = abs(mean(pss_corr(i:i+N-1)));
    val(i) = abs(pss_corr(i+(N-1)/2))*avg;
end
[p,locs]=findpeaks(val);
[A,B]=sort(p,'descend');
npt = 4*SS_Bursts;
peak_indices = sort(locs(B(1:npt)));
%what if 2 local maxima occur at one peak
figure(2);subplot(2,1,1);plot(val);title('Peak detection graph in pss-corr');
subplot(2,1,2);stem(peak_indices,val(peak_indices));
legend(['PSS Peaks locs in sub-sampled rx -
>',num2str(round(pss_locs/SUB_SAMPLE))],'Location','best');

%-----------------------------SSS_detection--------------------------------
%max peak found is pss_peak_index in 10+ slot buffer
sss_corr_f = SSS_det(fd_frame_sss,locs(B(1)),SUB_SAMPLE,cell_id);
```

Low pass filter passband cutoff is computed by fraction of PSS sequence length in FRAME_SIZE as PSS lies at centre of 4096 symbol in frequency domain, so, PSS is filtered. After sub-sampling the filtered time-domain received signal, correlation is computed between that time domain signal and ifft of $(\frac{FRAME\_SIZE}{SUB\_SAMPLE})$ point symbol with PSS in centre. This correlation is computed in time domain and peak index detected is interpolated to original sequence (w/o sub-sampling) in SSS_det( ) function. Here, Peak Detection is used to get clear peaks from the correlation output. Variable slots_per_burst = 5 which takes 5ms for 2 SSBursts transmitted. Receiver in 5ms should detect 8 PSS, that is, 4 PSS symbols per burst (L=4).

Next step is to take max peak of 8 peaks which indicates the beam directed to UE by the cell. Then index of max peak is passed as an argument to SSS_det( ) function.

Matlab code for SSS_det( ) function

```
%sss detection in frequency domain
function sss_corr_f = SSS_det(fd_frame_sss,pss_peak_index,SUB_SAMPLE,cell_id)
CP=288;
FRAME_SIZE=4096;
sampling_rate = 122.88*1e+6;
sss = nrSSS(cell_id);
cutoff1 = (round(length(sss)/2)+1)/(FRAME_SIZE/2);
cutoff2 = cutoff1+1e-2;
h = FIRtypeI(cutoff1,cutoff2,sampling_rate);

pss_peak_rx = SUB_SAMPLE*(pss_peak_index)+1;
fd_frame_rx1 = conv(fd_frame_sss,h,'same');
```

```
%removing CP and perfroming fft
rx_frame=[];
j=0;
t=1;%t={0,1,2}
%CP removing after peak index
for i =
pss_peak_rx+(t*FRAME_SIZE+CP):(FRAME_SIZE+CP):(pss_peak_rx+2*(FRAME_SIZE+CP))
    j=j+1;
    rx_frame((j-1)*FRAME_SIZE+1:j*FRAME_SIZE) = ...
        fftshift(fft(fd_frame_rx1(i:i+FRAME_SIZE-1)));
end
rx_frame = rx_frame';%actual rx signal in f domain
%rx_frame_z = [rx_frame;zeros(length(sss),1)];
%sss_corr in f domain
sss_corr_f = zeros(length(rx_frame)+length(sss),1);
%mag_sss = sqrt(sum(abs(sss).^2));
for i=1:length(rx_frame)-length(sss)+1
    sss_corr_f(i) = sum(rx_frame(i:i+length(sss)-1).*sss);%conj for sss is not
needed
    sss_corr_f(i) = sss_corr_f(i)/(sqrt(sum(abs(rx_frame(i:i+length(sss)-
1)).^2)));
end
sss_corr_f = abs(sss_corr_f).^2;

%----------------------Peak detection-----------------
[p1,locs1]=findpeaks(sss_corr_f);
[~,B1]=sort(p1,'descend');
sss_loc = locs1(B1(1));
sss_loc_f = (2-t)*(FRAME_SIZE)+ FRAME_SIZE/2 - round(length(sss)/2)+1;
figure(3);subplot(2,1,1);plot(sss_corr_f);title('sss correlation in freq
domain');
subplot(2,1,2);stem(sss_loc,sss_corr_f(sss_loc));title('SSS Peak from peak-
detection');
legend(['SSS-locs in freq domain-
>',num2str(sss_loc_f)],'Location','northeast');
sss_est = rx_frame(sss_loc:sss_loc+length(sss)-1);
sss_t = ifft(ifftshift(sss));
sss_est_t = ifft(ifftshift(sss_est));
figure(4);plot(unwrap(angle(sss_est_t)-angle(sss_t))*180/pi);
title('Phase differnce b/w sss-estimated and sss in freq
domain');ylabel('degrees');
figure(5);plot(1:127,unwrap(angle(sss_t))*180/pi,...
    1:127,unwrap(angle(sss_est_t))*180/pi);ylabel('degrees');
legend('phase of sss','phase of sss-estimated');
end
```

Unfiltered signal is filtered with sharp filter in software. Low pass filter taps in time domain are constructed using remez exchange algorithm. Next CP is removed with reference **pss_peak_rx**, perform **fft** to get frequency domain two or three 4096 point symbols from that reference point leaving PSS symbol. Then similar procedure is followed in detecting SSS but take 127 point SSS sequence for better detection. Absolute of SSS correlation output is computed for clear peaks. Then max peak of those peaks detected is taken for finding the 4096 symbol boundary. LPF in software → FIRtypeI( ) function takes three inputs passband, stopband cutoffs and sampling rate and returns taps of impulse response.

contents

Matlab code for FIRtypeI filter function

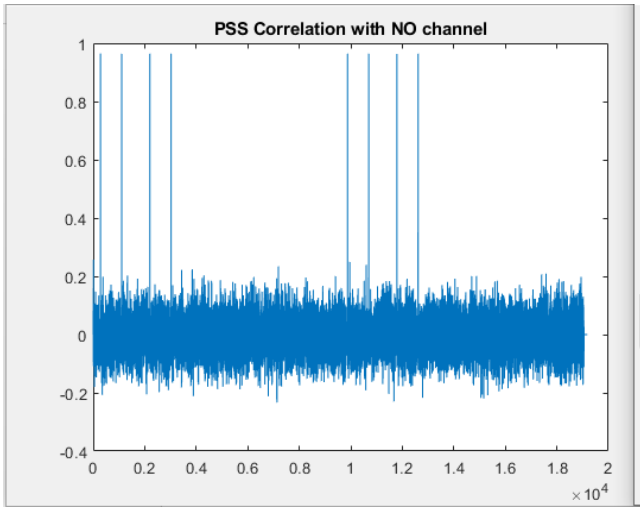
```
function h = FIRtypeI(cutoff1,cutoff2,sampling_freq)
rp = 0.1;          % passband ripple in dB
rs = 42;         % stopband ripple in dB
Fs = sampling_freq;       % sampling frequency
Kp = 1/(1-10^(-rp/20)); Ks = 10^(rs/20); %generally we choose 1/del_s,1/del_p
for Ks and Kp respectively
wp = cutoff1*pi; %passband edge frequency
ws = cutoff2*pi; %stop band edge frequency
wo = (ws+wp)/2;  %cutoff frequency
Length = 1e+5;
w = (0:Length)'*pi/Length;%discrete w declaration
bandedges = [wp*Fs/(2*pi) ws*Fs/(2*pi)];   % cutoff frequencies
dg = [1 0];       % desired amplitudes
dev = [(10^(rp/20)-1)/(10^(rp/20)+1)  10^(-rs/20)];
[n,fo,ao,w1] = remezord(bandedges,dg,dev,Fs);
if(mod(n+1,2)==0)
    N=n+2;   %Taking one extra sample and for this we get better specs

else
    N=n+1;   %n(=N-1) order can be found out by remezord function avaiable in
MATLAB
end
c = (N-1)/2;
R = c + 2;   % R = size of reference set

W = Kp*(w<=wp) + Ks*(w>=ws);%weighting function for error (A-D)
D = (w<=wo); %desired magnitude for filter required

SN = 1e-10;   % small number for stopping criteria

% initialize reference set (approx equally spaced where W>0)
f = find(W>SN);                         %find locations for f where W is
greater than SN
k = f(round(linspace(1,length(f),R))); %get R values uniformly for initial
iteration
m = 0:c;          %index for Ac(w)
Length=length(W)-1;
s = (-1).^(1:R)'; % signs
while 1
% --------------- Solve Interpolation Problem ---------------------
x = [cos(w(k)*m), s./W(k)]\D(k);
a = x(1:c+1); % cosine coefficients
del = x(c+2); % delta
h = [a(c+1:-1:2); 2*a(1); a(2:c+1)]/2;
A = Real_Amplitude(h,1,Length)';
err = (A-D).*W; % weighted error
% --------------- Update Reference Set ---------------------------
%[pks,locs1]=findpeaks(err);
%[pks,locs2]=findpeaks(-err);
newk = sort([locmax(err);locmax(-err)]);

errk = (A(newk)-D(newk)).*W(newk);

% remove frequencies where the weighted error is less than delta
v = abs(errk) >= (abs(del)-SN);
newk = newk(v);
errk = errk(v);
```

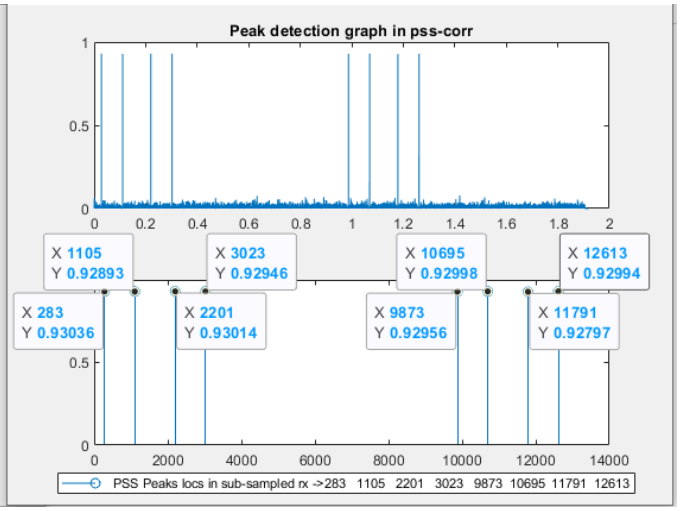
```
% ensure the alternation property
v = Update_Criteria(errk);
newk = newk(v);
errk = errk(v);

% if newk is too large i.e >R , remove points until size is correct
while length(newk) > R
if abs(errk(1)) < abs(errk(length(newk)))
newk(1) = [];
else
newk(length(newk)) = [];
end
end
% --------------- Check Convergence ------------------------------
if (max(errk)-abs(del))/abs(del) < SN
disp('done.')
break
end
k = newk;
end
del = abs(del);
h = [a(c+1:-1:2); 2*a(1); a(2:c+1)]/2;
end

%Real_Amplitude function
function [A,w] = Real_Amplitude(h,type,Length)

h = h(:)';      % make h a row vector
N = length(h);      % length of h
H = fft(h,2*Length);        % zero pad and fft
H = H(1:Length+1);      % select [0,pi]
w = (0:Length)*pi/Length;       % frequency grid
c = (N-1)/2;
if (type == 1)||(type == 2)
  H = exp(c*1j*w).*H;   % Type I and II
else
 H = -1j*exp(c*1j*w).*H;    % Type III and IV
end
A = real(H);
end

%Update_Criteria function
function v = Update_Criteria(E)

if size(E,1) > 1
a = 1;
else
a = 0;
end
j = 1;
xe = E(1);
xv = 1;
for i = 2:length(E)
if sign(E(i)) == sign(xe)
if abs(E(i)) > abs(xe)
xe = E(i);
xv = i;
end
else
v(j) = xv;
j = j + 1;
xe = E(i);
```

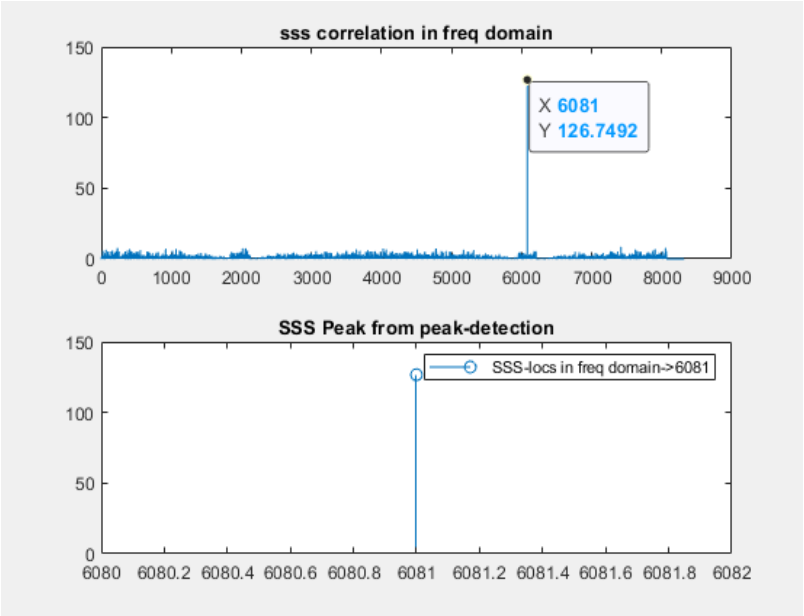
```
xv = i;
end
end
v(j) = xv;
if a == 1
v = v(:);
end
end

%localmax function
function maxind = locmax(x)
% finds indices of local maxima of data x
x = x(:);
n = length(x);
maxind = find(x > [x(1)-1;x(1:n-1)] & x > [x(2:n);x(n)-1]);
end
```

Plots with no channel

PSS peaks locs are actual locations known to transmitter and found locations are ones shown in boxes.

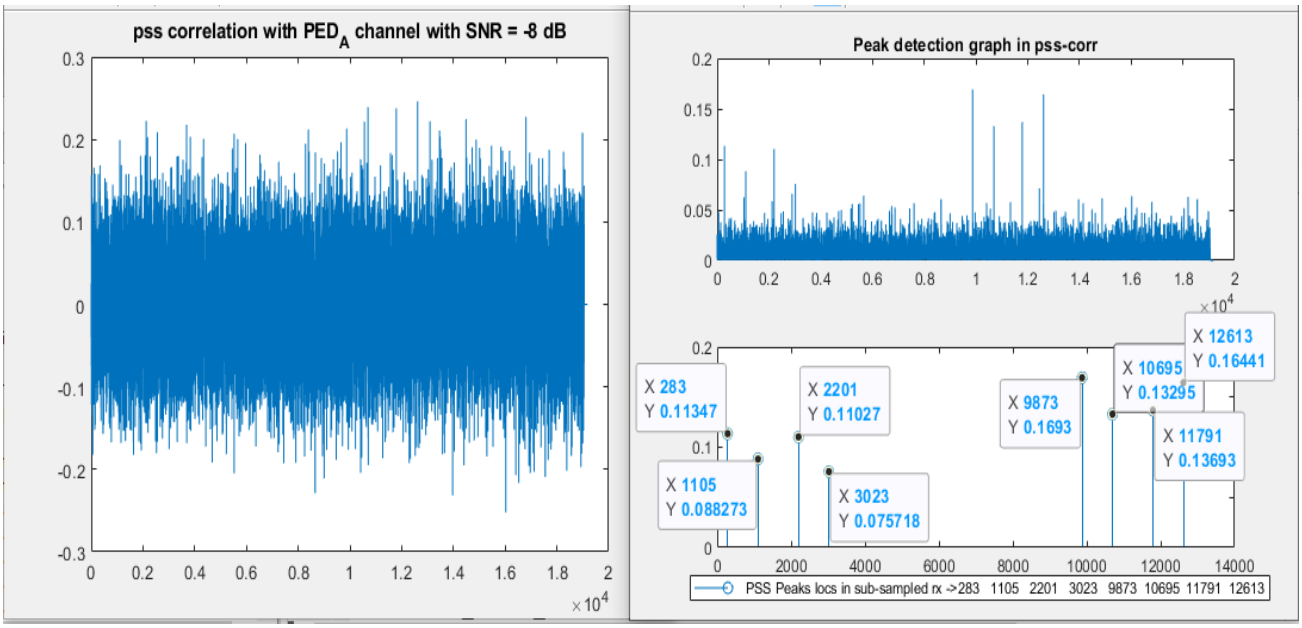




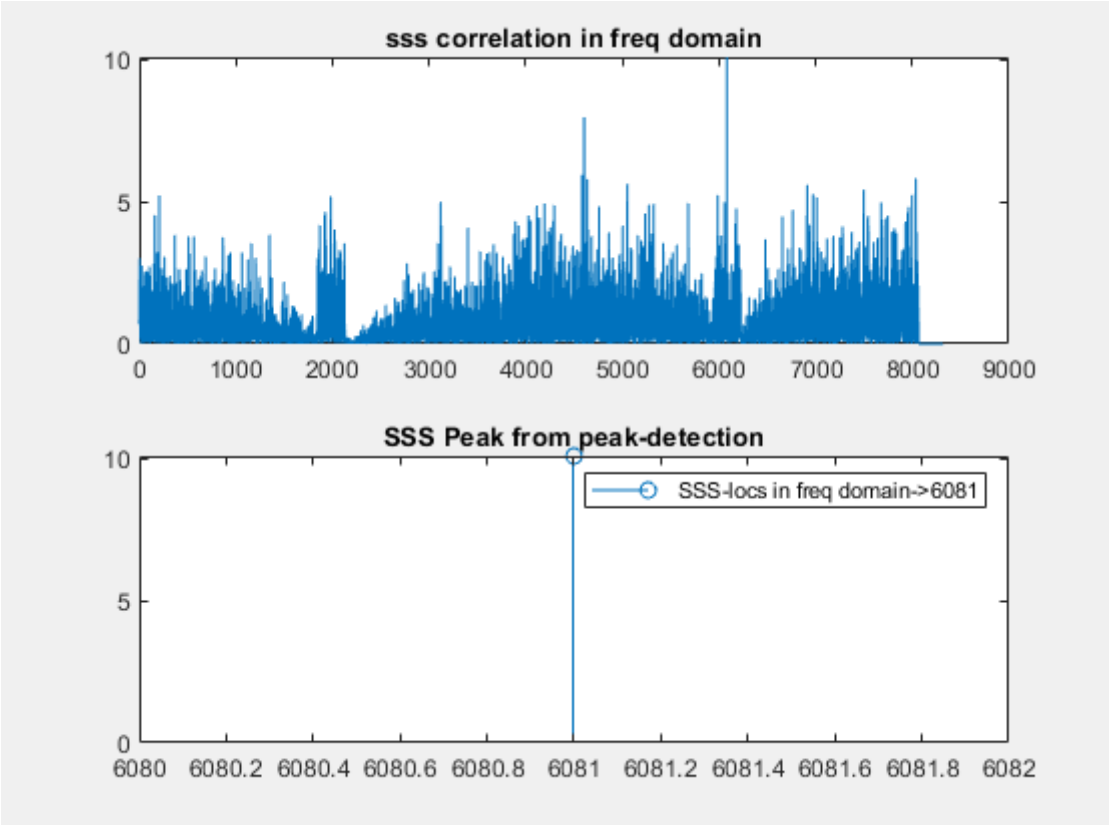


Plots with PED-A Channel

Here found peaks may miss actual locations due to noise but maximum peak will mostly be at actual location.

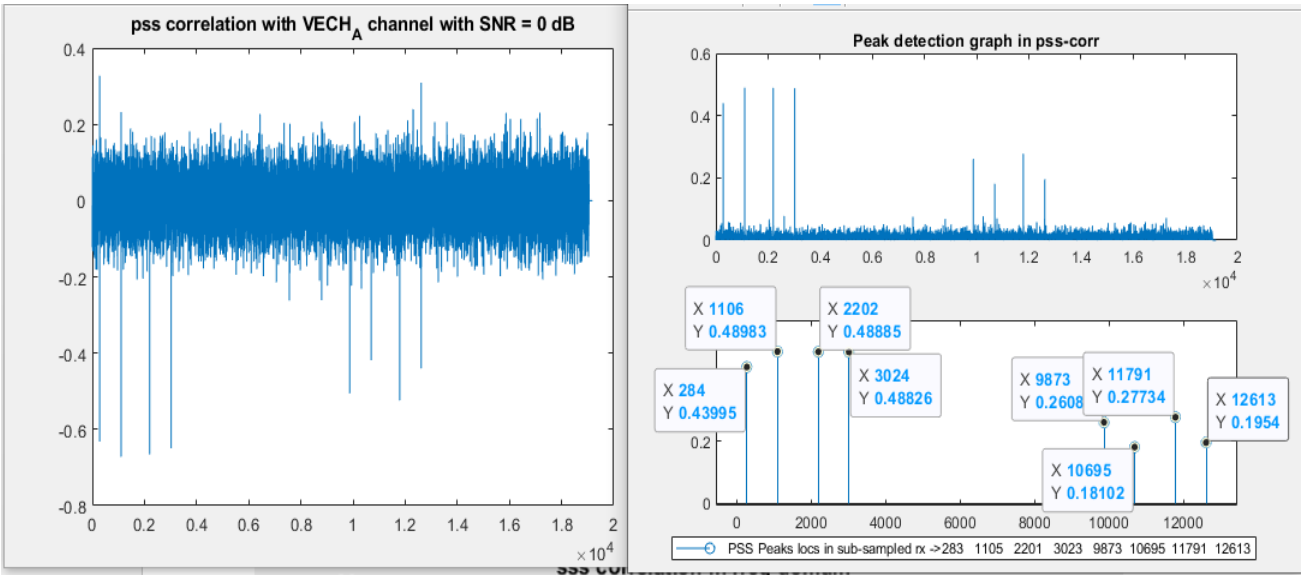


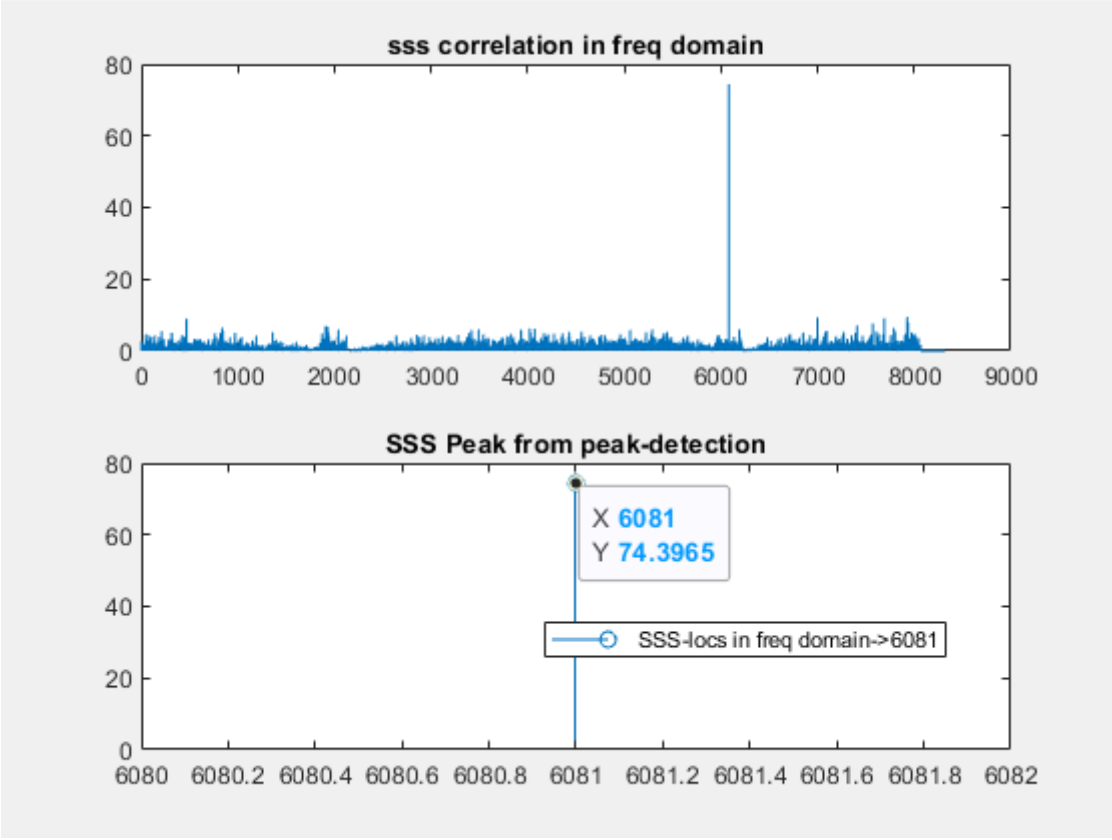
If maximum peak location in PSS detection is off from actual location, then SSS peak location will miss expected location which shown in legend box→6081.



contents

Plots with VECH-A Channel





contents

Future Scope

- Synchronization raster module can be added to find frequency domain location of SSBs'.
- In this report, I have made SSBurst for configuration L=4. Similarly it can be extended to L=8, 64 configurations[1].
- PSS and SSS detection can be made more real time.
- Simulation is carried over baseband, so, up-conversion and down-conversion modules can be added at transmitter and receiver respectively.
- Efficient filter can be designed rather than the one used here for PSS detection.

REFERENCES

[1] http://www.sharetechnote.com/html/5G/5G_FrameStructure.html#SS_PBCH_FrequencyDomainResourceAllocation

[2] Chapters 7 & 16 from 5G NR: The Next Generation Wireless Access Technology by Erik Dahlman, Stefan Parkvall, Johan Sköld.

[3] https://www.cse.wustl.edu/~jain/cse574-08/ftp/channel_model_tutorial.pdf

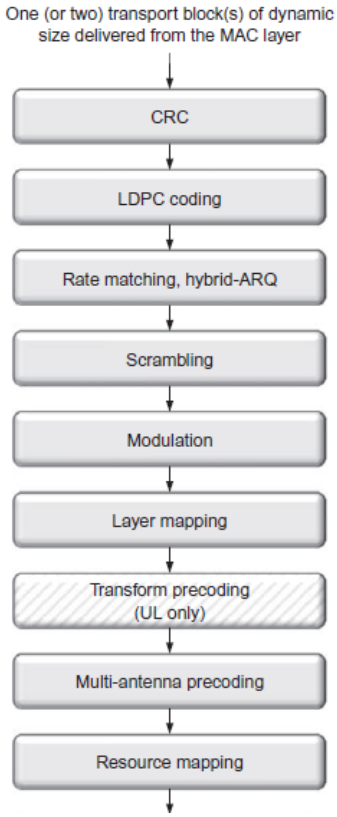
[4] https://nptel.ac.in/courses/117104099/12

contents

2. 5G NR PHY Layer in transmitter and Receiver

Introduction

In transmitter (**tx**) data from MAC layer is fed to PHY layer in units called transport blocks. Transport block is modified by various modules present in PHY layer before transmission. Wireless Channel corrupts this transmitted serial IQ data by noise, multipath and Doppler effects. PHY layer at receiver (**rx**) performs it's operations through same modules but in reverse order. Refer to below image for module operations carried in PHY layer of transmitter:



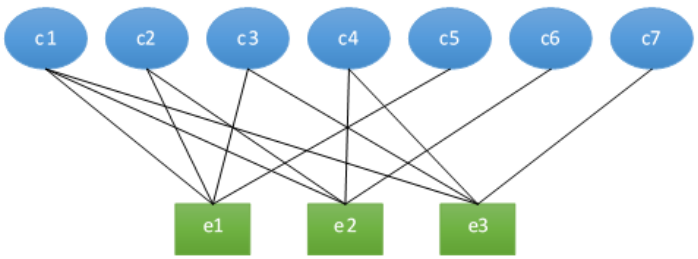
I will explain first six modules briefly followed by understanding of 5G tx & rx code written in C++ then follows integration of AWGN Channel module between tx and rx.

1. CRC Attachment:

This module computes CRC of each Transport block and attaches it at the end of block.

2. LDPC coding:

Low Density Parity Check is encoding module which uses parity for correcting channel errors. This is not usual parity bit attachment but each bit is backed up by multiple parity bits. Each error bit is connected to multiple data bits which can be visualized by graph called Tanner graph. For example,



If data and parity bits are represented by c_i {i=1, 2, 3, 4} and {i=5, 6, 7} respectively, error bits represented by e_i then the above graph can be interpreted in terms of equations given below:

$$c_1 \oplus c_2 \oplus c_3 \oplus c_5 = e_1$$

$$c_1 \oplus c_2 \oplus c_4 \oplus c_6 = e_2$$

$$c_1 \oplus c_3 \oplus c_4 \oplus c_7 = e_3$$

Parity bits should be chosen such that e_i =0 for each equation. Matrix G which operates on data stream **c** = {c_1, c_2, c_3, c_4} to generate whole codeword **v** = {c_1,…, c_7}.

$v = c * G$,where '*' operator carries out modulo 2 addition.

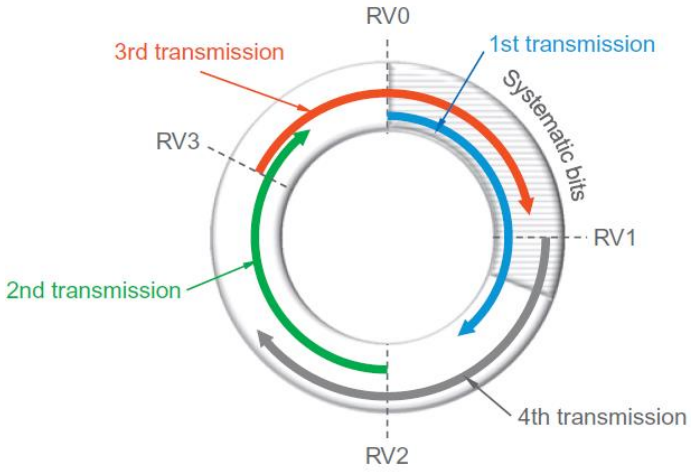
If Channel doesn't change **v**, Parity check matrix **H** (also called LDPC matrix) can check v by $H * v^T = 0$.

Matrix H is constructed from equations for example, first row of **H** is [1 1 1 0 1 0 0] as columns indicate codeword bits and a row indicates an equation. G matrix is constructed to satisfy equation: $G * H^T = 0$

This can be extended to more no. of data bits and different combinations of code digits {c_i} can be chosen to create error bits by using regular LDPC code or irregular LDPC code. Decoding is done through (Hard decision) Bit-Flipping and (Soft Decision) Message passing algorithms which are iterative in nature.

3. *Rate Matching & Hybrid ARQ:*

Rate-matching refers to the extracting a suitable fraction of the encoded data in order for the data to fit into available resources. The available resources depend on number of Resource blocks and the number of reserved elements. Rate matching is done for every code block. The resulting bits are fed into a circular buffer.
The bits to be transmitted are read from the circular buffer from starting points called revision version (RV). In case of re-transmission (hybrid-ARQ), a different set of coded bits (starting from the next RV) are sent which, when combined with previous transmission, increases coding gain.



Final step in rate-matching is interleaving of bits.

4. *Scrambler*

Scrambler module purpose is to differentiate interleaved data from interference data through **XOR** operation. Interleaved data is scrambled (XORed) with pseudo random sequence. So at receiver while de-scrambling required data could be separated from interference data.

5. *Modulation*

Modulation is conversion of bits into symbols by following certain conversion methods called Modulation schemes. Modulation schemes which are allowed in 5G NR are:

i. Pi/2-BPSK(in uplink)
ii. BPSK
iii. QPSK/4-QAM
iv. 16-QAM
v. 64-QAM
vi. 256-QAM

Modulation scheme used in simulation is 256-QAM.

Layer Mapping

Layer mapping is splitting the serial data of modulated symbols into N_L parallel arrays of data. Every n^{th} symbol maps to the nth layer. N_L varies from 1 to minimum{N_T,N_R} where N_L denotes the number of independent streams of data that can be transmitted in the channel. A single transport block can be mapped to at most 4 layers.

After Resource block allocation, **ifft( )** is performed on output of RBG Support (RB allocation) by breaking them into symbols of 4096 and adding CP (in simulation CP=0) which then, is transmitted through wireless channel. I have made changes to code for integrating AWGN channel which corrupts the data transmitted from ifftSupport (ifft). AWGNSupport is written in C++ to add AWGN Channel in simulation.

5G tx & rx code

In 5G NR transmitter, there are series of modules as explained before. These modules are written in C++ and each module has one Support module which does the job for the same. The same follows for receiver. Transmitter and receiver modules which are present in main file *NRTransmitter.cpp* are called in order. The variable *tx_ifftTimeDomainSignal* stores the output from transmitter and then hands it over to receiver fftSupport( ) module.

Attributes required for various modules are stored in C++ structure *transmit_parameters* and this structure contains other structures like *pdsch_transmitter_inparam , pdsch_transmitter_outparam,* etc,. Receiver contains similar parameters for processing received data.

AWGN CHANNEL

AWGN Channel function adds complex additive white Gaussian noise to IQ data in *tx_ifftTimeDomainSignal.*

SNR Calculation

Signal to noise ratio is measure of Signal's strength (power) against noise's. Assume signal and noise values are stored in complex arrays ifft_tx and noise_awgn. SNR can be defined as:

$SNR = \frac{E_S * R_S}{N_0 * BW}$ where E_s is signal energy given by $\sum_i((ifft(i).real)^2 + (ifft(i).imag)^2)$, noise energy E_n = N_0*BW is given by $\sum_i((noise\_awgn(i).real)^2 + (noise\_awgn(i).imag))^2$, BW is bandwidth of signal , R_s symbol rate and N_0 is power spectral density of complex Gaussian noise.

contents

Code for AWGN

Pointer to structure from AWGNSupport and pointer to AWGNclass are added to transmit_parameters.

```
        Complex *ifftTimeDomainSignal;
};
struct pdsch_transmitter_outparam{
        Complex *RBG;
        Complex *timeDomainSignal;
};
struct transmit_parameters
{
        struct pdsch_transmitter_inparam *pdsch_tx_inparam;
        struct pdsch_transmitter_outparam *pdsch_tx_outparam;
        struct module_support_struct_param *modSuppStructParam;
        struct place_holder *placeHolder;
        struct awgn *noise;
        AWGNclass *AWGNCh;
};
```

AWGNSupport.h

```
#ifndef AWGN_CHANNEL_H
#define AWGN_CHANNEL_H

#include <math.h>
#include <iostream>
#include <random>

#include "../include/complex_class.h"
#include "../include/typedef.h"

using namespace std;

struct awgn{
        Complex *channel_ip;//channel input to awgn channel
        Complex *channel_op;//signal for reception
        data_i tdl;         //each symbol length
        data_i ofdmsymno;   //current symbol no.
        data_i noport;      //no. of ports
        data_i port;        //current port no.
        data_d psd_complex; //power spectral density in complex
};

class AWGNclass{
        public:
        void AWGN_Channel(awgn *noise);
};
#endif
```

C++ Structure **awgn** contains channel input, channel output, each symbol length, current symbol number which AWGNChannel is operating on, number of ports, current port number and power spectral density of complex noise. **AWGNclass** contains AWGNChannel ( ) function which is called in *NRTransmitter.cpp.*

contents

Changes made in NRTransmitter.cpp

Noise Model definition:

```
struct transmit_parameters txParam;

struct pdsch_transmitter_inparam pdsch_tx_inparam;
struct pdsch_transmitter_outparam pdsch_tx_outparam;
struct module_support_struct_param modSuppStructParam;
struct place_holder placeHolder;
struct awgn model;

txParam.noise = &model;
txParam.pdsch_tx_inparam=&pdsch_tx_inparam;
txParam.pdsch_tx_outparam=&pdsch_tx_outparam;
txParam.modSuppStructParam=&modSuppStructParam;
txParam.placeHolder=&placeHolder;
```

Noise Model declaration:

```
));
        Complex *tx_ifftTimeDomainSignal=(Complex*)calloc(pdsch_tx_inparam.RBG_noSymbol*pdsch_tx_inparam.noPort*pdsch_tx_inparam.timeDomainL
ength,sizeof(Complex));
         Complex *noise_rx = (Complex*)calloc(pdsch_tx_inparam.RBG_noSymbol*pdsch_tx_inparam.noPort*pdsch_tx_inparam.timeDomainLength,sizeof
(Complex));

        double SNRdB = 1.0;
        double psd = 1/pow(10.0, SNRdB/10.0);

        model.channel_op = noise_rx;
        model.psd_complex = psd;
        model.tdl = pdsch_tx_inparam.timeDomainLength;
        model.noport = pdsch_tx_inparam.noPort;
```

Noise Model handover to rx parameters:

```
    rx_placeHolder.rx_TimeDomainSignal=noise_rx;
    rx_placeHolder.rx_RBGrid=rx_RBGrid;
    rx_placeHolder.rx_RBGrid_dmrs_demapped=rx_RBGrid_dmrs_demapped;
    rx_placeHolder.rx_RBGrid_data_demapped=rx_RBGrid_data_demapped;
    rx_placeHolder.rx_layerMap=rx_layerMap;
    rx_placeHolder.rx_layerDeMappedStream=rx_layerDeMappedStream;
    rx_placeHolder.llrBits=&llrBits;
    rx_placeHolder.bittildaSoft=&bittildaSoft;
    rx_placeHolder.decodedBits=decodedBits;
    rx_placeHolder.rx_portRBGrid=rx_portRBGrid;

    pdsch_rx_inparam.RBG_noRB=RBG_noRB;
    pdsch_rx_inparam.RBG_noSymbol=RBG_noSymbol;
    pdsch_rx_inparam.Rate=Rate;
```

Now in *NRTransmitter.cpp* file a function containing these modules in order is called which does each module's operation. By the time it reaches ifftSupport( ) where AWGN is added to each symbol and stored in complex array of size tx_ifftTimeDomainSignal.

contents

AWGN Channel function in ifftSupport:

```
void ifftSupport(struct transmit_parameters *txParam)
{
                data_i noSymbol=txParam->pdsch_tx_inparam->RBG_noSymbol;
                data_i noPort=txParam->pdsch_tx_inparam->noPort;
                data_i timeDomainLength=txParam->pdsch_tx_inparam->timeDomainLength;
                data_i RBG_RESize=txParam->pdsch_tx_inparam->RBG_RESize;
                Complex *ifftTimeDomainSignal=txParam->placeHolder->ifftTimeDomainSignal;
                //Complex *noise_rx = txParam->noise->channel_op;

                for(data_i ofdmSymbolNo=0;ofdmSymbolNo<noSymbol;ofdmSymbolNo++)
                {
                        txParam->noise->ofdmsymno = ofdmSymbolNo;
                        for(data_i portNo=0;portNo<noPort;portNo++)
                        {
                                txParam->modSuppStructParam->ifftImplParam->symbNo=ofdmSymbolNo;
                                txParam->modSuppStructParam->ifftImplParam->x=&txParam->placeHolder->RBGrid[portNo*RBG_RESize*noSymbol];
                                txParam->modSuppStructParam->ifftImplParam->y=&ifftTimeDomainSignal[(ofdmSymbolNo*noPort*timeDomainLength)
+(portNo*timeDomainLength)];
                                txParam->pdsch_tx_inparam->fft_Implementation->ifft_generic(txParam->modSuppStructParam->ifftImplParam);
                                txParam->noise->channel_ip = &ifftTimeDomainSignal[(ofdmSymbolNo*noPort*timeDomainLength)+(portNo*timeDoma
inLength)];
                                txParam->noise->port = portNo;
                                txParam->AWGNCh->AWGN_Channel(txParam->noise);
                        }
                }
#ifdef DEBUG_ENABLED
```

channel_ip takes one ofdm symbol from ifftTimeDomainSignal and other attributes required are passed to C++ structure *awgn* in *ifftSupport( )* and *AWGN_Channel( )* is called at last. This returns one corrupted symbol to noise_rx, a complex array for which memory is allocated in *NRTransmitter.cpp*.

AWGN_Channel( ) function:

```
#include "../include/AWGNSupport.h"

void  AWGNclass::AWGN_Channel(awgn *noise){
        //noise power = BW_rx*psd
        float sigma = sqrt((noise->psd_complex)/2);
        int N = noise->tdl;
        int pN = noise->port;
        int sN = noise->ofdmsymno;
        int nP = noise->noport;
        float x,y;

        Complex *symbol_rx = &(noise->channel_op[(sN*nP*N)+(pN*N)]);
        Complex *symbol_tx = noise->channel_ip;

        default_random_engine generator;
        normal_distribution<float> distribution;
        //adding white noise to ifft_signal
        for(int i=0;i<N;i++){
                //x and y are distinct values by random
                x = distribution(generator)*sigma;
                symbol_rx[i].real = symbol_tx[i].real + x;
                y = distribution(generator)*sigma;
                symbol_rx[i].imag = symbol_tx[i].imag + y;
        }
}
```

x,y are two independent Gaussian noise values generated with sigma calculated from PSD value passed by psd_complex to *awgn* structure pointer *noise.* Thus after all iterations, noise added tx signal will be stored in noise_rx.

Run simulation

$ mkdir bin	//in main directory
$ cd bin	//next copy makefile from main directory to directory bin
$ cmake ./	
$ make –j 10	
$ cd . .	//change directory to main
$ ./myMain.run	//copy .run file from bin folder to main directory

Simulation result

BER value = 0 with no channel, that is, x=0, y=0:

```
-----given SNRdB = 10 and psd = 3.50986e-07 -----
IFFT start
normal distr sigma = 0.000418919
IFFT end

NR_tx.cpp                ....
--------------------Result------------------
No of Bits Transmitted : 12224
No of Error Free Bits Received : 12224
No of Bits Received in Error : 0
BER : 0

-----------------Timing Report---------------
Time Teken by LDPCEncoderSupport : 4455(microsec) or 0.004455(sec)
Time Taken by ScramblerSupport : 313(microsec) or 0.000313(sec)
Time Taken by ModulationSupport : 50(microsec) or 5e-05(sec)
Time Taken by LayerMapperSupport : 30(microsec) or 3e-05(sec)
Time Taken by DigitalPrecoderSupport : 529(microsec) or 0.000529(sec)
Time Taken by RBGSupport : 6737(microsec) or 0.006737(sec)
Time Taken by ifftSupport : 147942(microsec) or 0.147942(sec)
Time Taken by fftSupport : 27079(microsec) or 0.027079(sec)
Time Taken by RX_RBGSupport : 2652(microsec) or 0.002652(sec)
Time Taken by LayerDeMapperSupport : 754(microsec) or 0.000754(sec)
Time Taken by LLRDeModulationSupport : 12258(microsec) or 0.012258(sec)
Time Taken by DeScramblerSupport : 720(microsec) or 0.00072(sec)
Time Taken by LDPCDecoderSupport : 1040(microsec) or 0.00104(sec)

Total Time Taken for Transmission : 160056(microsec) or 0.160056(sec)
Total Time Taken for Reception : 44503(microsec) or 0.044503(sec)
Total Time Taken : 204559(microsec) or 0.204559(sec)

avg Es = 3.50986e-06
avg En = 0
SNRdB calculated from simulation 10log10(Es/En) = inf
```

BER values for each SNR:

For better results simulation was run multiple times meaning multiple frames being transmitted as *then NRTransmitter.cpp* simulates only one frame.

```
pchintala@vivado:~/basebandv0-master$ ./myMain.run
-----given SNRdB = 20 and psd = 48.3258 -----

no. of bits transmitted = 1222400
BER = 0.500338
avg Es = 4755.57
avg En = 48.1435
from simulation SNRdB = 10log10(Es/En) = 19.9466
pchintala@vivado:~/basebandv0-master$
```

```
pchintala@vivado:~/basebandv0-master$ ./myMain.run
-----given SNRdB = 100 and psd = 4.83258e-07 -----

no. of bits transmitted = 1222400
BER = 0.499166
avg Es = 4991.23
avg En = 4.81443e-07
from simulation SNRdB = 10log10(Es/En) = 100.157
```

Conclusion

Simulation BER results under awgn channel were unsatisfactory with given 5G transmitter and receiver code.

-CH Phani Vignan

EE15B021

Date: 23rd May 2019.

contents