Appendix H

The Design of Rake Receiver by Using VHDL Code

Program Description

Input:

- Reset signal.
- Flag signal for the 1^{st} finger.
- Frame synchronization signal for the 1^{st} finger.
- Frame synchronization signal for the 2^{nd} finger.
- Frame synchronization signal for the 3^{rd} finger.
- Clock signal.
- In-phase digital signal (6 bits).
- Quadrature digital signal (6 bits).
- Initial scrambling code (24 bits).
- Spreading factor (3 bits).
- OVSF code number (9 bits).

Output:

- Frame synchronization signal from MRC.
- Clock signal of I channel from MRC.
- Decoded I data from MRC.
- Clock signal of Q channel from MRC.
- Decoded Q data from MRC.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Use these part when test only this file %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dc_ratio = 960/15;
noise = -3; % SNR
bpfi = 9600; % Bit per frame of I channel
sfi = 4; % spreading factor of I channel
sample = 344; % Delay in sample

%%% Read Rx I & Q Multipath Rayleigh signals %%%
I = dlmread('C:\384kbps\Idata.txt');

Q = dlmread('C:\384kbps\Qdata.txt');

%% user #2 %%
%I_2 = dlmread('C:\12.2kbps\Idata.txt');
%Q_2 = dlmread('C:\12.2kbps\Qdata.txt');

%complex_form = (I-j*Q)'+(I_2-j*Q_2)'; % 2 users
complex_form = (I-j*Q)';

%%% Add noise %%%
complex_form = awgn(complex_form,noise);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Load these files and parameters when need to test BER %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Read files I&Q binary %%%
I_binary = dlmread('C:\384kbps\I_binary.txt');

Q_binary = dlmread('C:\384kbps\Q_binary.txt');

bpfq = 150;
scramb = [];
N = 10;

for n = 1:N
    scramb = [scramb c_long2];
end

OVSF_I = dlmread('C:\MATLAB6p5\work\ovsf_4_1.txt');

OVSF_Q = dlmread('C:\MATLAB6p5\work\ovsf_256_0.txt');
sfq = 256;

pilot_slot_0 = [1 1 1 1 1 0]; pilot_slot_1 = [1 0 0 1 1 0];
pilot_slot_2 = [1 0 1 1 0 0]; pilot_slot_3 = [1 0 0 1 0 0];
pilot_slot_4 = [1 1 0 1 0 1]; pilot_slot_5 = [1 1 1 1 1 0];
pilot_slot_6 = [1 1 1 1 0 0]; pilot_slot_7 = [1 1 0 1 0 0];
pilot_slot_8 = [1 0 1 1 1 0]; pilot_slot_9 = [1 1 1 1 1 1];
pilot_slot_10 = [1 0 1 1 0 1]; pilot_slot_11 = [1 1 0 1 1 1];
pilot_slot_12 = [1 1 0 1 0 0]; pilot_slot_13 = [1 0 0 1 1 1];
pilot_slot_14 = [1 0 0 1 1 1];

ref_pilot = -2*([pilot_slot_0 pilot_slot_1 pilot_slot_2
pilot_slot_3 pilot_slot_4 pilot_slot_5 pilot_slot_6 pilot_slot_7
pilot_slot_8 pilot_slot_9 pilot_slot_10 pilot_slot_11
pilot_slot_12 pilot_slot_13 pilot_slot_14])+1;

%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Rake-Finger %%% (1st)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%

%signal_path_1 = complex_form(1,path_1(1,2):sfq*bpfq*N*4); % column#2 is sample
signal_path_1 = complex_form(1,sample:sfq*bpfq*N*4);

%% Down sampling %%
down_path_1 = downsample(signal_path_1,4);
down_signal_1 = down_path_1(1,1:(N-1)*sfq*bpfq); % consider N-1 frames

%% Descrambling %%
de_scramb_1 = down_signal_1.*conj(scramb(1,1:(N-1)*sfq*bpfq));

%%%%%%%%%%%%%%%%%
%% Despreading %%
%%%%%%%%%%%%%%%%%

despread_i = []; despread_q = [];
%%% Any bit rates %%%
for ovsfi = 1:((N-1).*bpfi)
    despread_i = [OVSF_I despread_i];
end

for ovsfq = 1:((N-1).*bpfq)
    despread_q = [OVSF_Q despread_q];
end

%% for I (data) %%
I_f1 = ((de_scramb_1).*despread_i);

I_f2 = reshape(I_f1,sfi,(N-1).*bpfi);
I_despread_1 = [];

for x = 1:((N-1).*bpfi)
    I_f3 = 0;
    for a = 1:(sfi)
        I_f3 = I_f2(a,x)+I_f3;
    end
    I_test = (I_f3)/sfi;
    I_despread_1 = [I_despread_1 I_test];
end

%% for Q (control) %%
Q_f1 = ((de_scramb_1).*despread_q);

Q_f2 = reshape(Q_f1,sfq,(N-1).*bpfq);
Q_despread_1 = [];

for x = 1:((N-1).*bpfq)
    Q_f3 = 0;
    for a = 1:(sfq)
        Q_f3 = Q_f2(a,x)+Q_f3;
    end
    Q_test = (Q_f3)/sfq;
    Q_despread_1 = [Q_despread_1 Q_test];
end Q_despread_1 = Q_despread_1*-j;

%%% Demultiplex pilot %%%
pilot_signal = []; for dem = 1:(N-1)*15
    ps = Q_despread_1(1,1+(dem-1)*10:6+(dem-1)*10);
    pilot_signal = [pilot_signal ps];
end

%%% Multiplied with reference pilots %%%
ref_pilot_new = []; for y = 1:(N-1)
```

```
    ref_pilot_new = [ref_pilot_new ref_pilot];
end pilot_correlate = ref_pilot_new.*pilot_signal;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% WMSA Channel Estimator %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Summation of pilots %%%
Mp = 6; % # pilot bits
dirac = []; for dem = 1:(N-1)*15
    pilot_summ = sum(pilot_correlate(1,6*dem-5:6*dem))/Mp;
    dirac = [dirac pilot_summ];
end

%%% Estimated impulse response %%%
lamda_0 = 1; lamda_1 = 0.6; % fixed parameters
alpha(1,1) = dirac(1,1); for i = 2:dem-2
    alpha(i) = lamda_1*(dirac(i-1)+dirac(i+2))+lamda_0*(dirac(i)+dirac(i+1));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Channel Equalizer %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

vector_gen = ones(1,4);

imp = (real(alpha)-j*imag(alpha))'*vector_gen; est_imp_1 = [];
imp_slot_1 = [pilot_correlate(1,1:6) Q_despread_1(1,7:10)];

for x = 2:134
    imp_slot = [pilot_correlate(1,6*x-5:6*x) imp(x-1,:)];
    est_imp_1 = [est_imp_1 imp_slot];
end est_imp = [imp_slot_1 est_imp_1];

ch_eqq = Q_despread_1(1,1:1340).*conj(est_imp);

est_imp_i = ones(dc_ratio,1)*est_imp; est_imp_new =
reshape(est_imp_i,1,1340*dc_ratio);

ch_eqi = I_despread_1(1,1:1340*dc_ratio).*conj(est_imp_new);

%%% BER %%%
ber_i =
symerr(sign(real(ch_eqi)),I_binary(1,1:1340*dc_ratio))/(1340*dc_ratio)

ber_q = symerr(sign(real(ch_eqq)),Q_binary(1,1:1340))/(1340)
```