Weekly Report

Yuan Yao

02/28/2015

In this week, I have used Y parameter to find the transfer function of channel because I added a DC point in the “20cm” S-parameter. And Meng and I have made a model to see the final eye diagram.

Following are the codes we use:

%% differential only serdes channel simulation

z0\_t = input('Enter the reference impedance: ');

z0\_r = input('Enter the terminal impedance: ');

bit\_rate = 28e9; %bits-per-second

bit\_time = 1/bit\_rate; % second

samples\_per\_bit = 64;

sampling\_rate = bit\_rate \* samples\_per\_bit;

sample\_step = bit\_time / samples\_per\_bit;

seqlength = 10000; % length of random sequence bits

%% setup IBIS AMI model path and filename:

modelpath = 'C:\Users\younger\Documents\MATLAB\\_develop\\_chasim\Qlogic\_Cetus\_wrk';

tx\_amifilename = 'g28\_28g\_tx.ami';

rx\_amifilename = 'g28\_28g\_rx.ami';

tx\_dllfilename = 'G28\_28G\_Tx\_32.dll';

rx\_dllfilename = 'G28\_28G\_Rx\_32.dll';

tx\_amifile = fullfile(modelpath, tx\_amifilename);

rx\_amifile = fullfile(modelpath, rx\_amifilename);

tx\_dllfile = fullfile(modelpath, tx\_dllfilename);

rx\_dllfile = fullfile(modelpath, rx\_dllfilename);

addpath(modelpath)

%% get channel impulse response

addpath('C:\Users\younger\Documents\MATLAB\\_develop\\_chasim\impulse\_response\_oct2012');

addpath('C:\Users\younger\Documents\MATLAB\\_develop\\_chasim\SYZ\_conversions');

s4ppath = 'C:\Users\younger\Documents\MATLAB\\_develop\\_chasim\Qlogic\_Cetus\_wrk';

SNP1\_s4pname = 'm\_qlogic\_baker\_e049\_fc\_tx0\_shrt.s4p';

SNP1\_s4pfile = fullfile(s4ppath, SNP1\_s4pname);

SNP2\_s4pname = '20cm\_sparam.s4p';

SNP2\_s4pfile = fullfile(s4ppath, SNP2\_s4pname);

SNP3\_s4pname = 'm\_qlogic\_baker\_e049\_fc\_rx0\_shrt.s4p';

SNP3\_s4pfile = fullfile(s4ppath, SNP3\_s4pname);

SNP1\_s4pstruct = tsnpimporti(SNP1\_s4pfile);

SNP2\_s4pstruct = tsnpimporti(SNP2\_s4pfile);

SNP3\_s4pstruct = tsnpimporti(SNP3\_s4pfile);

%input pindex1 pindex2 pindex 3

%need to be chosen accorinding the pin index of the networks

pindex1 = [1 2;3 4];

pindex2 = [1 2;3 4];

pindex3 = [1 2;3 4];

% calculate DC point of S-parameter in 20cm\_sparam.s4p

Rp = 0.3556 / 0.035 \* 0.0175;

Rpg = 0.2;

R1 = z0\_t;

R2 = z0\_r;

totR = Rp + Rpg + R1 + R2;

dcpoint = zeros(4, 4);

dcpoint(1, 1) = 1 - (2 \* R1) / totR;

dcpoint(1, 2) = (2 \* sqrt(R1 \* R2)) / totR;

dcpoint(2, 1) = dcpoint(1, 2);

dcpoint(2, 2) = 1 - (2 \* R2) / totR;

dcpoint(3, 3) = dcpoint(1, 1);

dcpoint(3, 4) = dcpoint(1, 2);

dcpoint(4, 3) = dcpoint(2, 1);

dcpoint(4, 4) = dcpoint(2, 2);

% insert DC point

SNP2\_s4pstruct0 = snpinsertdc(SNP2\_s4pstruct, dcpoint);

fq = SNP2\_s4pstruct0.freqlist;

[SNPO\_conca,pindex,fq,err,errmsg] = concatenate(SNP1\_s4pstruct,SNP2\_s4pstruct0,pindex1,pindex2,fq);

[SNPO\_conca,pindex,fq,err,errmsg] = concatenate(SNPO\_conca,SNP3\_s4pstruct,pindex,pindex3,fq);

%% get channel impulse response

znp1 = s2z(SNPO\_conca);

ynp = npmatinv(znp1.data);

mv = [1 0 -1 0; 0 1 0 -1; 0.5 0 0.5 0; 0 0.5 0 0.5];

mi = [0.5 0 -0.5 0; 0 0.5 0 -0.5; 1 0 1 0; 0 1 0 1];

for i=1:2001

ym(:,:,i) = (mi \* ynp(:,:,i)) \* inv(mv);

end

ydd = ym(1:2,1:2,:);

u = [1 0;0 1];

zt = [-2\*z0\_t 0;0 -2\*z0\_r];

for i=1:2001

Htrans(:,:,i) = 2 \* inv(u - zt \* ydd(:,:,i));

end

%

Htrans1 = Htrans(2, 1, :); % channel transfer function H(jw)

hac\_t = ifft(Htrans1); % channel impulse response hAC(t)

nfp = length(fq);

fstop = fq(nfp); % in Hz

fstep = fstop / (nfp - 1); % in Hz

fspan = fstop;

tstep = 1/fspan; % in seconds

tspan = 1/fstep; % in seconds

tlist = 0:tstep:( (nfp-1) + 0.1 )\*tstep;

impresi = reshape(real(hac\_t), 1, 2001);

tinterp = 0:sample\_step:5000\*sample\_step;

[channel\_impres, e2, em2] = wavesample(tlist, impresi, tinterp);

plot(tinterp, channel\_impres,'--rs', tlist, impresi,'-.bo');

title('Impulse Response');

xlabel('Time');

ylabel('Impulse');

legend('Resampled');

grid on;

%% IBIS AMI\_Init

%[ impresotx, amiparotx, mhtx, initmsgtx, errtx, errmsgtx ] = amiinit\_1207a(tx\_amifile, tx\_dllfile, channel\_impres, sample\_step, bit\_time);

amipari = txt2str( tx\_amifile );

if isrow(channel\_impres)

channel\_impres = channel\_impres';

end

[dllpath, dllname, dllext] = fileparts(tx\_dllfile);

addpath(dllpath);

lload = loadlibrary(tx\_dllfile);

isloadedok = libisloaded(dllname);

libfun = libfunctions(dllname, '-full');

[row\_size,aggressors] = size(channel\_impres);

amiparo\_ptrptr = libpointer('stringPtrPtr', {''});

amimemhdl\_ptrptr = libpointer('voidPtrPtr', []);

amimsg\_ptrptr = libpointer('stringPtrPtr', {''});

[ret0, impresotx, pario, amiparo, mhtx, amimsg ] = ...

calllib(dllname, 'AMI\_Init', channel\_impres, row\_size, aggressors-1, sample\_step, bit\_time, ...

amipari, amiparo\_ptrptr, amimemhdl\_ptrptr, amimsg\_ptrptr);

% [long, doublePtr, cstring, stringPtrPtr, voidPtrPtr, stringPtrPtr

pauseanchor=1;

plot(tinterp, channel\_impres,'--rs', tinterp, impresotx','-.bo');

title('Impulse Response');

xlabel('Time');

ylabel('Impulse');

legend('Whatever');

grid on;

% [ impresorx, amiparorx, mhrx, initmsgrx, errrx, errmsgrx ] = amiinit\_1207a(rx\_amifile, rx\_dllfile, impresotx, sample\_step, bit\_time);

amipari = txt2str( rx\_amifile );

if isrow(impresotx)

impresotx = impresotx';

end

[dllpath, dllname, dllext] = fileparts(rx\_dllfile);

addpath(dllpath);

lload = loadlibrary(rx\_dllfile);

isloadedok = libisloaded(dllname);

libfun = libfunctions(dllname, '-full');

[row\_size,aggressors] = size(impresotx);

amiparo\_ptrptr1 = libpointer('stringPtrPtr', {''});

amimemhdl\_ptrptr1 = libpointer('voidPtrPtr', []);

amimsg\_ptrptr1 = libpointer('stringPtrPtr', {''});

[ret0, impresorx, pario, amiparo, mhrx, amimsg ] = ...

calllib(dllname, 'AMI\_Init', impresotx, row\_size, aggressors-1, sample\_step, bit\_time, ...

amipari, amiparo\_ptrptr1, amimemhdl\_ptrptr1, amimsg\_ptrptr1);

% [long, doublePtr, cstring, stringPtrPtr, voidPtrPtr, stringPtrPtr

pauseanchor=1;

plot(tinterp, channel\_impres,'--rs', tinterp, impresotx','-.bo', tinterp, (impresorx - 0.05)',':md');

title('Impulse Response');

xlabel('Time');

ylabel('Impulse');

legend('Modified');

grid on;

% IBIS AMI\_GetWave, Tx

seqi = prbs1(seqlength);

[wavei,sample\_step\_txgw, tlistt, etxgw,emtxgw] = bnrzmod(seqi, bit\_rate, samples\_per\_bit);

wave\_time = tlistt;

clk\_count = seqlength + 2;

clki = rand(1, clk\_count) \* seqlength \* bit\_time;

[ waveo, clko, amiparo, amimemhdl, err, errmsg ] = amigetwave\_1208a(tx\_dllfile, mhtx, wavei, clki);

plot(wave\_time, wavei,'--rs', wave\_time, waveo,'-.bo');

title('Impulse Response');

xlabel('Time');

ylabel('Impulse');

%legend('Whatever');

grid on;

%% convolution with channel impulse response

wave\_rxin = conv(waveo, channel\_impres');

wave\_time\_rxin = ( 0: (length(wave\_rxin)-1) ) \* sample\_step\_txgw;

plot(wave\_time, wavei,'--ms', wave\_time, waveo,'--rd', wave\_time\_rxin, wave\_rxin\*0.1,'-.bo');

title('Impulse Response');

xlabel('Time');

ylabel('Impulse');

%legend('Whatever');

grid on;

%% IBIS AMI\_GetWave, Rx

seqlength\_rxin = length(wave\_rxin) / samples\_per\_bit ;

clk\_count = int64(seqlength\_rxin + 10);

clki = rand(1, clk\_count) \* seqlength\_rxin \* bit\_time;

[ wave\_rxo, clk\_rxo, amipar\_rxo, amimemhdl\_rxo, errrxo, errmsgrxo ] = amigetwave\_1208a(rx\_dllfile, mhrx, wave\_rxin\*0.1, clki);

%plot(wave\_time, wavei,'--ms', wave\_time, waveo,'--rd', wave\_time\_rxin, wave\_rxin,'-.bo');

%plot( wave\_time, waveo,'--rd', wave\_time\_rxin, wave\_rxin,'-.bo', wave\_time\_rxin, wave\_rxo,'-.ms');

% % plot( wave\_time\_rxin, wave\_rxin\*0.1,'-.bo', wave\_time\_rxin, wave\_rxo,'-.ms');

% % title('Impulse Response');

% % xlabel('Time');

% % ylabel('Impulse');

% % legend('Whatever');

% % grid on;

clf;

wave\_time\_rxin\_12 = wave\_time\_rxin(2000:(2000+samples\_per\_bit\*100));

wave\_rxin\_12 = wave\_rxin(2000:(2000+samples\_per\_bit\*100));

wave\_rxo\_12 = wave\_rxo(2000:(2000+samples\_per\_bit\*100));

plot( wave\_time\_rxin\_12, wave\_rxin\_12\*0.1,'-.bo', wave\_time\_rxin\_12, wave\_rxo\_12,'-.ms');

title('Impulse Response');

xlabel('Time');

ylabel('Impulse');

%legend('Whatever');

grid on;

h = eyediagram(wave\_rxo(2000:(2000+samples\_per\_bit\*100)), 2\*samples\_per\_bit);

pausehere=1;

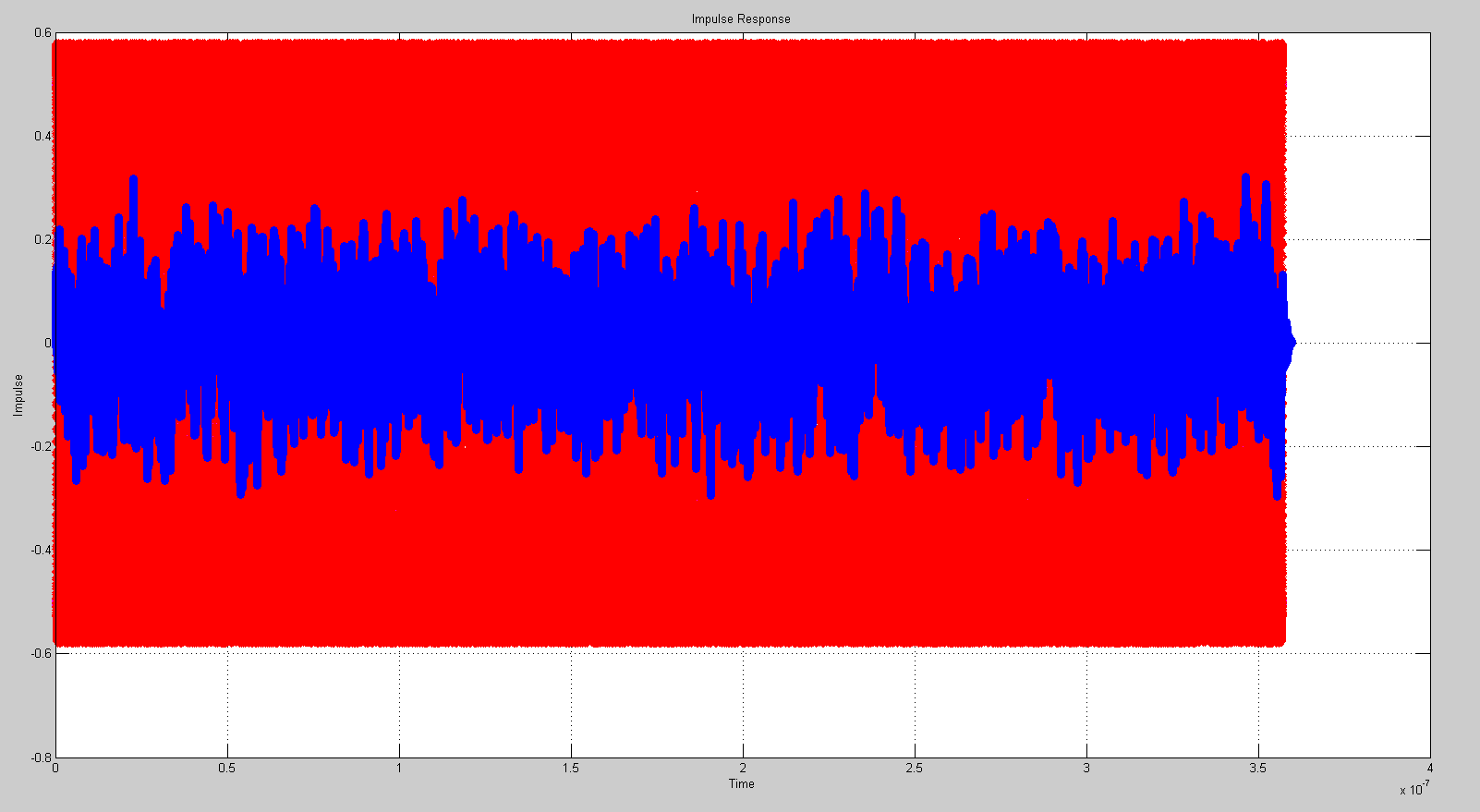
Last time we have trouble using AMI\_Init function, now we can use it by dragging out the codes in the function instead of calling AMI\_Init function. However it can only run once. When I run it twice, My MATLAB will crash down. I think it may because I haven’t cleared all the pointers but I don’t know how to do that.

And following are the graphs our codes run out.

This is the channel impulse response channel\_impres. It’s not smooth enough and may because I haven’t interpolated the hac\_t properly.

This is the output of TX AMI\_Init. I see it a bad shape because my impulse response is not smooth.

This is the output of RX AMI\_Init.

This is the convolution of x(t) (after AMI\_Getwave) and channel\_impres.

This is the final eye diagram.

This is the final impulse response.

The biggest problem for now is how to get the correct channel impulse response. I’m looking at the reference that James wrote but I have some trouble understanding how to make time correspond with frequency. I think when I figure it out, I will get the correct impulse response.