COM 2.51 with rxFFE updates

Richard Mellitz, Samtec

Oct. 3, 2018

Table of Contents

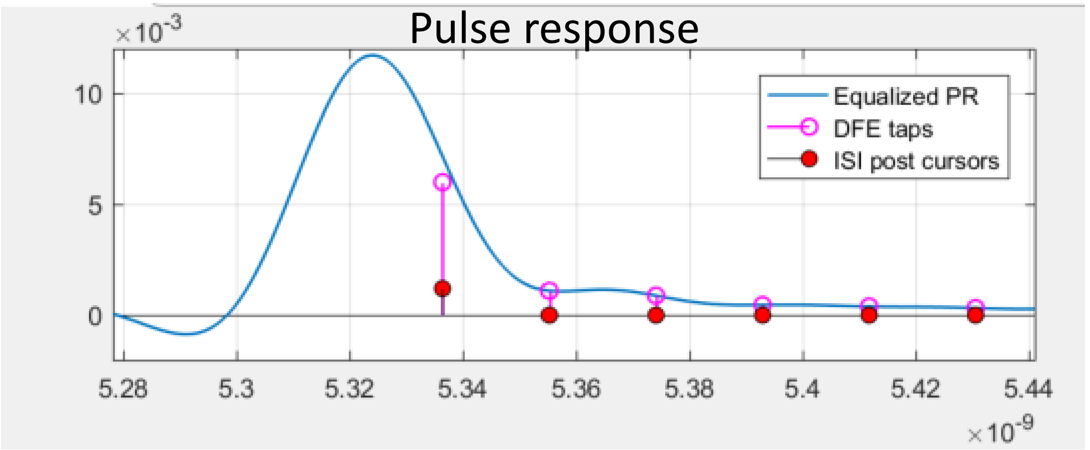
COM 2.51 may be used to investigate 2 Signal Architectures

- ❑ Zero Forced DFE (Annex 93A) … No change
- ❑ One DFE tap and a number of (Rx)FFE taps
  - FFE tap adjustments, algorithm modifications, and index corrections added for COM 2.51

Zero Forced DFE (Annex 93A) … No change

- ❑ Same as Clause 93A COM
- ❑ Review presented in mellitz_3ck_01_0718 (slide 5)

Example where 1st DFE tap reach limit creating ISI noise



$$FOM = 10\log_{10}\left(\frac{A_s^2}{\sigma_{TX}^2 + \sigma_{ISI}^2 + \sigma_J^2 + \sigma_{XT}^2 + \sigma_N^2}\right) \quad (93A–36)$$

The FOM is calculated for each permitted combination of $c(-1)$, $c(1)$, and g_{DC} values per Table 93A–1. The combination of values that maximizes the FOM, including the corresponding value of t_s, is used for the calculation of the interference and noise amplitude in 93A.1.7 and the calculation of COM in 93A.1.

http://www.ieee802.org/3/ck/public/18_07/mellitz_3ck_01_0718.pdf

One DFE tap and a number of (Rx)FFE taps

- ❑ Vector forcing algorithm to determine equalization settings
- ❑ Review presented in mellitz_3ck_01_0718 (slide 5)
- ❑ Does not necessarily resemble a receiver

Evaluation COM reference Model with Rx FFE

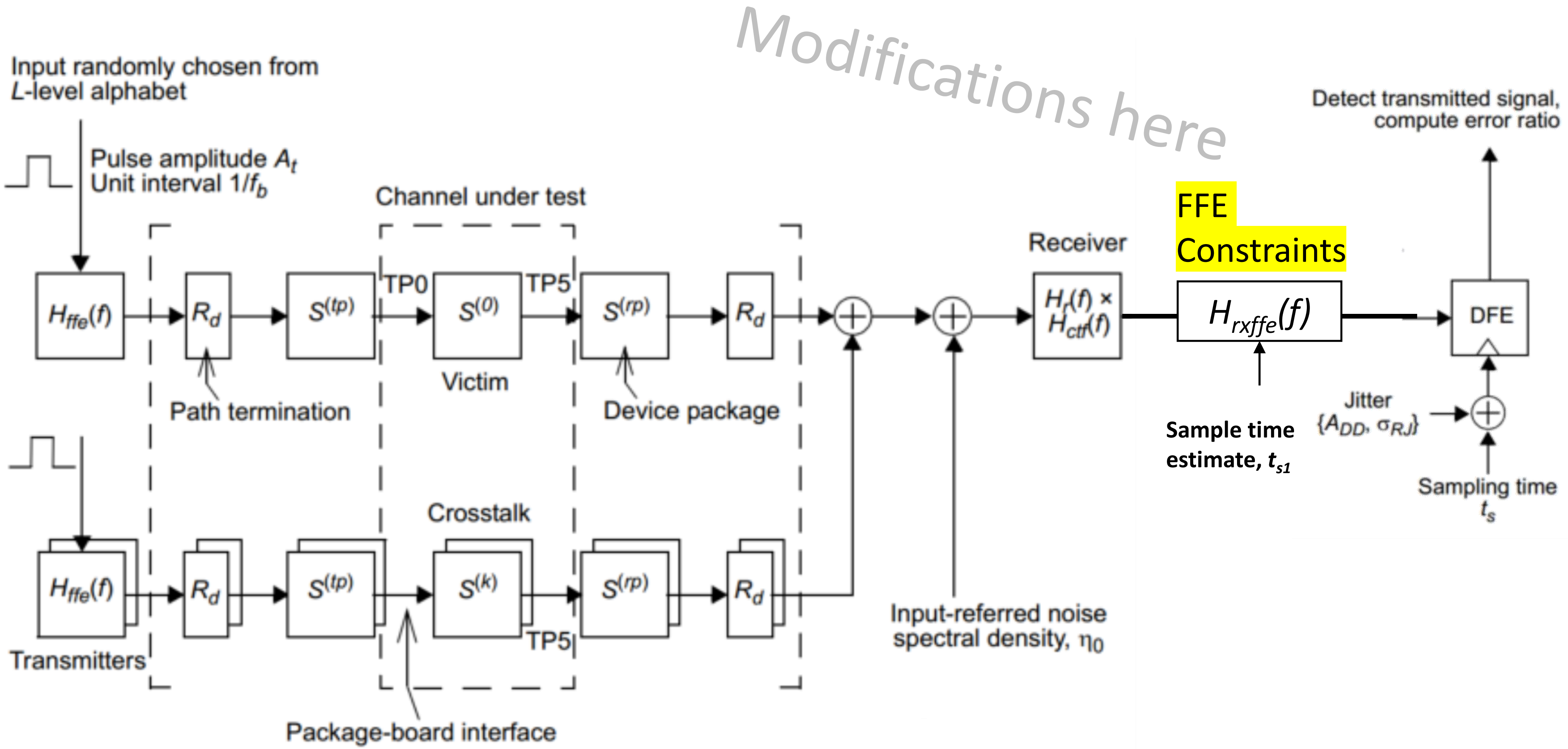


Figure 93A–1—COM reference model

[* http://www.ieee802.org/3/ck/public/18_07/mellitz_3ck_01_0718.pdf](http://www.ieee802.org/3/ck/public/18_07/mellitz_3ck_01_0718.pdf)

COM is based on the pulse response (Annex 93A)

- Thru (ISI) channel response is $h^{(0)}(t)$ i.e. the pulse response

The pulse response $h^{(k)}(t)$ is derived from the voltage transfer function $H^{(k)}(f)$ (see 93A.1.4) using Equation (93A–24).

$$h^{(k)}(t) = \int_{-\infty}^{\infty} X(f)H^{(k)}(f)\exp(j2\pi ft)dt \quad (93A\text{–}24)$$

Same as mellitz_3ck_01_0718

- The following uses pulse response plots to describe COM equalization
- Best FOM for full grid searching for all ctf and Tx ffe values determines setting for the ctf and Tx FFE to compute COM

Adding the long FFE with DFE in COM 2.51

Same full grid as for zero forced DFE except compute C, Hisi, and FOM for each grid setting

- ❑ Find the Rx FFE taps settings, C , with LMS vector force method
- ❑ Readjust sample point
- ❑ Apply C to form a new Hisi to be used to compute a the FOM
  - With noise terms

$$FOM = 10\log_{10}\left(\frac{A_s^2}{\sigma_{TX}^2+\sigma_{ISI}^2+\sigma_J^2+\sigma_{XT}^2+\sigma_N^2}\right)$$

- ❑ Settings with be best FOM are used to compute COM

Next… how to find C

Same as mellitz_3ck_01_0718

Determining FFE taps, C within the inside loop

- $C=((HH^T*HH^{-1}*HH^T)^T*FV^T$
  - C are the Rx FFE taps
  - HH is derived from $h^{(0)}(t)$
  - HH is shifted sampled ISI matrix
- FV is the forcing vector ,
  - FV = [... 0, 0,FV0,FV1, 0, 0, 0, 0...]
- FV for the cursor tap is
  - FV0= $h^{(0)}(t_s)$
  - This forces the cursor tap to 1
- ☞ Modified from mellitz_3ck_01_0718:
  FV for the post cursor tap (2.51 update)
  - FV1 = sign($h^{(0)}(t_s+T_b)$) $min(|h^{(0)}(t_s+T_b)|, |b_1 h^{(0)}(t_s)|)$
  - This makes sure the b_1 is not violated for the DFE
- $h_{fferx}(f)$ is computed from the C found as in eq 93A-21

NEW: Adjust C with an inside loop

- ❑ H_{isi} is the resampled (1 UI or T_b spaced) pulse response
- ❑ Apply the Rx FFE with tap values C to H_{isi}
  - Shift, multiply add method
  - This creates H_{isi}_filtered (filtered pulse response)
- ❑ Problem: late cursors taps near reflections in the PR are too strong. Data suggest the less number of taps performs better.
  - Solution: just use less number taps for the solution
- ❑ Determine an interim FOM for H_{isi}_filtered by dividing the cursor value by the root sum square (RSS) of all the other values and converting to dB
- ❑ Incrementally remove C taps starting with last tap, and compute a new H_{isi}_filtered and FOM. The code only goes back 4 taps now
- ❑ Use the C with best interim FOM
- ❑ Continue with the grid loop to determine original FOM with eq. 93A-36

Additional Inner loop constraints an settings for evaluation

ffe_pre_tap_len	**3**
ffe_post_tap_len	16

If both set to zero Rx FFE computation is eliminated and default back to original COM method

ffe_tap_step_size	0.01
ffe_main_cursor_min	0.7
ffe_pre_tap1_max	0.3
ffe_post_tap1_max	0.3
ffe_tapn_max	0.125

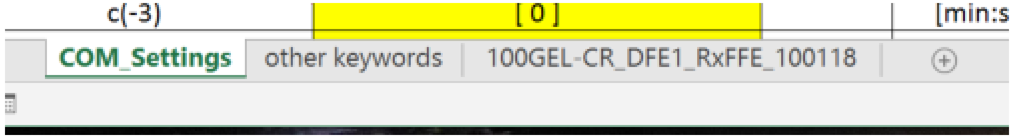
If set to zero, taps are not quantized

These just break the loop for now.

Included in ZIP

- config_com_ieee8023_93a=100GEL-CR_DFE_100118.xls

- config_com_ieee8023_93a=100GEL-KR_DFE_100118.xls



- com_ieee8023_93a_251.m

Backup: Computing HH, shifted sampled ISI matrix, force() C code

Start with Pulse Response, and Resample

Let the pulse response be $h^{(0)}(t)$ by

Apply $H_r(f)$, $H_t(f)$, $H_{CFT}(f)$ and $H_{FFE}(f)$ setting to $H_{21}^{(0)}(f)$

Find sample point ts and resample as hisi(n)

Example pulse response for 20 UI

```
hisi=[ hisi1, hisi2, hisi3, hisi4, hisi5, hisi6, hisi7, hisi8, hisi9, hisi10,
hisi11, hisi12, hisi13, hisi14, hisi15, hisi16, hisi17, hisi18, hisi19, hisi20]
```

Example: Let hisi(9) correspond to the sample point

Example; 2 pre cursors, 5 post cursors

C is the set of cursors (c1… cn)

Zero pad hsis in preparation for circshift function

[0, 0, hisi1, hisi2, hisi3, hisi4, hisi5, hisi6, hisi7, hisi8, hisi9, hisi10, hisi11, hisi12, hisi13, hisi14, hisi15, hisi16, hisi17, hisi18, hisi19, hisi20, 0, 0, 0, 0, 0]

Find C with Vector Forcing LMS

Define HH array of shifted hisi vectors: HH =

```
[ hisi9, hisi10, hisi11, hisi12, hisi13, hisi14, hisi15, hisi16, hisi17]
[ hisi8,  hisi9, hisi10, hisi11, hisi12, hisi13, hisi14, hisi15, hisi16]
[ hisi7,  hisi8,  hisi9, hisi10, hisi11, hisi12, hisi13, hisi14, hisi15]
[ hisi6,  hisi7,  hisi8,  hisi9, hisi10, hisi11, hisi12, hisi13, hisi14]
[ hisi5,  hisi6,  hisi7,  hisi8,  hisi9, hisi10, hisi11, hisi12, hisi13]
[ hisi4,  hisi5,  hisi6,  hisi7,  hisi8,  hisi9, hisi10, hisi11, hisi12]
[ hisi3,  hisi4,  hisi5,  hisi6,  hisi7,  hisi8,  hisi9, hisi10, hisi11]
[ hisi2,  hisi3,  hisi4,  hisi5,  hisi6,  hisi7,  hisi8,  hisi9, hisi10]
```

FV is the forcing vector , FV =

```
[ 0, 0, FV0, FV1, 0, 0, 0]
```

Such that

```
FV =HH.'*C.'
```

And we solve for C

```
C=((HH'*HH)^-1*HH')'*FV';
```

force

```
function [ Vfiltered, Cmod ] = force( V ,param, OP , ix, C)
% Vfilter is vector forced filtered sbr
% Cmod is the ffe tap co-efficient vector
% if C is passed, just process V with C else compute C
% cmx=param.rx_cmx; number of pre cursor taps
% cpx=param.rx_cps; number of post cursor taps
% V=sbr; pass pulse response
% ix the sample point in the passed pulse response
% the sample point is recomputed by optimize_fom
% OP not used for now
% test with load('SBR_FIR_resp.mydata','-mat')
%% set up parameters --------------------------------------------
----------
if ~exist('cursor_gain','var'), cursor_gain=0; end
csm = @(b,n) [ b((length(b)-n)+1:length(b)) b(1:(length(b)-n))];
% minus circshift
csp = @(b,n) [ b(n:length(b)) b(1:n-1) ]; %  plus circshift
if ~exist('ix','var'),ix=find(V==max(V),1,'first');end
cmx=param.RxFFE_cmx;
cpx=param.RxFFE_cpx;
cstep=param.RxFFE_stepz;
ndfe=param.ndfe;
spui=param.samples_per_ui;
%% resample to align with sample point ---------------------------
----------
```

```
if ix < length(V);
    if isrow(V)
if mod(ix,spui) == 0
            vsampled_raw =
[V(spui+mod(ix,spui):spui:(mod(ix,spui)+spui*(floor(ix/spui)-
1)))'; V(ix:spui:end)'];
        else
            vsampled_raw =
[V(mod(ix,spui):spui:(mod(ix,spui)+spui*(floor(ix/spui)-1)))';
V(ix:spui:end)'];
        end

    else
if mod(ix,spui) == 0
            vsampled_raw =
[V(spui+mod(ix,spui):spui:(mod(ix,spui)+spui*(floor(ix/spui)-
1))); V(ix:spui:end)];
        else
            vsampled_raw =
[V(mod(ix,spui):spui:(mod(ix,spui)+spui*(floor(ix/spui)-1)));
V(ix:spui:end)];
        end
    end
else
    if isrow(V)
        vsampled_raw = V(mod(ix,spui):spui:end)';
    else
        vsampled_raw = V(mod(ix,spui):spui:end) ;
    end
end
```

Force ‘cont’d

```
vsampled=[zeros(1,cmx) vsampled_raw' zeros(1,cpx)];% pad for pre
and post cursor prior to shifting
%% create VV matrix of shifted UI spaced sample of the pulse
response
ishift=cmx+1;
for i=1:cmx
    ishift=ishift-1;
    VV(i,:)=circshift(vsampled,[0,-ishift]);
end
VV(cmx+1,:)=vsampled;
ishift=0;
for i=1:cpx
    ishift=ishift+1;
    VV(i+cmx+1,:)=circshift(vsampled,[0,ishift]);
end
% find the index for the sample point but in the UI resample
vector, vsampled
ivs=find(vsampled==V(ix),1,'first');% ivs is the sample point for
V
if ~exist('C','var')
    % cmx+1 is the cursor or sample point
    VV=VV(:,ivs-cmx:ivs+cpx); % only consider the VV matrix that
correstonds to the FFE taps
    FV=zeros(1,cmx+cpx+1); % zero the forceing veotor, FV first
    FV(cmx+1)=vsampled(ivs)*10^(param.current_ffegain/20); %
force the voltage at sample point
    if param.ndfe~=0
FV(cmx+2)=min(param.bmax(1)*FV(cmx+1),abs(vsampled(ivs+1)))*sign(
vsampled(ivs+1));
    end
    C=((VV'*VV)^-1*VV')'*FV'; % solve for FFE taps, C
```

```
if cstep ~= 0
C=C/C(cmx+1); % r241 make cursor tap 1
        Cmod=floor(abs(C/cstep)).*sign(C)*cstep;% r250 quantize
with floor ad sign(C)
    else
%           C=C/sum(C); % r240
        Cmod=C/C(cmx+1); % r241 constrain taps to sum of taps = 1
    end
    Cmod=Cmod(1:cmx+1+cpx);
else
    Cmod=C;%just us the FFE taps, C, passed for filtering
end
fom_ffe=0;
Cmod1=Cmod;
for i=0:4
    Cmodtest=[ Cmod(1:1+cpx+cmx-i).' zeros( 1,i) ].';
    Vtest=FFE(Cmodtest , param.RxFFE_cmx,1, vsampled );
    Vs=Vtest(ivs);
    Vtest(ivs)=0;
    fom_ffe_test= db( Vs/norm(Vtest));
    if abs(fom_ffe_test) > abs(fom_ffe)
        fom_ffe=fom_ffe_test;
        Cmod1=Cmodtest; % modify Cmod1
    end
end
Cmod=Cmod1;
%%
%% filter the pulse response with the solved FFE
Vfiltered=FFE( Cmod , param.RxFFE_cmx,spui, V );
```