Behavioral Simulation of a High-Speed Link Transceiver Using VppSim

6.22s High-Speed I/O Design Techniques

Sanquan Song, Vladimir Stojanovic
July 12, 2008

Table of Contents

Introduction

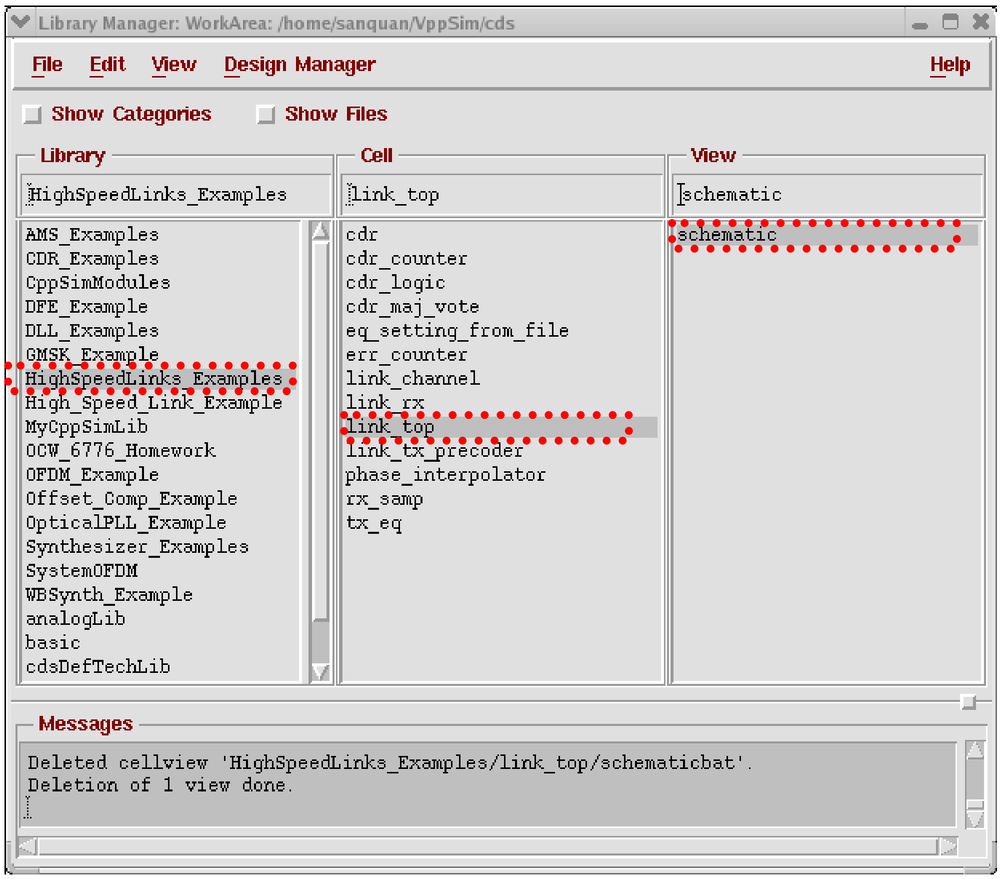
Pre-emphasis, clock-data-recovery (CDR) and decision-feed-back (DFE) equalization are the keys techniques for high-speed-links. Since tutorial *Behavioral Simulation of Decision Feedback Equalizer Architectures Using VppSim* covers DFE, this tutorial focuses on the modeling and simulation of pre-emphasis and clock-data-recovery. The simulation setup includes:

• Transmitter:	Pre-emphasis technique
• Channel:	FIR filtering with impulse response derived from measurement
• Receiver:	Data Sampler with CDR
• Support block:	Error Counter

Preliminaries

A. Opening Cadence Schematics

Select the **HighSpeedLinks_Examples** library from the **Cadence Library Manager**. The library should now look as follows:

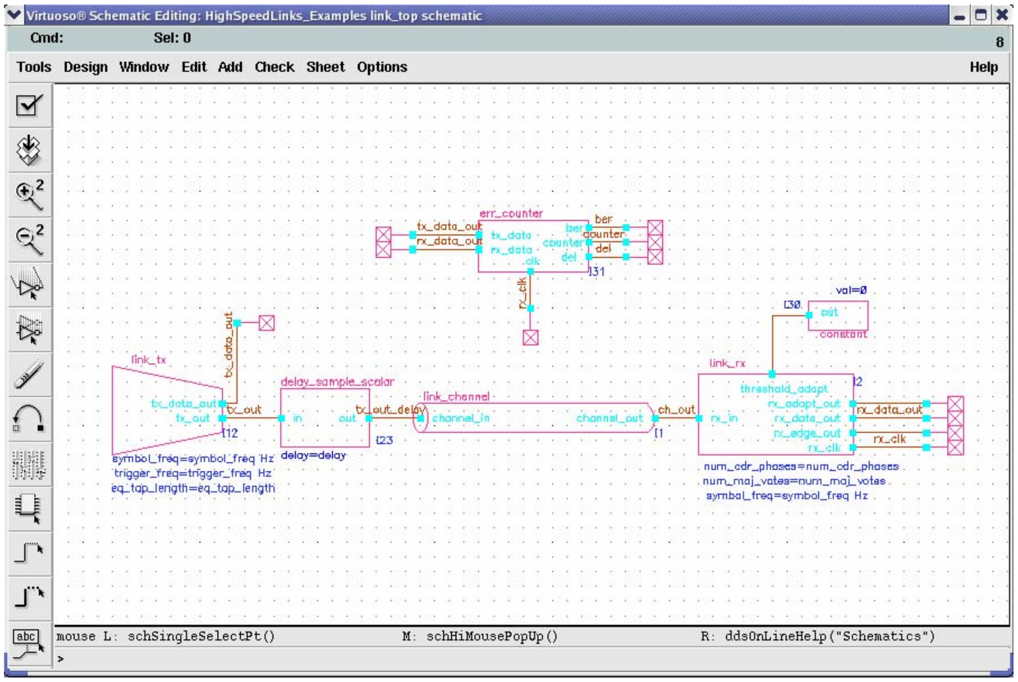


Library: HighSpeedLinks_Examples
Cell: link_top
View: schematic

```
AMS_Examples               cdr                     schematic
CDR_Examples               cdr_counter
CppSimModules              cdr_logic
DFE_Example                cdr_maj_vote
DLL_Examples               eq_setting_from_file
GMSK_Example               err_counter
HighSpeedLinks_Examples    link_channel
High_Speed_Link_Example    link_rx
MyCppSimLib                link_top
OCW_6776_Homework          link_tx_precoder
OFDM_Example               phase_interpolator
Offset_Comp_Example        rx_samp
OpticalPLL_Example         tx_eq
Synthesizer_Examples
SystemOFDM
WBSynth_Example
analogLib
basic
cdsDefTechLib
```

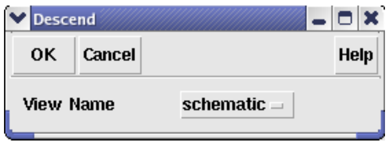
```
Deleted cellview 'HighSpeedLinks_Examples/link_top/schematicbat'.
Deletion of 1 view done.
```

The schematic is shown below:


Virtuoso® Schematic Editing: HighSpeedLinks_Examples link_top schematic
Cmd:
Sel: 0
Tools Design Window Edit Add Check Sheet Options Help
err_counter
ber
link_tx
delay_sample_scalar
link_channel
link_rx
threshold_adapt
val=0
constant
symbol_freq=symbol_freq Hz
trigger_freq=trigger_freq Hz
eq_tap_length=eq_tap_length
delay=delay
num_cdr_phases=num_cdr_phases
num_maj_votes=num_maj_votes
symbol_freq=symbol_freq Hz

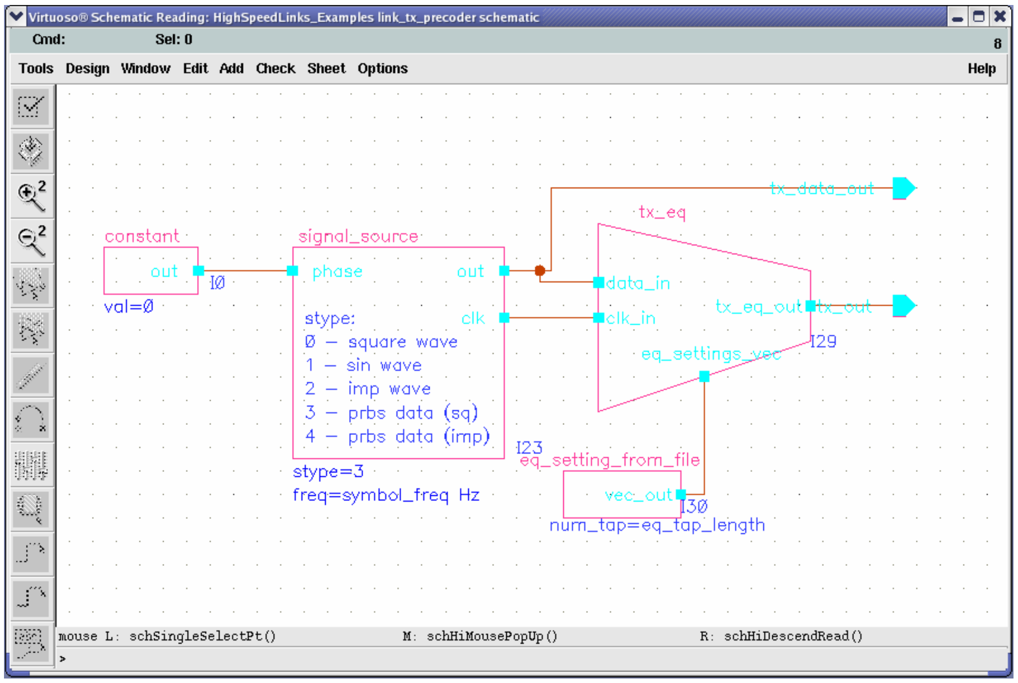
```
mouse L: schSingleSelectPt()          M: schHiMousePopUp()          R: ddsOnLineHelp("Schematics")
```

Select the **link_tx** icon within the above schematic, and then press **e** to descend down into the associated schematic. The following box will pop up:



View Name schematic

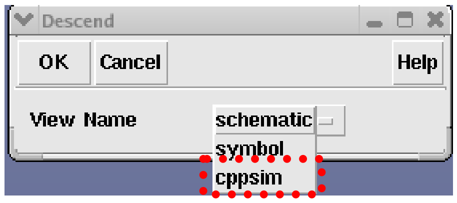
Press **OK** to see the schematic of the transmitter:



```
mouse L: schSingleSelectPt()          M: schHiMousePopUp()          R: schHiDescendRead()
```

The transmitter includes a PRBS data generator (**signal_source**) and a pre-emphasis (FIR) filter (**eq_setting-from_file** and **tx_eq**). The matlab script *transceiver_testBench.m* (saved in …/SimRuns/HighSpeedLinks_Examples/link_top) generates the impulse response from the measurement into file *link_channel.dat* and derives the pre-emphasis taps into file *eq_taps.dat*. Cell **eq_setting-from_file** loads the pre-empasis coefficients from the file *eq_taps.dat* and feed it into the FIR filter **tx_eq**.

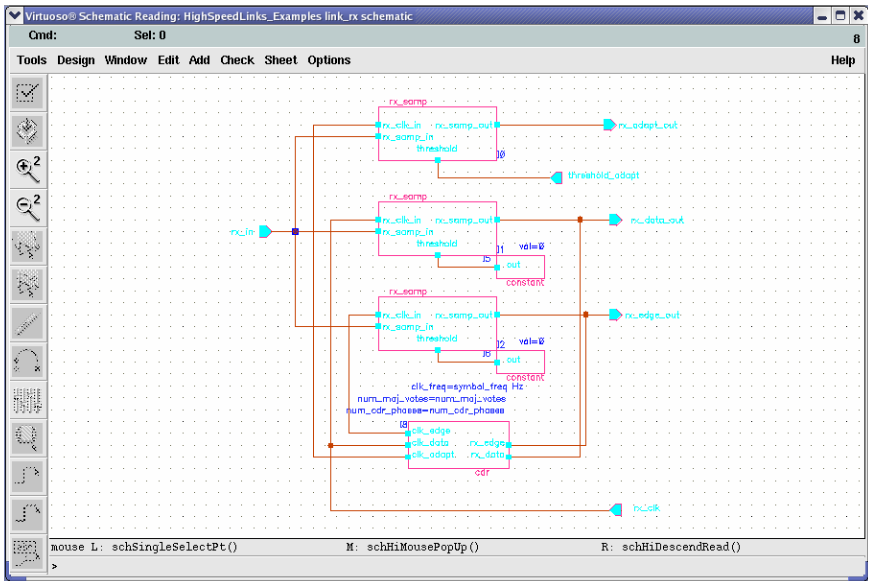
Press **Ctrl+e** to return to the top level. Select **link_channel**, press key **e** and select *CppSim* to check its code:



```
module: link_channel

description:  loads the channel impulse response from link_channel.dat and
convolves it with incomming data stream - time step is Ts for now
- although could try using lower sampling rate since bandlimited channel
and then interpolate at the Rx

parameters:
inputs: double channel_in
outputs:  double channel_out
static_variables:
classes: List channel_taps() Filter filt_ch("1","1")

init:

    channel_taps.load("link_channel.dat");
    filt_ch.set(channel_taps,"1");
    filt_ch.reset(0.0);

code:
   filt_ch.inp(channel_in);
   channel_out=filt_ch.out;
```

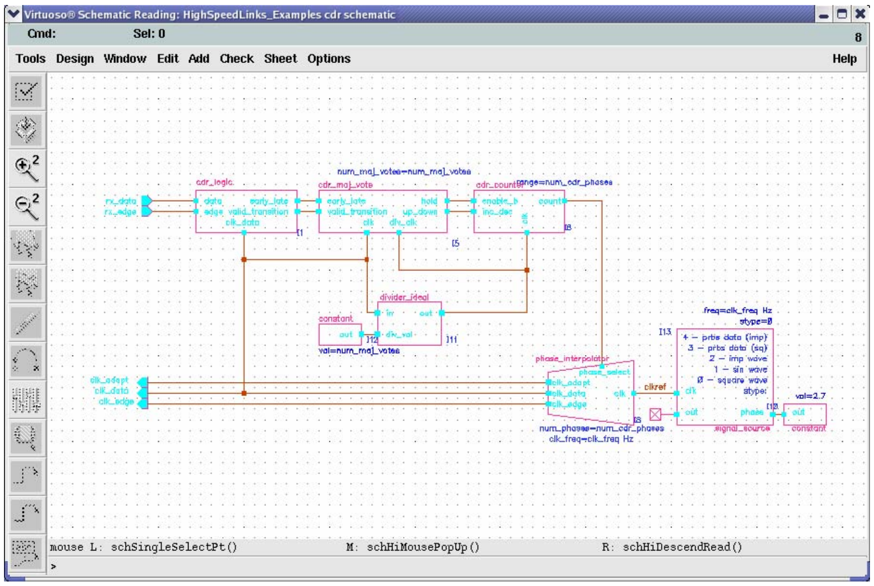
Select **link_rx** cell and press key **e** to see the details of the receiver. It includes three components:

1. Data sampler
2. Edge sampler
3. Clock Data Recovery Cell



```
mouse L: schSingleSelectPt()          M: schHiMousePopUp()          R: schHiDescendRead()
>
```

Select **cdr** cell and press key **e** to see the details of the Alexander-type clock-data-recovery cell:

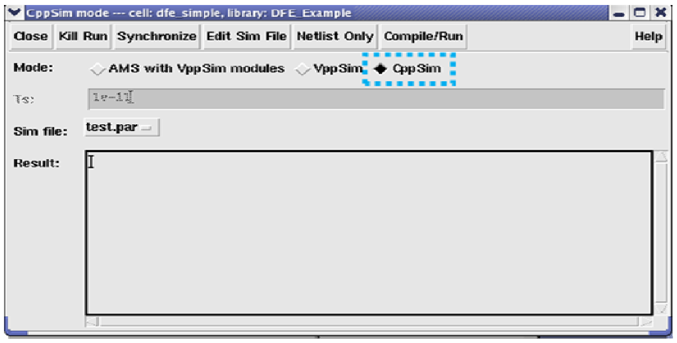


```
mouse L: schSingleSelectPt()          M: schHiMousePopUp()          R: schHiDescendRead()
>
```

The CDR collects the statistic of the signs of the data samples and *valid* edges samples. If these signs are prone to be different, the phase of the PLL output will be increased. Otherwise, it will be decreased until they are independent.
Then press **Ctrl+e** twice to return to the top level.

B. Running VppSim Simulations

Click **Options→ VppSim** to open the **VppSim GUI**.
Click **CppSim** radio button within the **VppSim GUI**.



Mode: AMS with VppSim modules VppSim [✓ CppSim]

Ts: 1e-11

Sim file: test.par

Result:

Click **Edit Sim File** to check the test.par file:

```
//////////////////////////////////////////////////////////////
// VppSim Sim File: test.par
// Cell: link_top
// Library: HighSpeedLinks_Examples
//////////////////////////////////////////////////////////////

mex_prototype: [tx_out,  tx_out_delay, ch_out, rx_data_out, tx_data_out, ber, counter, del,
xi2.xi8.phase] = link_top(ratio, symbol_freq, trigger_freq, eq_tap_length, num_cdr_phases,
num_maj_votes, delay, num_sim_steps, Ts);

// Number of simulation time steps
num_sim_steps: 5e5

// Time step of simulator (in seconds)
Ts: 1/160e9

// Output File name
output: test

// Nodes to be included in Output File
probe: tx_out  tx_out_delay ch_out rx_data_out tx_data_out ber counter del xi2.xi8.phase

//////////////////////////////////////////////////////////////
// Note:  Items below can be kept blank if desired
//////////////////////////////////////////////////////////////

// Values for global parameters used in schematic
global_param:    ratio=20                      symbol_freq=1/Ts/ratio
                 trigger_freq=symbol_freq      eq_tap_length=5
                 num_cdr_phases=20                     num_maj_votes= 1024
                 delay= 10

// Rerun simulation with different global parameter values
// Example: alter: in_gl = 90:2:98
// See pages 37-38 of CppSim manual (i.e., alter: section)
alter:
```

Start the simulation by clicking **Compile/run** button.

Coefficients Setting and Time-Domain Results Plotting

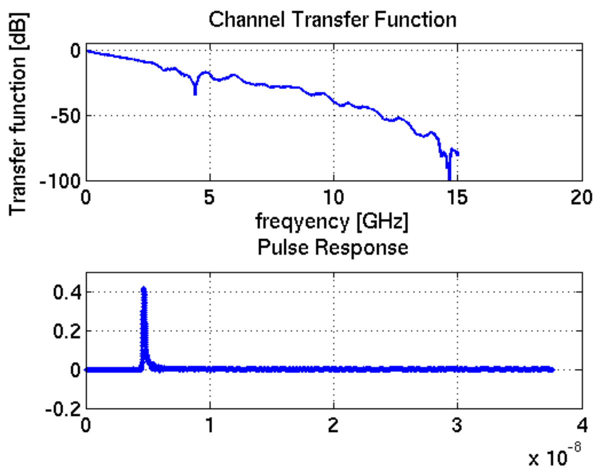
With the Matlab code:
…/VppSim/SimRuns/HighSpeedLinks_Exampls/link_top/Transceiver_testBench.m , you should be able to

- Determines the channel, data rate and generates the corresponding impulse response
- Set the pre-emphasis taps
- Plot the outputs

A. Determines the channel, data rate and generates the corresponding impulse response

The following code reads the channel s4p file and derives the impulse response. For the current setting, the data rate is **8Gbps** and the simulation oversampling ratio is **20**. The VppSim settings (*Ts, ratio*), should be consistent with the Matlab code.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Part1: Read and plot the s4p file
channelName='channel_data.s4p';
mode='s21';
[f,H]=extract_mode_from_s4p(channelName,mode);
figure(1)
subplot(211),plot(f*1e-9,20*log10(abs(H)),'b'); xlabel('freqyency [GHz]'); ylabel('Transfer function [dB]'); grid
on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Part2: Generate and plot the channel impulse reponse
Tsym = 1/8e9;
Ts=Tsym/20;
imp=xfr_fn_to_imp(f,H,Ts,Tsym); %%% Ts sampled impulse response
nsym_short=300;
   %%% persistance of the impulse response tail in the channel in terms of the number of symbols
imp_short=imp(1:min(length(imp), floor(nsym_short*Tsym/Ts)));
subplot(212), plot(conv(imp, ones(1, floor(Tsym/Ts))),'b.-'); hold on;
save link_channel.dat imp_short -ascii;
```



B. Determine the pre-emphasis taps

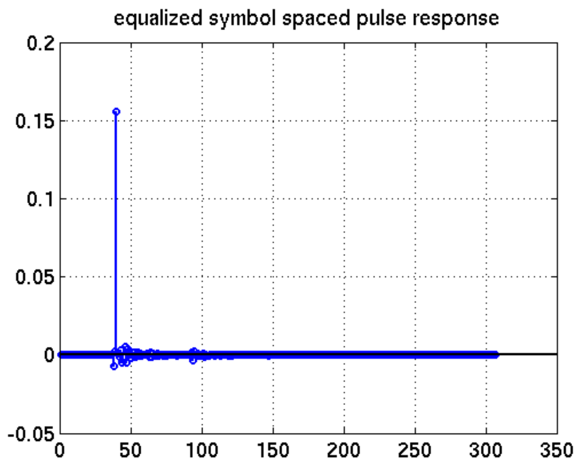
The following code derives the zero-forcing pre-emphasis tap weights.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% %% Part3: Compute initial ZFE equalizer solution
eq_tap_length=5;    % length of the pre-emphasis filter
eq_tap_pre=2;       % length of taps for post-cursor
eq_taps=zfe(imp_short,Tsym,Ts,eq_tap_length,eq_tap_pre)
% eq_taps=[0 1 0 0 0];
save eq_taps.dat eq_taps -ascii;
```

For the current channel, one group of pre-emphasis tap weights is:

[-8.8770e-02   4.7865e-01 -2.7445e-01   1.0411e-01 -5.4023e-02]

Given the existence of the pre-emphasis equalization, the symbol spaced channel response is:

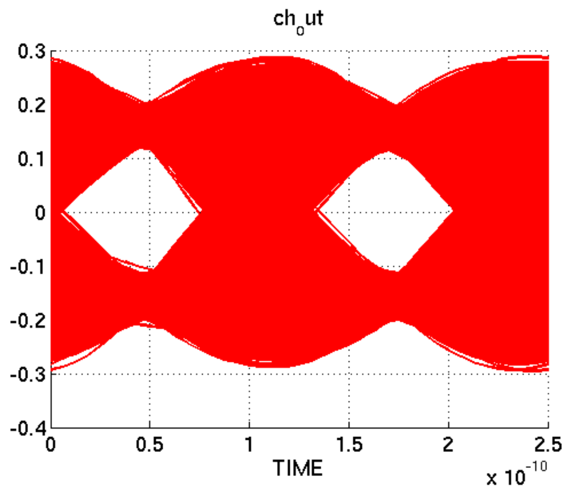


Thus, the channel impulse response and tap weights are set correctly. You can re-run the VppSim simulation with these new settings.

C. Draw the eye diagram

After re-run the VppSim simulation, the following scripts draws the eye diagram.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% %% Part4: Eye diagram drawing
%%% Load the signals after simulation
x=loadsig('test.tr0');
lssig(x)
figure(3)
eyesig(x, 2*Tsym, 100*Tsym,'ch_out');
```



D. Count the error bits

To counter the error bits, an **err_counter** cell is developed. It sweeps the delay for the transmitted data within a pre-set range (phase_min, phase_max) until it matches the channel phase delay and then counts the errors. There are several ways to get this range. For instance, the equalized symbol spaced pulse response shows that the index for the main tap is 39. (If the pulse response is not available, the correlation of the transmitter data and received data can give this information too.)
We can set the sweep delay from 35 cycles to 45 cycles. The err_counter finds the right delay (40 cycles) and shows that:

```
Update...                          delay = 40.00
# of simulated bits=22000          # of errors = 0
```

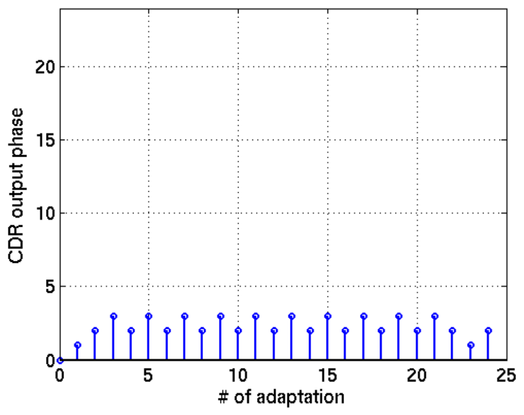
E. Check the CDR performance

The following code plot the CDR output phase:

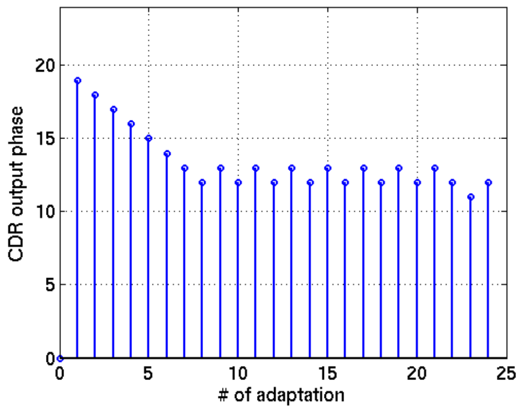
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Part5: CDR output phase
y = loadsig('phase.tr0');
lssig(y)
tp = evalsig(y, 'TIME');
phase = evalsig(y, 'phase');
figure(6)
plot(tp, phase);
```

In the current setting, the number of the CDR output phase is the same as the simulation oversampling ratio, which is 20. VppSim offers the interpolation function so that there are more CDR phases than the oversampling ratio.

In the test.par file, parameter **delay** is **0**. Thus, the delay introduced by the **delay_sample_scalar** cell is zero. The CDR output phase is:



When **delay** is 10 (10 simulator time samples), the CDR output phase is:



Conclusions

In this tutorial, a simple behavior model of a high-speed-link transceiver with transmitter pre-emphasis and Alexander type clock-data-recovery technique is completed. VppSim with Matlab simulates this model and shows the output waveforms. A Matlab code is developed to derive the zero-forcing pre-emphasis taps and an error counter cell is given to adjust the phase and count the errors.

Appendix I: Run VppSim within Matlab

VppSim nominally runs as a standalone executable, and interaction with Matlab occurs through file transfer using ‘probe:’ statements in VppSim and ‘loadsig(‘xxx’)’ statements in Matlab. However, there are times when it is more convenient to work directly with a VppSim object in Matlab. The ‘mex prototype:’ command allows automatic generation of a Matlab mex file corresponding to a given VppSim system which can then be compiled and run directly in Matlab.

You can find the details of this procedure in the CppSim Reference Manual (http://www.cppsim.com). In the section, we repeat the same simulations with this method.

Go back to the schematic of **link_top** in the **HighSpeedLinks_Examples** lib, open the **VppSim GUI**, and click button ‘**Edit Sim File**’ to open test.par file. Be sure that the following script is in the file:

```
mex_prototype: [tx_out,  tx_out_delay, ch_out, rx_data_out, tx_data_out, ber, counter, del, xi2.xi8.phase] =
link_top(ratio, symbol_freq, trigger_freq, eq_tap_length, num_cdr_phases, num_maj_votes, delay,
num_sim_steps, Ts);
```

If it is not in your sim file, add it and re-run the simulation once in the **VppSim GUI** to generate **compile_link_top.m** file. Whenever this script is changed, we need to re-run the simulation once in the **VppSim GUI**.

Return to the Matlab window, open file
…/SimRuns/HighSpeedLinks_Exampls/link_top/Transceiver_testBench_MexPrototype.m

Before running the simulation, make sure that the **Matlab current working directory** is:
…/SimRuns/HighSpeedLinks_Exampls/link_top/Transceiver

Thus run command ‘**mex -setup**’ once and Choose the Template Options file for buiding **gcc** MEX-files (i.e., gccopts.sh).

Afterwards, run the command ‘**compile_link_top**’.

Upon completion of the above, you can then run the mex function (link_top(…)) in Matlab as specified by the prototype format.

Now we are ready to run *Transceiver_testBench_MexPrototype.m*, and plot channel transfer function, pulse response, equalized symbol pulsed response, channel output eye diagram and CDR output phase.

You are free to change the configurations freely in the Matlab. Mex function transfers them to the VppSim simulator to guarantee that both of them run with the consistent simulation parameters.

Appendix II: Tx-emphasis Adaptation and DFE Adaption

In the **HighSpeedLinks_Examples** lib, a framework **link_top_pg** for Tx-emphasis adaptation is given. You can find the adaptation engine: **eq_setting_from_file_pg** cell and please check its code. In comparison with the **link_top** schematic, there are several differences:

1. Tx pre-emphasis is adaptive. The adaptive engine monitors the channel output signal and transmitted stream to generate the adaptation information.
    a. The main tap (tap1) and the first post tap (tap2) are adaptive and the rest are fixed.
    b. The desired received signal dlev is adaptive
    c. It adapts on the positive bits.
    d. The error is defined to be the difference between dlev and the received signal given the condition that +1 is transmitted.
    e. It adapts on the fourth bits (adapt_divide_rate = 4). Averaging scheme can reduce the noise further by reducing the adaptation rate.

2. The output phase of the receiver cdr is fixed (=5) to break the cross-coupling between the pre-emphasis adaptation loop and cdr phase adaptation loop.

The essential script for adaptation is:

```
//////////////////////////////////////////////////////////////////
/////////   Edit this part to make it adaptive    ///////////////
//////////////////////////////////////////////////////////////////

    if (rx_signal - dlevadapt > 0)  //case:  error >0
     {
        while (txdatalist.notdone )
        {
          if (txdatalist.read() > 0)
             deltalist.inp(-1 * stepsize * (rx_signal - dlevadapt) );      //generates deltas for taps
          else
             deltalist.inp(          stepsize * (rx_signal - dlevadapt) );
        }
              dlevadapt += stepsize * (rx_signal - dlevadapt) ;
      }
     else                    //case:  error < 0
     {
        while (txdatalist.notdone )
        {
          if (txdatalist.read() > 0)
             deltalist.inp( -1*  stepsize * (rx_signal - dlevadapt) );     //generates deltas for taps
          else
             deltalist.inp(      stepsize * (rx_signal - dlevadapt) );
        }
              dlevadapt += stepsize * (rx_signal - dlevadapt);
      }

//////////////////////////////////////////////////////////////////
/////////                  End of Edit                ///////////////
//////////////////////////////////////////////////////////////////
```

After running the simulation, you can run the Matlab code …/*VppSim/SimRuns/HighSpeedLinks_Exampls/link_top_pg/Transceiver_pg_testBench.m* to check the results.

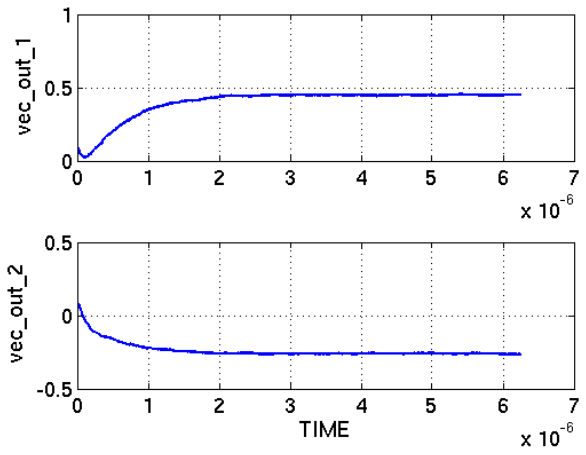


Figure 1 Evaluation of the main tap (vec_out_1) and the first post-tap (vec_out_2)

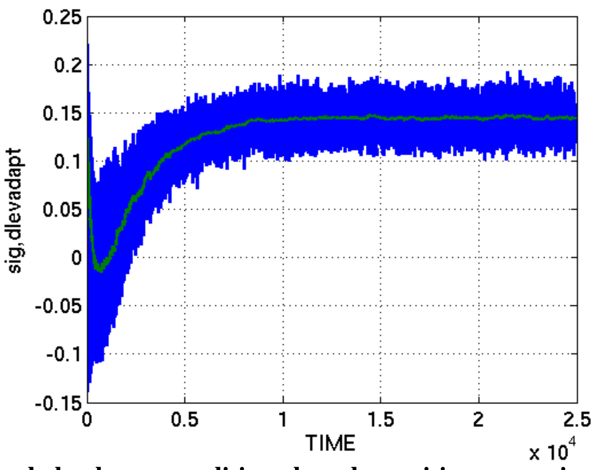


Figure 2 Receiver sampled voltages conditioned on the positive transmitted bits and dlev evolution

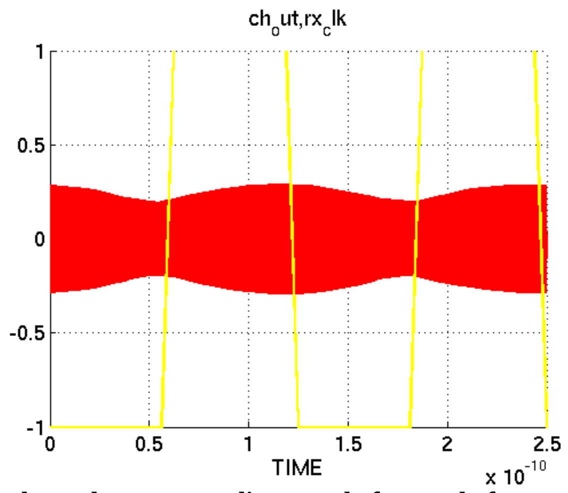


Figure 3 Superposition of the channel output eye diagrams before and after convergence with receiver clock.

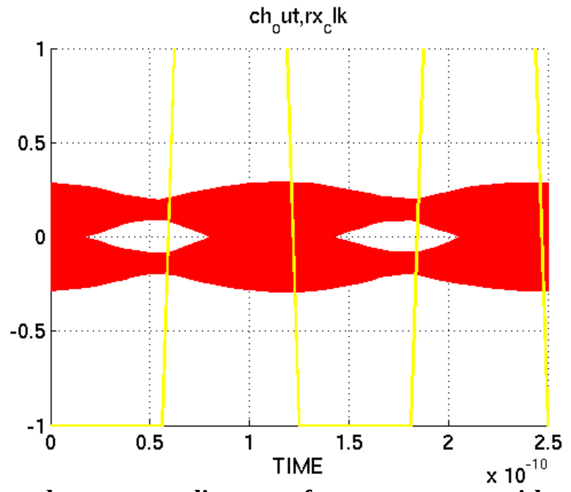


Figure 4 Channel output eye diagram after convergence with receiver clock.