Proposal for modeling advanced SERDES Discussion on API

IBM, Cadence

July 2006

Key Modeling Requirements

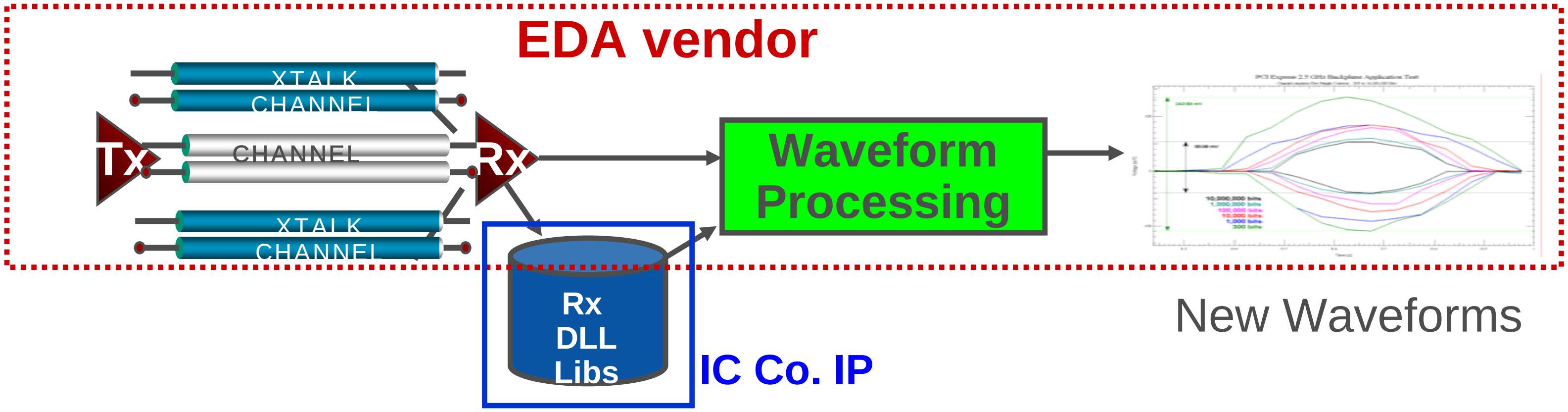
- Ability to capture complex ***algorithms***
  - DSP / Filter optimization: CDR, DFE, …
- Minimal model development time
- High accuracy (hardware correlated) with minimum simulation time
- Protection of IP (Silicon vendors)
- Architectural modeling
  - Ability to model & evaluate IP before silicon is developed (pre-silicon)
- Integration with PCB design environment
- Interoperability of models from different IP/IC Vendors
- Supported by EDA vendors
- Available as a public standard
- Available as soft IP for measurement vendors

Overview of this proposal

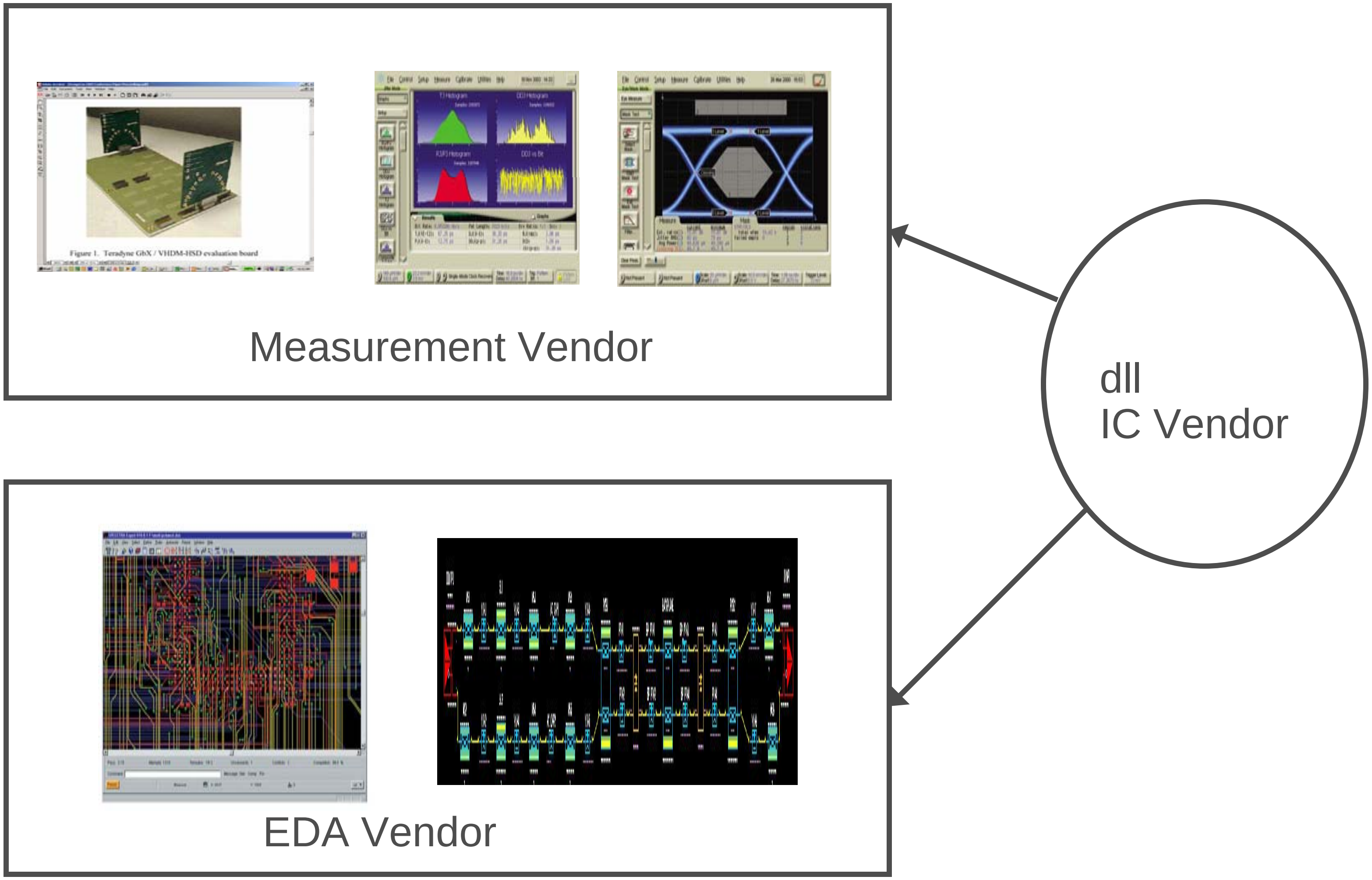
- Chip to chip modeling is infeasible with device level simulators of today
- Chip to chip communications have strong DSP content
- Algorithm modeling platform is a natural choice for chip to chip communication
  - Offers high performance for jitter analysis and budgeting
  - Offers measurements correlation capability
  - Enables compliance testing
- Algorithm model platform is prevalent in IC companies
- Proposal standardizes interface to algorithm platform

Proposed Solution & Architecture

- Allow IC companies to develop “executable” algorithm based models that plug into the simulator through a dynamically linked library (dll)
- Simplest possible public API (C-wrapper)
- Algorithmic Models in a dll
  - Can capture and encapsulate complex algorithms
  - Can add Jitter
  - Can include CDR modules
  - Protects IP without tool-specific encryption, no simulator specific encryption needed
  - Provides SERDES and EDA vendor independent interoperability if standardized
  - Can complete measurement loop – pluggable soft IP



Measurement Loop



Measurement Vendor

dll
IC Vendor

EDA Vendor

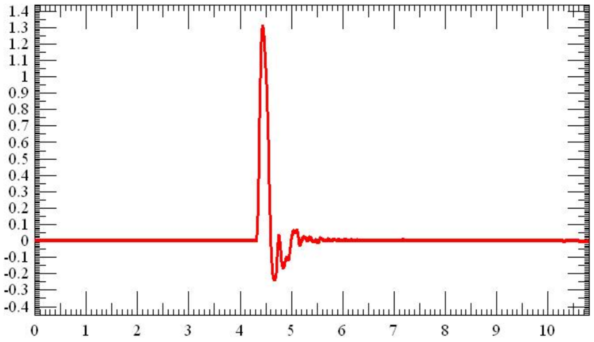
Simple API

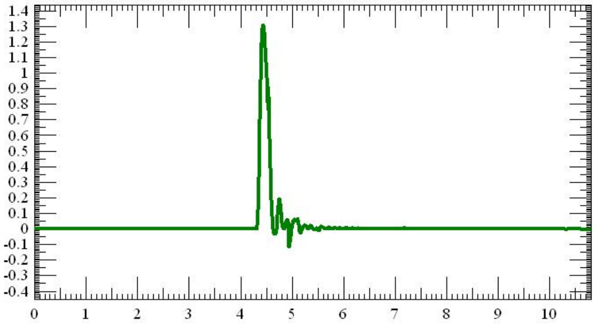
- Init
  - Initialize and optimize channel with Tx / Rx Model
  - This is where the IC DSP decides how to drive the system: e.g., filter coefficients, channel compensation, …
  - Input: Channel Characterization, system and dll specific parameters from configuration file
    - bit period, sampling intervals, # of forward/backward coefficients, …
  - Output: Modified Channel Characterization, status
- GetWave
  - Modify continuous time domain waveform [CDR, Post Processing]
  - Input: Voltage at Rx input at specific times
  - Output: Modified Voltage, Clock tics (dll specific), status
- Close
  - Clean up, exit

Parameters passed by the system simulation platform are in red

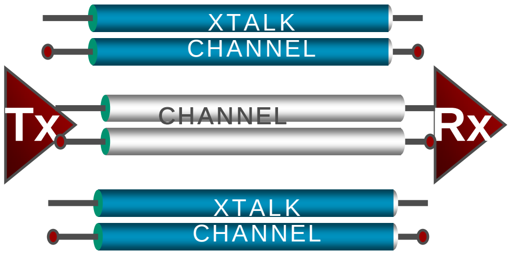
Simulator – Model interaction sequence

1. Characterize Channel (convolution engine)
2. Pass Impulse response to Tx & receive modified impulse response from Tx (Init call)
3. Send modified impulse response to Rx & receive Rx modified impulse response (init call)
4. Bit by Bit simulation
5. Send waveform data to Rx dll (GetWave call)
6. Close when done



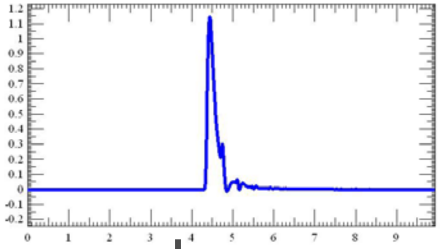


Rx_init

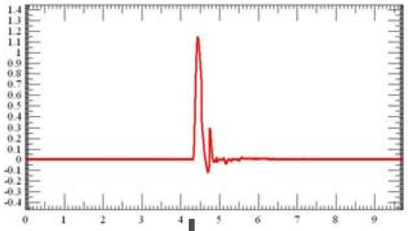


Characterized Channel

Pass characterization
in matrix ‘a’



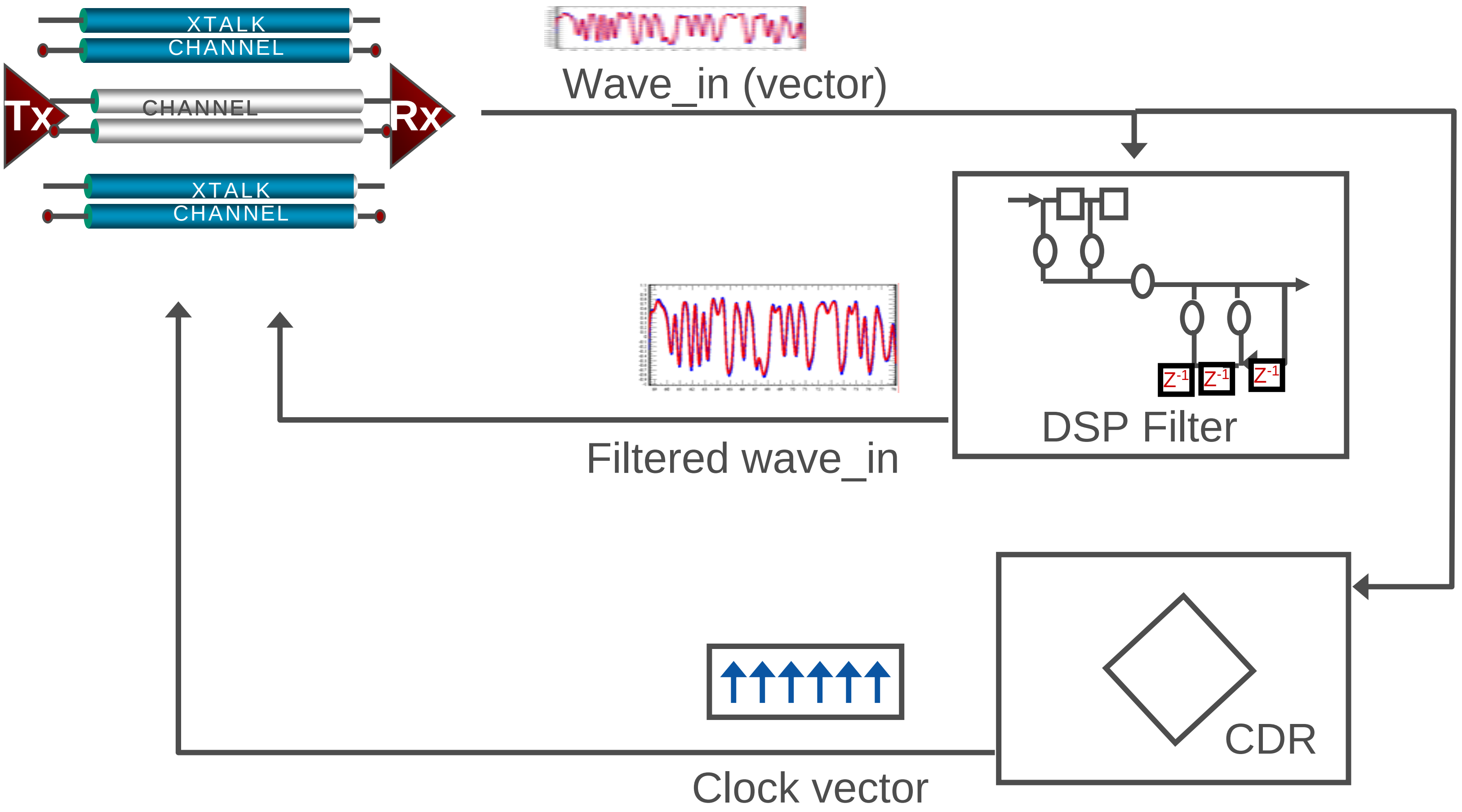
DSP algorithms
modify characterization



Send modified char back
(modified matrix ‘a’)

Internal storage

Rx_getwave


XTALK CHANNEL
Tx
CHANNEL
Rx
XTALK CHANNEL
Wave_in (vector)
Z^{-1}
Z^{-1}
Z^{-1}
DSP Filter
Filtered wave_in
CDR
Clock vector

API Call Params

- long ***rx_init*** (double *a, long row_size, long col_size, double bitp, double tr, double tf, void **pdll_server_param_obj, void *dll_client_param, char *dllcontrol, [genchdllmsg_type **msg])
  - Input: Channel Characterization, system and dll specific parameters from config file
    - bit period, sampling intervals, # of forward/backward coefficients, …
  - Output: Modified Channel Characterization, status

- long ***rx_getwave*** (double *wave_in, long size, double dt, double *clk, void *dll_server_param_obj, void *dll_client_param, [genchdllmsg_type **msg])
  - Input: Voltage at Rx input at specific times
  - Output: Modified Voltage, Clock tics (dll specific), status
- long ***rx_close*** (void **ptr_2_dll_server_param_obj)
  - Clean up, exit

Note: items in [ ] are optional and can be 0(null)

Rx_init

long ***rx_init*** (double *a, long row_size, long col_size, double pulse_width, double tr, double tf, void **pdll_server_param_obj, void *dll_client_param, char *dllcontrol, [genchdllmsg_type **error_msg])

Call:

long status = rx_init(…), status >=1 for success, 0 for failure

a = matrix of row_size x col_size

col_size = (number of channels + 1)

- first column is time

pulse_width = pulse width of the characterization

tr, tf = rise and fall times, useful for synthesizing filters

pdll_server_param_obj – place holder for data structure created by the dll. dll's can use this to store and retrieve additional information

dllcontrols – this is string in a tree data base format and will contain information like dll version number. It can be also used by the dlls to manage additional features and controls

dll_server_obj – This is an optional argument. The dll server use this place holder to create a dll structure for its own use. In this way the dll need not use global variables.

error_msg – optional error message

The dll should not free memory of a/txids

Rx_init – input matrix indexing

a, the input matrix is a one dimensional double array

– The index into the array is given by

index = row_size * j + i

where i is the row index and j is the col index. i and j start from 0

- ‘a’ is the normalized impulse response i.e. it is the channel response for a unit pulse

Rx_getwave

long ***rx_getwave*** (double *wave_in, long size, double dt, double *cdrclkbuff, void *dll_server_param_obj, void *dll_client_param, [genchdllmsg_type **error_msg]))

Call: long status = rx_getwave (..)

wave_in – vector of input voltage

dt – sampling intervel for wave_in

size – the size of wave_in vector

On return the rx_getwave replaces wave_in with the computed wave_out

cdrclkbuff – This is the vector of clk edges with a size of ‘size’, same as the wave_in buffer.If the dll includes a cdr function , you can fill this vector with the expected edge times. If there is no cdr function, ignore this vector. This vector will be initialized with a ‘-1’ at the 0th position. If the vector is not modified i.e. on return if the caller still finds the -1, the caller will conclude there is no cdr function.

The times in the vector should be referenced to the start of the cdrclkbuff. For example

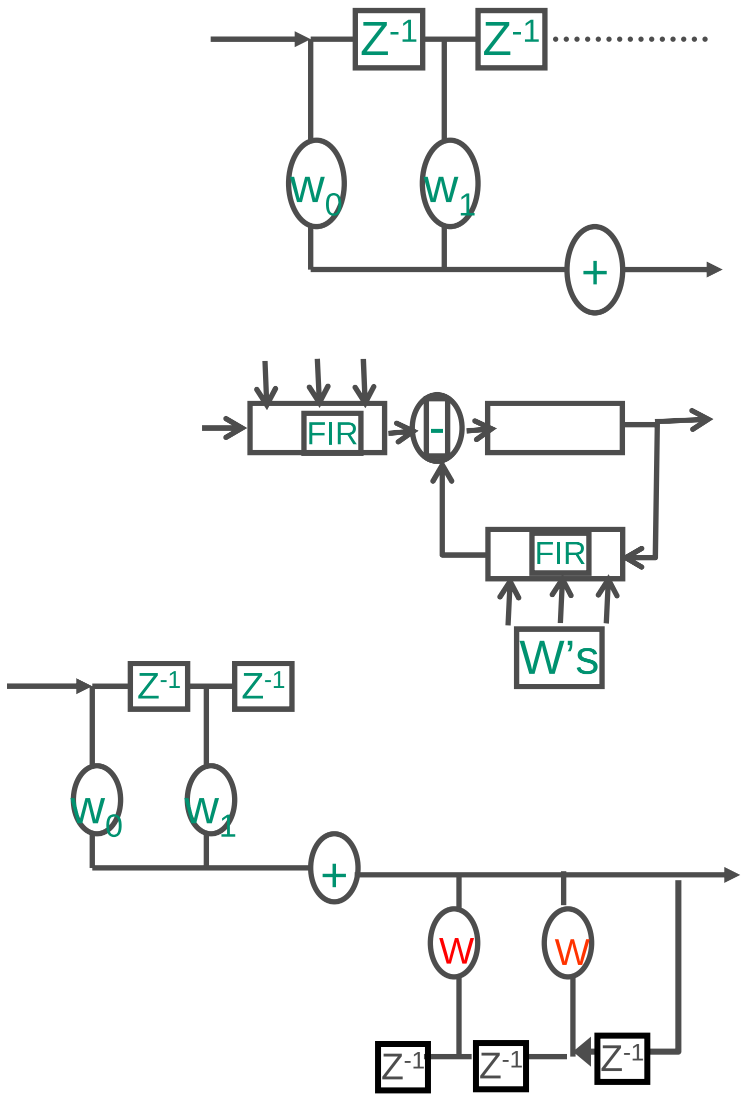
cdrclkbuff = [30n 30.2n 30.4n 0 0 ….]

Means that the only 3 clock edges were found at 30n, 30.2n and 30.4n

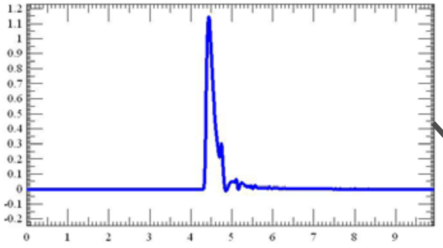
All memory will be freed by the caller

Sample models

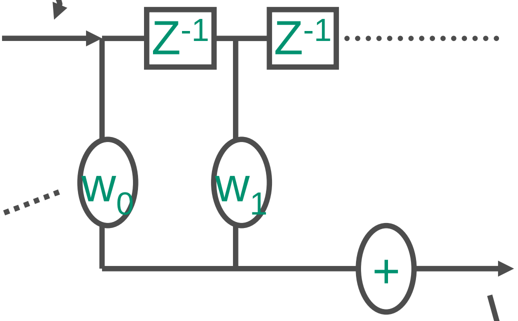
1. chffefilt
   - Optimized Feed Forward Filter
2. chdfefilt
   - Decision Feedback Filter
3. chfbefilt
   - Feed back equalization
4. chcdr
   - Clock and Data Recovery unit with Proportional Integral (PI) control



Sample FFE Filter



- Example FFE Filter
- Multi tap FFE
- MMSE Optimize FFE weights for given channel
- Apply FFE bit by bit



Adjustable number of taps

(chffefilt (fwd 5)(pulsein ffein.txt) (pulseout ffeout.txt))

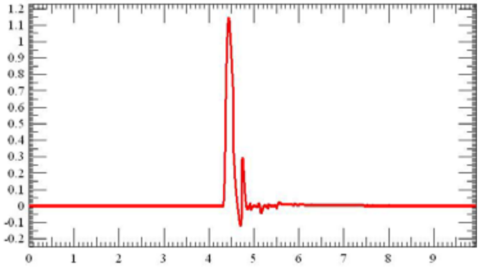
dll Name

Parameters

5 taps

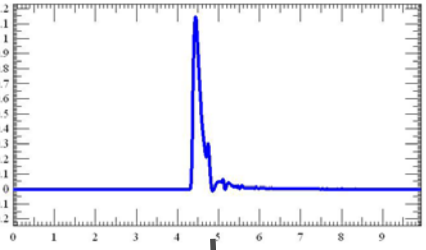
Read pulse from ffein.txt

Write pulse from ffeout.txt

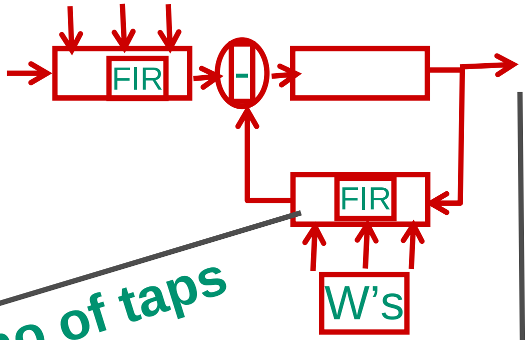


MMSE: Minimum Mean Square Error

Sample DFE Filter



- Multitap FFE+DFE
- MMSE* optimization for FFE
- Zero forcing DFE
- Modify pulse response



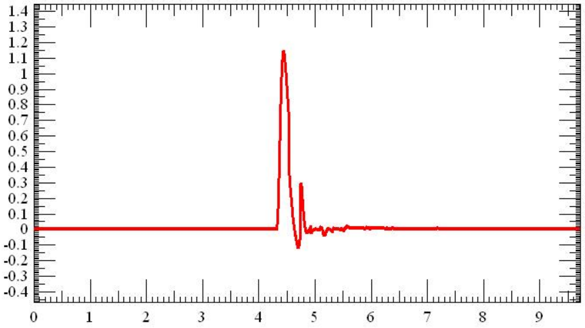
Adjustable no of taps

chdfefilt (bwd 12)(pulseout dfeout.txt))

Dll Name

Parameters

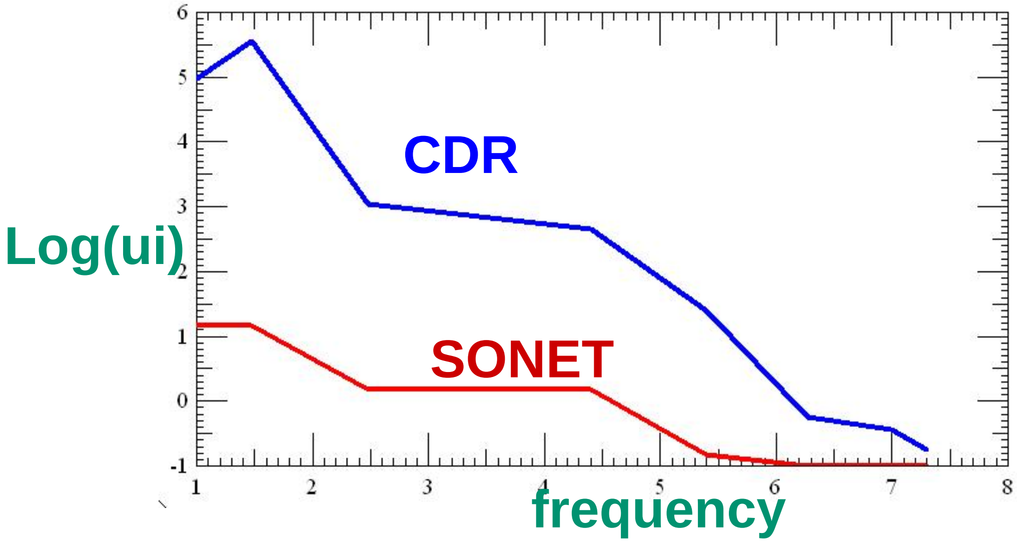
Backward # DFE taps

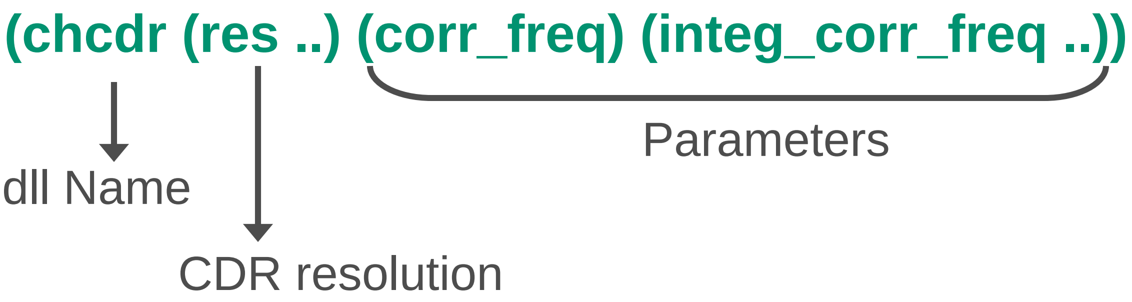


MMSE: Minimum Mean Square Error

Sample CDR model

- Clock and Data Recovery unit
- Proportional + integral error control
- Adjustable resolution
- Jitter tolerance





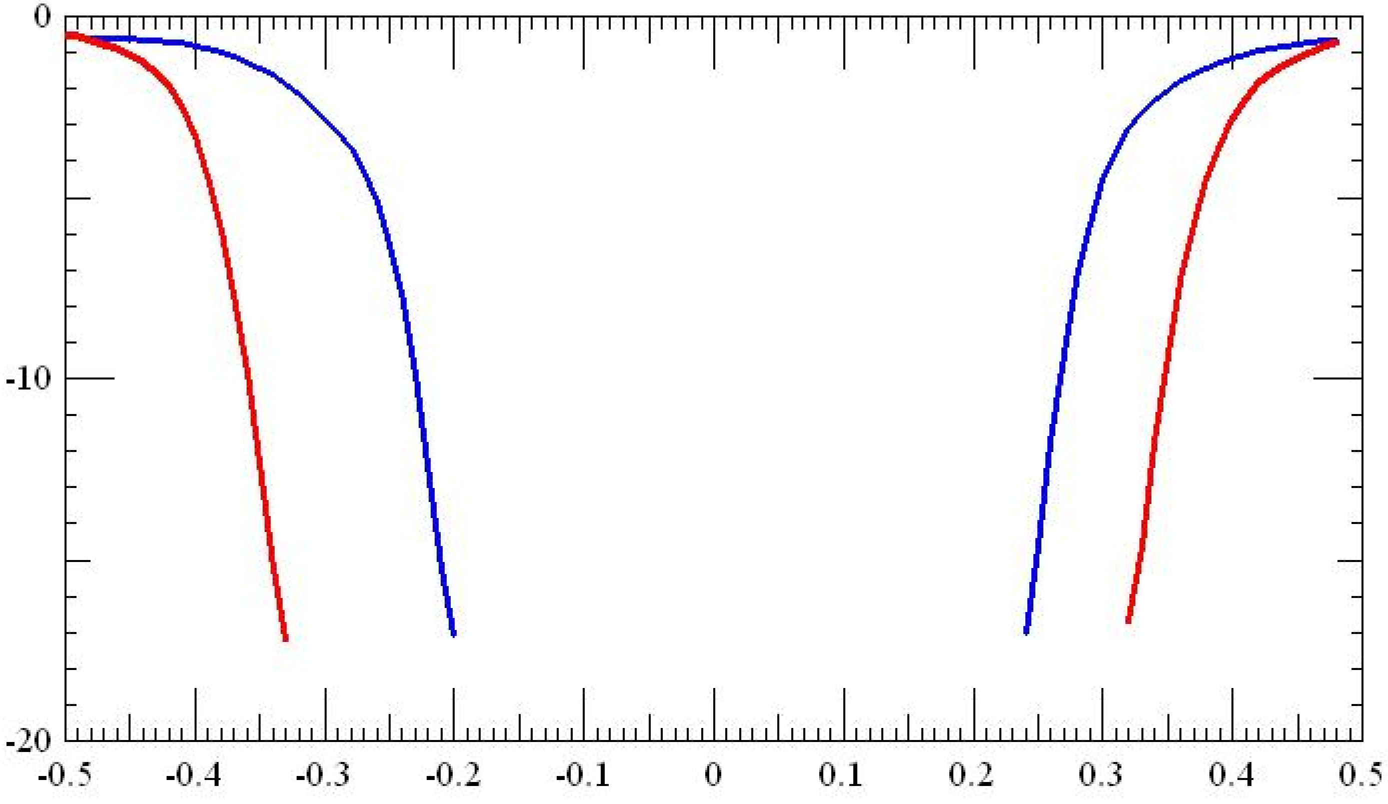
(chcdr (res ..) (corr_freq) (integ_corr_freq ..))

dll Name

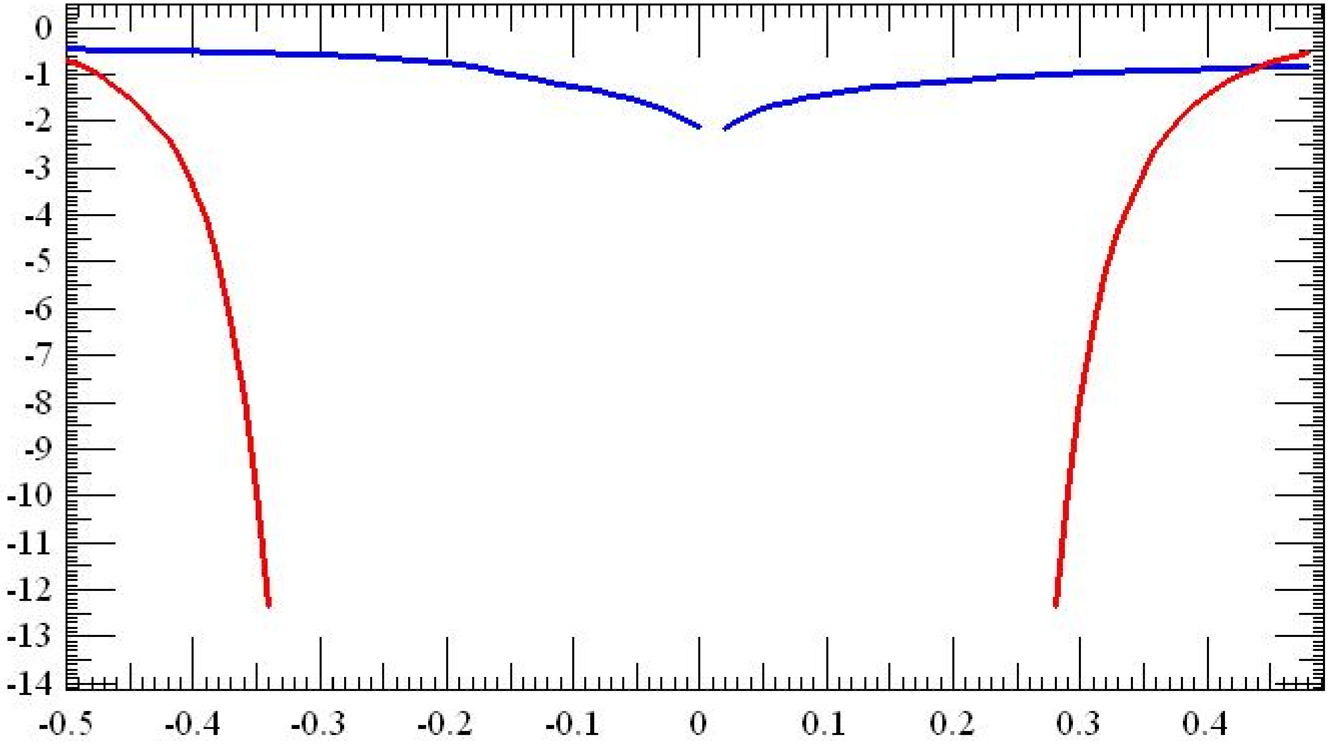
CDR resolution

Parameters

Bathtub Curve
- no filter vs. chfbefilt



Bathtub Curve
- no cdr vs. with cdr



Chffefilt code

```
extern long rx_init (double *a, long row_size, long col_size, double bitp, double tr,
   double tf, void **pdll_server_param_obj, char *dllcontrols, void *dll_client_param,
   genchdllmsg_type **msg)
{
 ,,,,,,,,,,,,
/* get the filter, pass the input matrix ‘a’ */
status  = dotaps (dll_object, a, row_size, col_size, bitp, tr, tf);
 DONE:
  if(!status)
    { genDestroy (dll_object); dll_object=0;}

  if(msg)
    printlogo (dll_object, msg);

  if(status > 1)
    destroy (dll_object);
  else if(pdll_server_param_obj)
    *pdll_server_param_obj = dll_object;

   return status;
}
```

Chffefilt - dotaps

```
/* create taps using MMSE */

 taps = genfilttbl_fwdcoeff (mx, fwd, bitp, trm, 0.0, offset, fbitp,
  forcepulse, &resp, &error, nonaveraging, &snr);
```

Chffefilt: rx_getwave

extern long rx_getwave (double *wave_in, long size, double dt, double *cdrclkbuff, void *dll_server_param_obj, void *dll_client_param, genchdllmsg_type **msg)

Apply the filter, Modify the input wave vector

```
for (i=0; i<size; i++)
  {
   double volt=0;

   if (dll_object->time <= 0) /* first time intitialize dc */
            dll_object->td = genfilttd_initstd (dll_object->taps, wave_in[i]);


   volt = genfilttd_y (dll_object->td, dll_object->time, wave_in[i]);


   wave_in [i] = volt;

   dll_object->time += dt;
  }
```

Chcdr:

```
extern long rx_getwave (double *wave_in, long size, double dt,
  double *cdrclkbuff, void *dll_server_param_obj, void
  *dll_client_param, genchdllmsg_type **msg)
```

Start early/late determination

```
  for (tedge += bitp; tedge < tlast; tedge += bitp)
```

Return clock information

```
        cdrclkbuff[edge_id++] = (tedge-tstart);
```

Summary

- Top down algorithm modeling
- Model IP in dll
- EDA vendors and measurement vendors
- Code examples