Algorithm Modeling Approach for SERDES Devices

Lance Wang – Cadence
Joe Abler – IBM

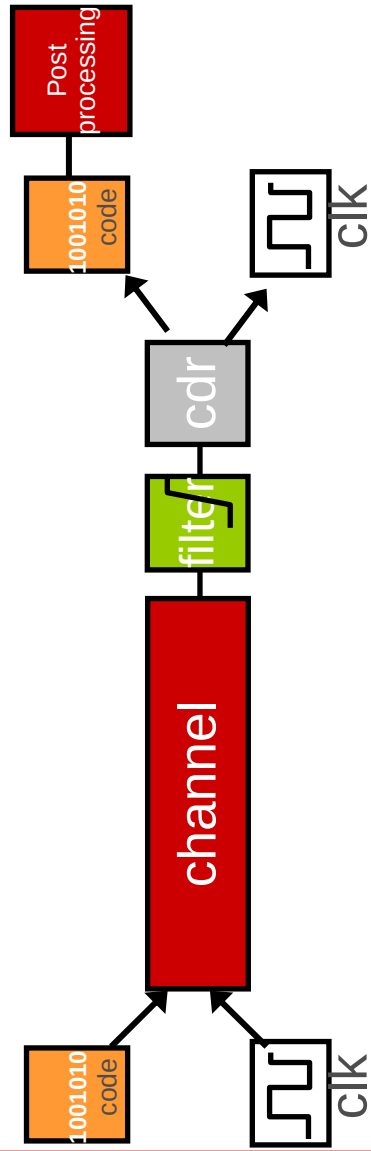
IBIS Summit - DAC 2006 – July 25, 2006

Outline

- Why Algorithm Modeling is important for High-Speed SERDES Designs and Validations
  - Challenges
  - The needs for System / Algorithmic Level Modeling
- Experimental case study and results
  - Algorithm Models and simple API
  - Case Study and results
  - Next Step

Advanced Serdes Modeling Challenge

- For 5+Gbps Serdes devices, complex signal processing ***algorithms*** often need to be represented, like:
  - FFE/DFE tap coefficient optimization (with/without crosstalk)
  - CDR algorithms
  - proprietary noise cancellation techniques
  - proprietary post-processing of data
- These algorithms are very difficult to represent with traditional device modeling techniques
  - They are typically modeled in higher level programming languages like C or Matlab
- There is currently no industry-standard way to handle this
  - IP suppliers have resorted to developing their own proprietary tools, increasing their support costs
  - This doesn't allow Serdes devices from multiple vendors to be simulated together (no interoperability)



Current Approaches Are Unsuitable

- Traditional structural modeling is too low-level
  - Even with major extensions, insufficient capacity for growth
- Transistor-level SPICE
  - Insufficient simulation capacity
  - Unable to capture algorithms
  - Unable to capture end to end system level simulation
- Behavioral MacroModeling / AMS
  - Insufficient simulation capacity
  - Unnatural choice for capturing algorithms
    - geared for detailed circuit modeling
  - No IP protection

Limitations of Device Level Modeling

- **Device level models have always been very simple with significant limitations**
  - Combination of Tx output stage with Rx load model
  - Do not even provide a complete jitter budget analysis
- **These models allow one to:**
  - Analyze the ISI introduced across the channel
    - Which must be factored into an independently (hand-calculated) jitter budget
  - Determine optimum pre-emphasis and launch voltage settings
  - Do basic electrical compatibility checking across vendor parts
- **This worked when:**
  - Jitter budgets were vast and generous (relatively)
  - Eyes were open & receiver equalization not required
  - Tx drivers only had a handful of pre-emphasis and launch voltage settings to twiddle with
  - Device non-linearity was a primary consideration
  - Other system effects had insignificant impact on overall performance
    - Crosstalk, Data pattern dependencies, Duty cycle distortion, CDR misalignment, ...
- **Extending this approach does not appear adequate nor practical**
  - Need to accurately model end to end equalization architecture
- **Need to move to system level modeling and channel analysis**

The Need for System Level Modeling

- **As speeds increase, system level effects have significant performance impacts**
  - Crosstalk
  - Duty cycle distortion - jitter amplification
  - DFE & CDR sampling alignment
- **DFE & CDR control algorithms are becoming more critical to system performance**
  - “Reference model" approaches (ala StatEye) will quickly become inadequate
  - Exclusion of algorithms will not allow accurate modeling of next generation systems
  - Algorithms are vendor specific, and modeling platform must provide a capability to support these
- **Complexity of serdes architecture and simulation requirements continue to grow**
  - Advanced DFE control algorithms
  - Adaptive equalization algorithms
  - Receiver FFE/DFE combinations
  - Complex crosstalk cancellation technologies
- **End to end linearity is a valid assumption**
  - Devices are designed for high linearity to optimize DFE performance
  - Device level modeling of I/O is less critical (particularly when AC coupled)

Data Pattern Dependencies/Xtalk Effects

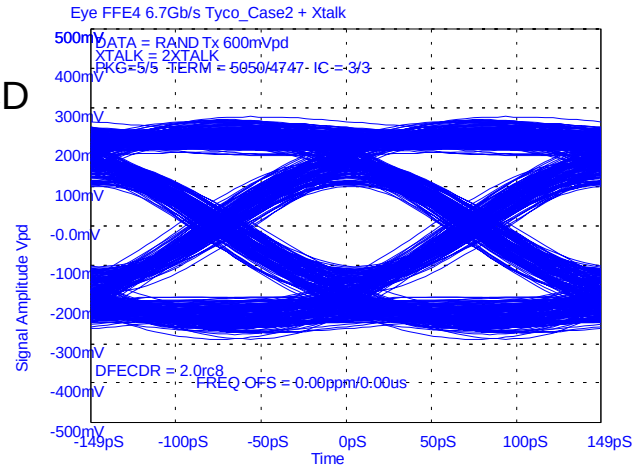
- **Adequate simulation time required to capture system level effects**
  - Data pattern dependencies
  - Crosstalk
- **Simulation results for example case**
  - Tyco Case 6
    - Includes Tyco xtalk channels
  - 10.3 Gbps operation
  - Random data
  - Only simulation length is varied
- **Modeling approach must allow fast, efficient simulation**
  - Else an inoperable channel can easily be evaluated as a channel with margin
  - IBM recommends the following simulation times:
    - 1M bit times for through channel analysis
    - 10M bit times for crosstalk analysis

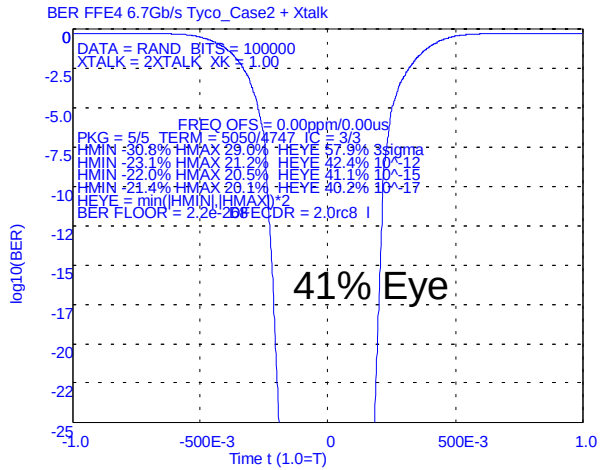
Sim length (bit times)	Sim time (minutes)	Heye margin (BER E-12)	Heye margin (BER E-15)	Heye margin (BER E-17)
100K	0.2	19.8%	17.9%	17.0%
1M	1.6	12.5%	11.3%	10.4%
10M	20.9	8.4%	5.8%	4.9%
100M	159	6.7%	2.9%	0.4%

Tx DCD Effects through Lossy Channel

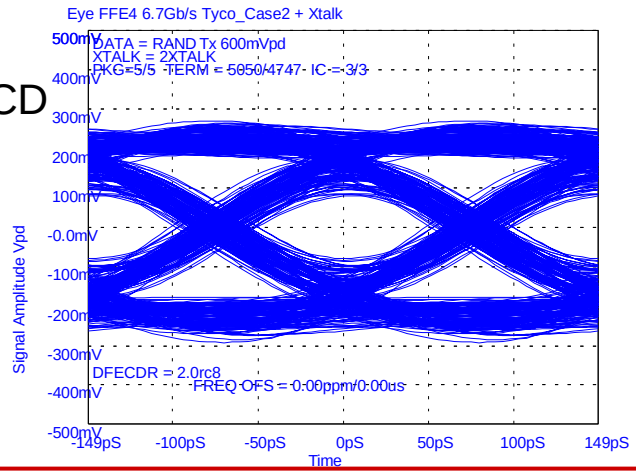
- **Tx DCD (& resulting jitter amplification) can quickly close an eye...**

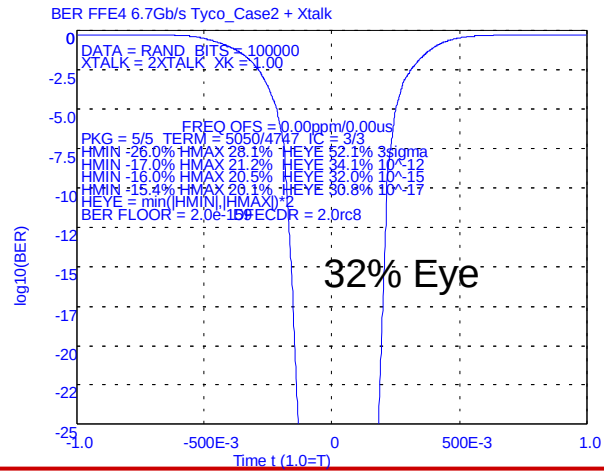
0% DCD





3.1% DCD



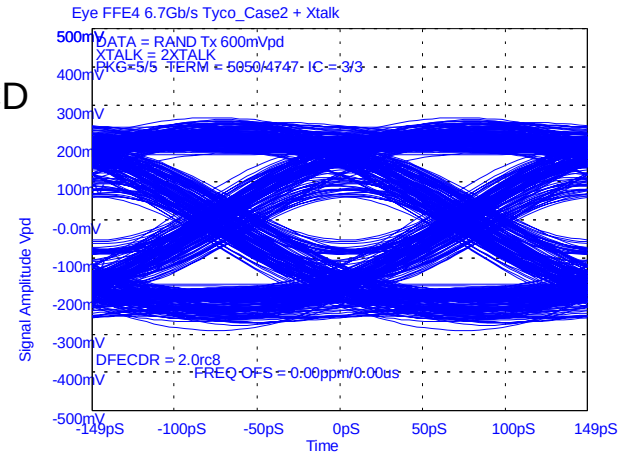


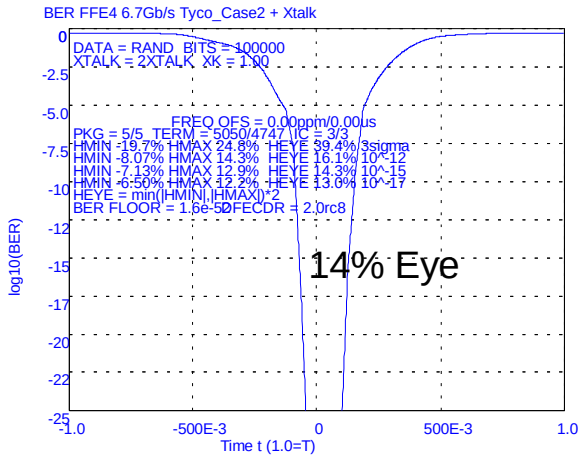
Tx DCD (Duty Circle Distortion) Effects through Lossy Channel

…Down to nothing!

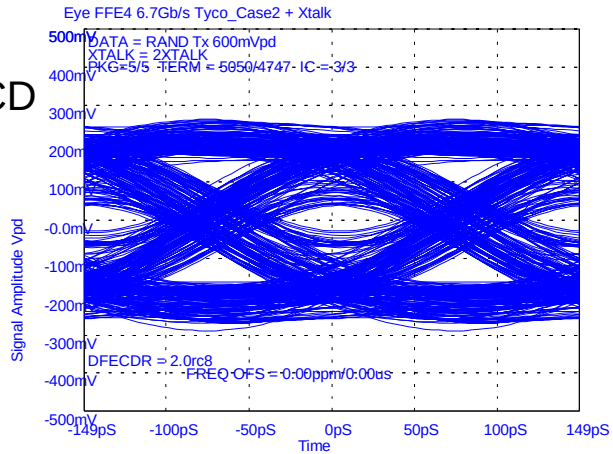
Control algorithms must be accurately modeled to analyze performance

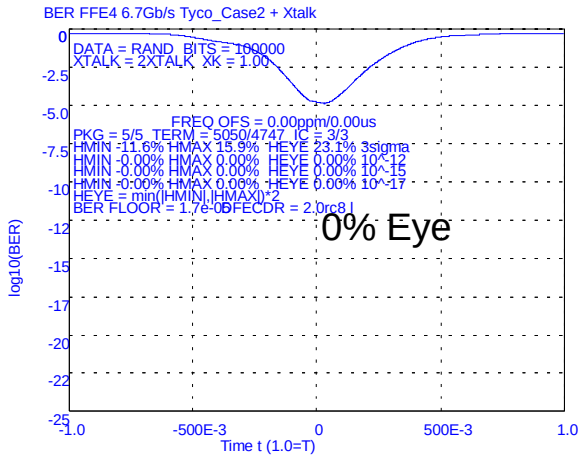
6.3% DCD





9.4% DCD





System / Algorithmic Level Modeling

- **Performance analysis requires a complete end-to-end system model**
  - Transmit device model => package => channel => package => receive device model
  - Must account for system level impairments
    - Complete jitter budget including RJ effects
    - Crosstalk and other noise sources
- **Serdes device models must be comprehensive**
  - Include all jitter sources
  - Accurately model datapath
    - Device I/O, including parasitic extractions
    - Transmitter FFE stages
    - Receiver peaking, AGC, & DFE stages
  - Fully model control algorithms
    - DFE amplitude centering
    - CDR centering & tracking
- **Results need to provide system level analysis & information**
  - Optimization of transmit FFE coefficients
  - System level BER analysis
  - Computations to enable hardware to model correlation
    - Expected DFE coefficient settling
    - Bathtub curves

Comparison of approaches

Key components	Circuit / Event driven	System simulation
Filtering (FFE, DFE, VITERBIE, ..)	No (requires pre-solved coefficients)	Yes
Optimization	No	Yes
CDR	Yes	Yes
Jitter components	Yes (Difficult)	Yes
Bathtub post processing	No	Yes
Channel Compliance	No	Yes
High Capacity Simulation (1 to 10 M bits)	No	Yes
Pre-Silicon modeling and evaluation	No	Yes

Key Modeling Requirements for Silicon/IP vendors

- Ability to capture complex ***algorithms***
  - DSP / Filter optimization: CDR, DFE, …
- Minimal model development time
- Best possible performance for 1 to 10M bits simulation
- Protection of IP
- Ability to model IP before silicon is developed (pre-silicon)
- Supported by EDA vendors
- Available as a public standard

Key Modeling Requirements for Systems Companies

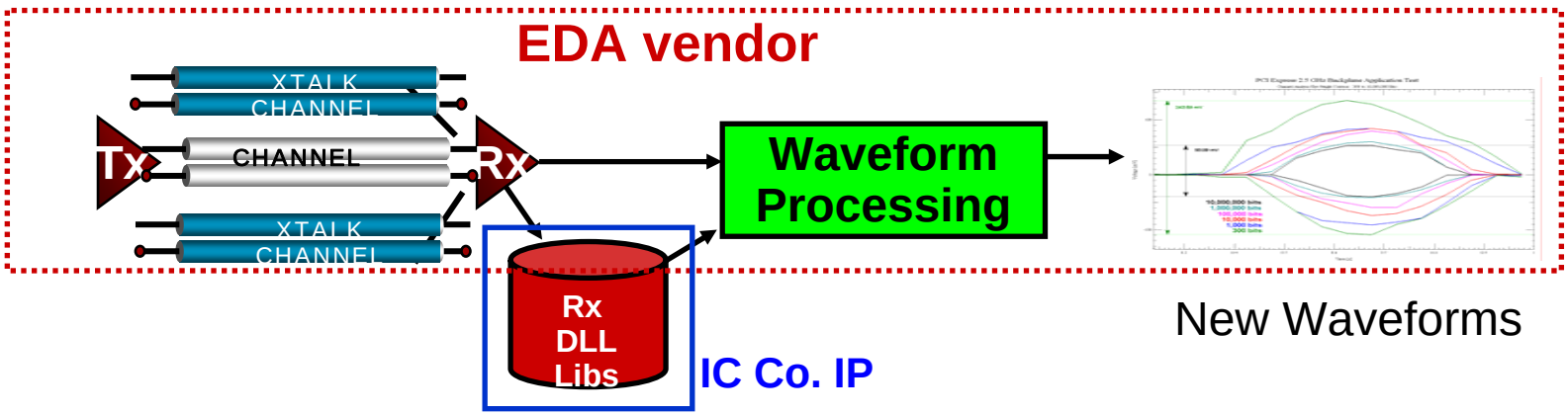
- Best possible performance for 1 to 10M bits simulation
- Interoperability between multiple IP providers
- Ability to evaluate IP before silicon is built (pre-silicon)
- Supported by EDA vendors
- Available as a public standard

Experimental case study and results

- Algorithm Models and simple API
  - Proposed to IBIS Macro Subcommittee for API standardization consideration
  - Continue to work on the details with IBIS Macro Subcommittee Group
- Case Study and results
- Next Steps

Proposed Solution & Architecture

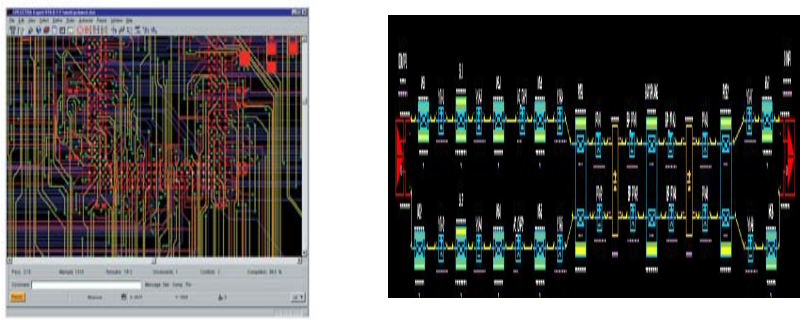
- Allow IC companies to develop “executable” algorithm based models that plug into the simulator through a dynamically linked library (dll)
- Simplest possible public API (C-wrapper)
- Algorithmic Models in a dll
  - Can capture and encapsulate complex algorithms
  - Can add Jitter
  - Can include CDR 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

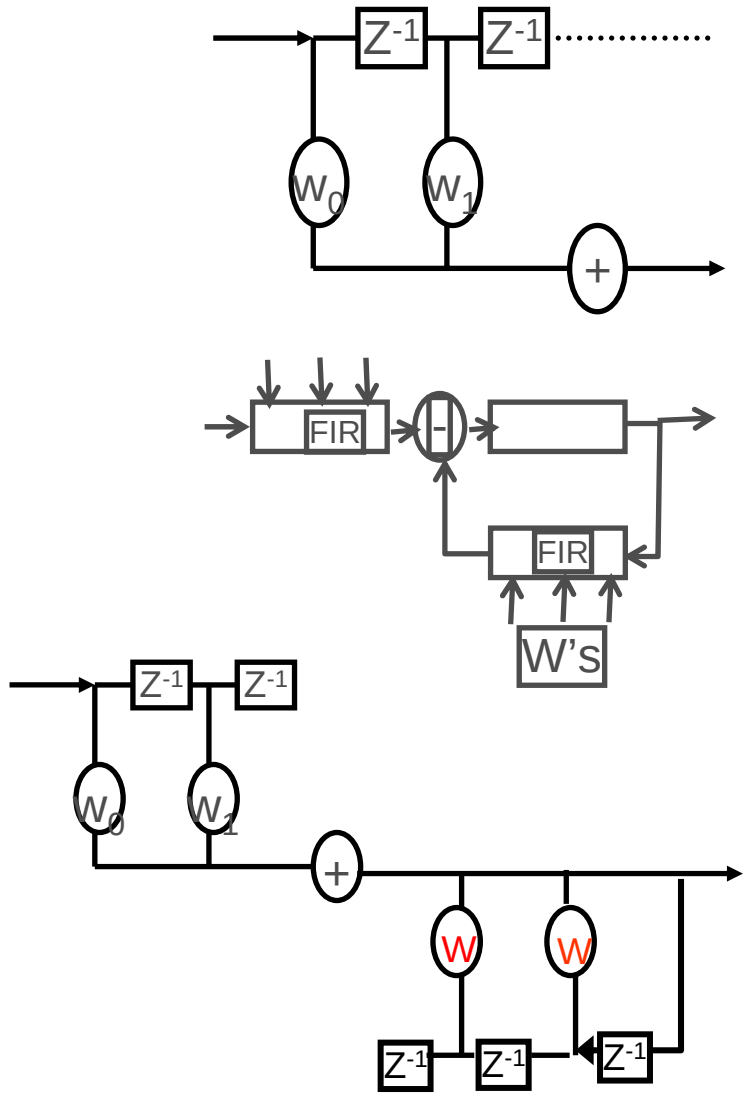
dll
IC Vendor



EDA Vendor

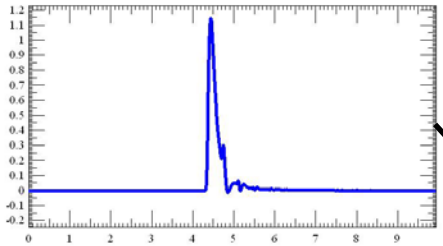
Sample models

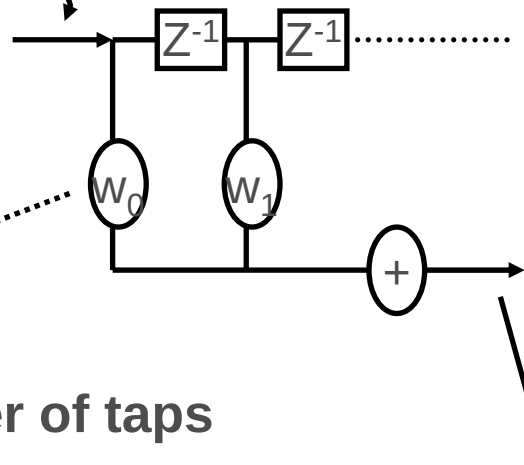
1. chffefilt
   - Optimized Feed Forward Filter
2. chdfefilt
   - Decision Feedback Filter
3. chfbefilt
   - Feed back equalization
4. chcdr
   - Clock and Data Recovery unit with Proportional Integral (PI) control



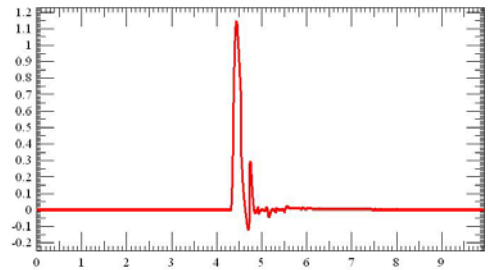
Sample FFE Filter

- 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

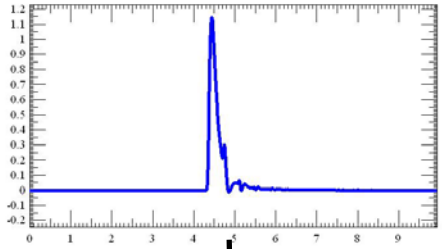
Read pulse from ffein.txt

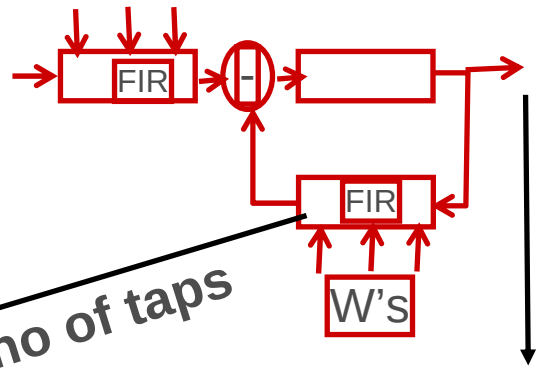
Write pulse from ffeout.txt

MMSE: Minimum Mean Square Error

Sample DFE Filter

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





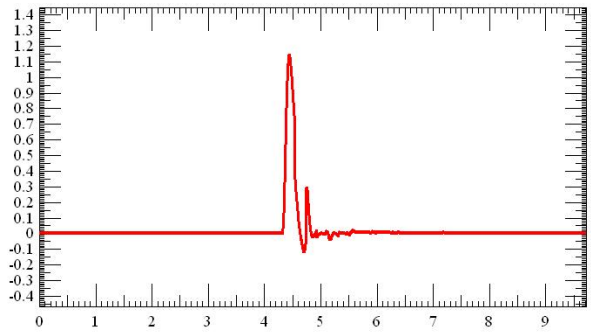
Adjustable no of taps

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

Dll Name

Parameters

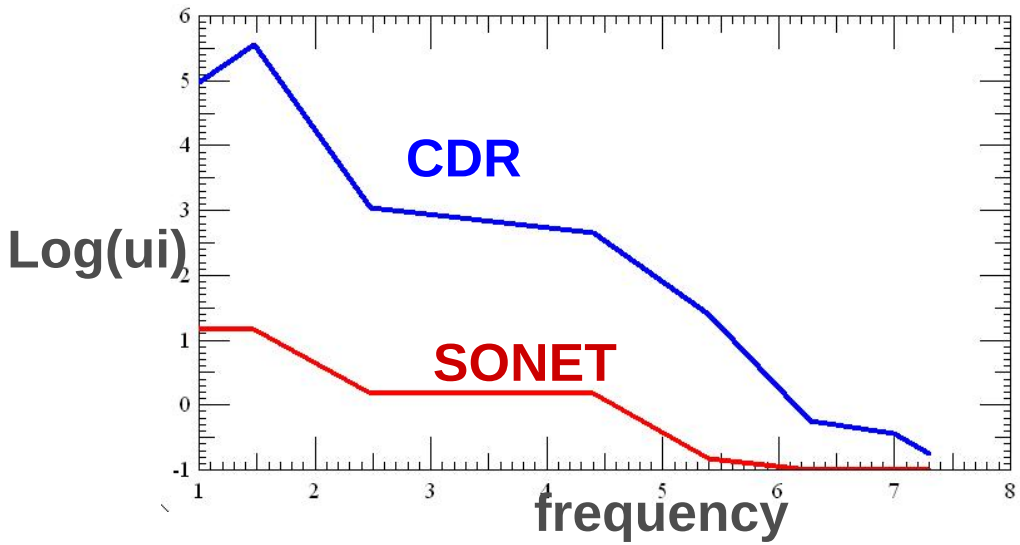
Backward # DFE taps



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

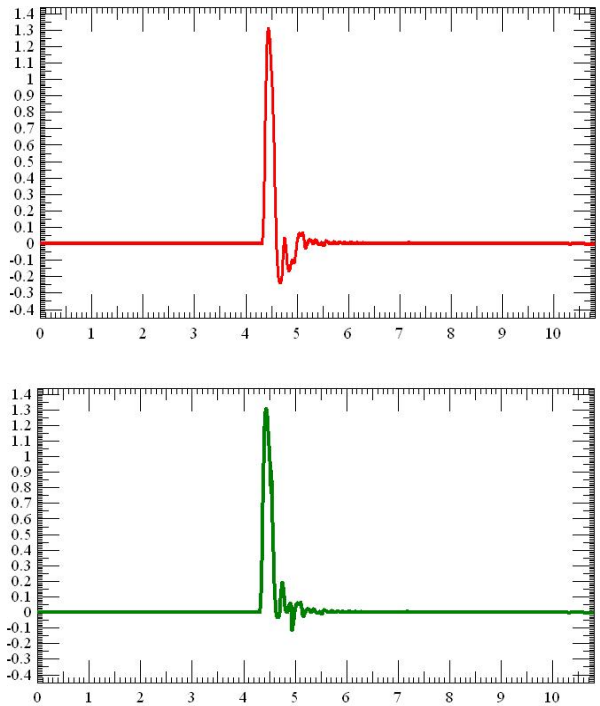
Simple API

- 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

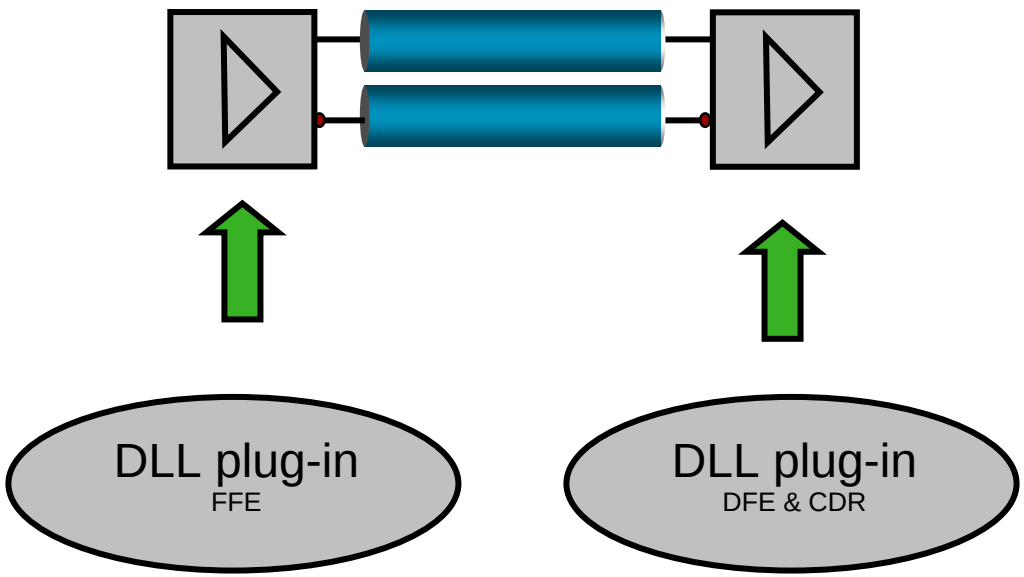
Parameters passed by the system simulation platform are in red

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



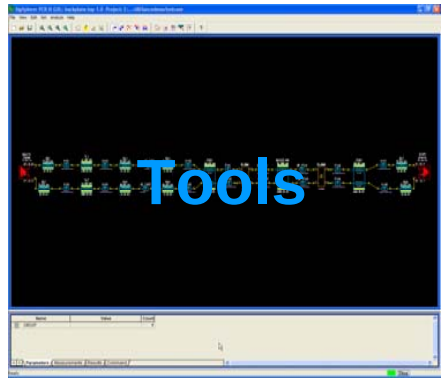
Test cases and Results
- Topology Structure



PCIe backplane (27”) @ 6.25 Gbps

DLL Libraries and AMI (algorithmic Modeling Interface) Pointers

Device Model Library




```
(ami
 (chdfefilt "/center_libs/lib_dll" )
  (forcepulse )
  (pulsein "pin.txt" )
  (pulseout "pout.txt" ) )
 (chdfefilt "/center_libs/lib_dll" )
  (forcepulse )
  (pulsein "pin.txt" )
  (pulseout "pout.txt" ) )
 (chcdr "/center_libs/lib_dll" )
)
```

dlls can be located in the different libraries

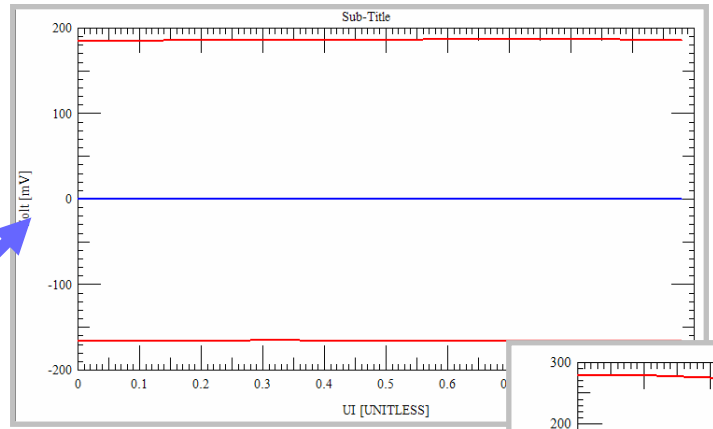
Test Cases and Results

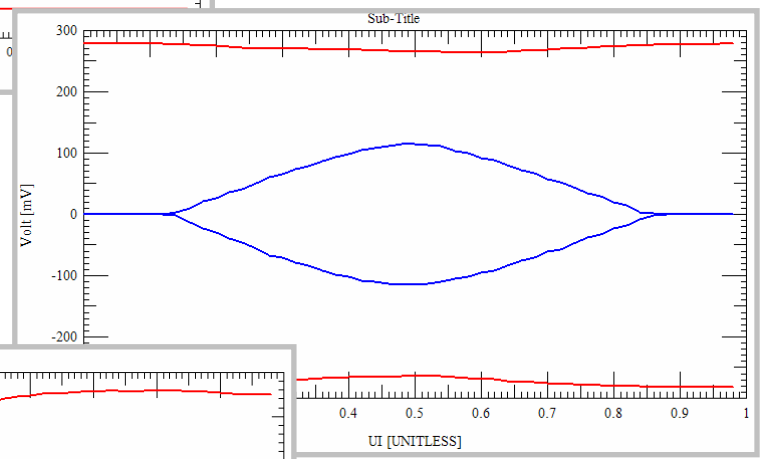
3 Cases

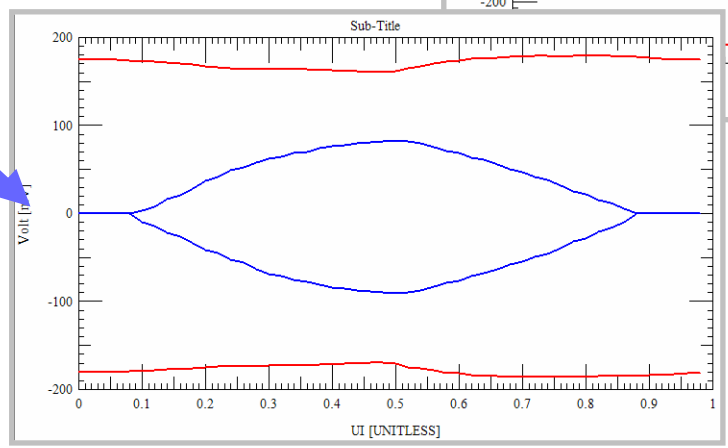
No filters

FFE, no DFE

DFE model with FFE coeff=1







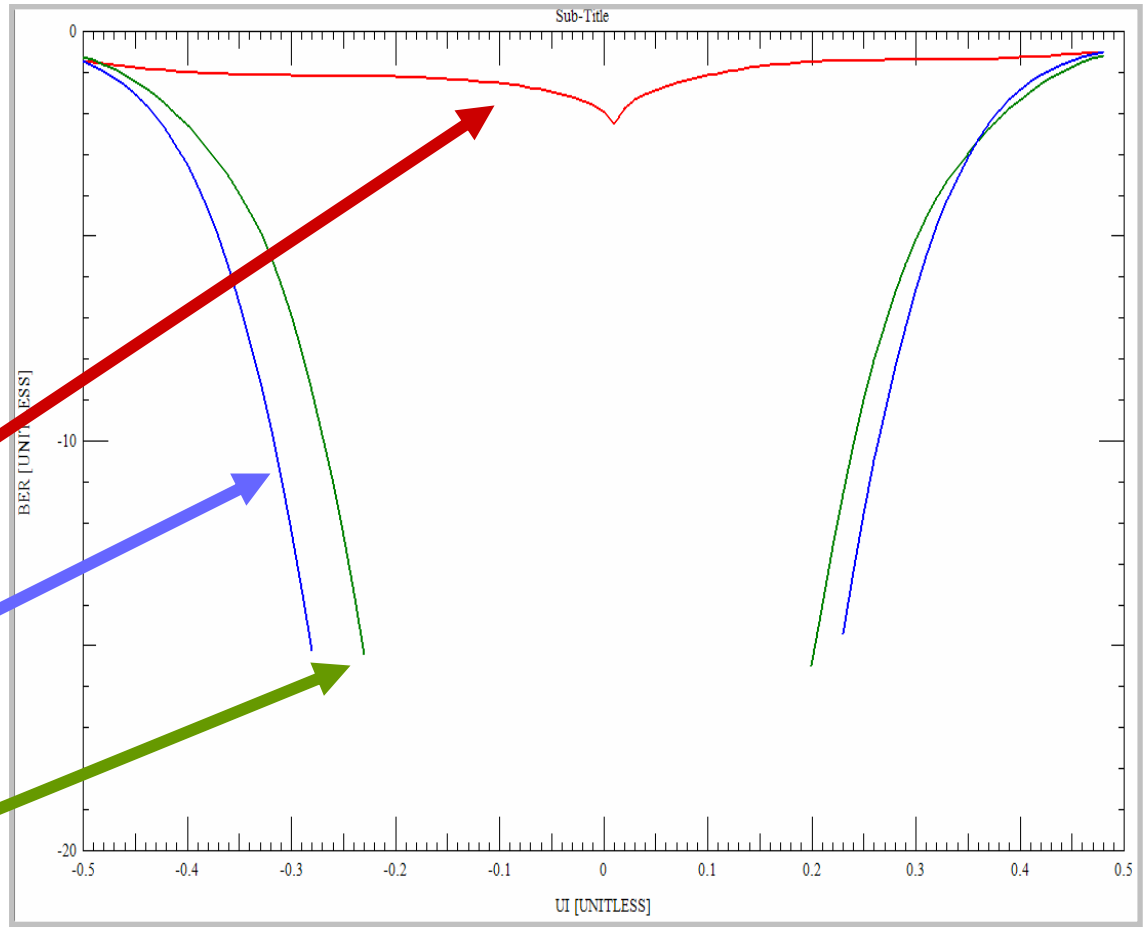
Test Cases and Results

3 Cases

No filters

FFE, no DFE

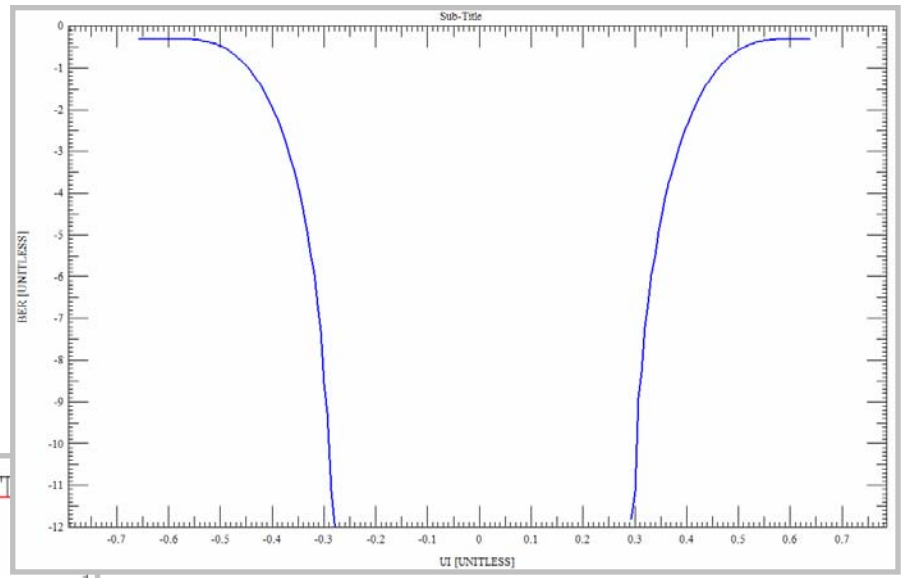
DFE model with FFE coeff=1

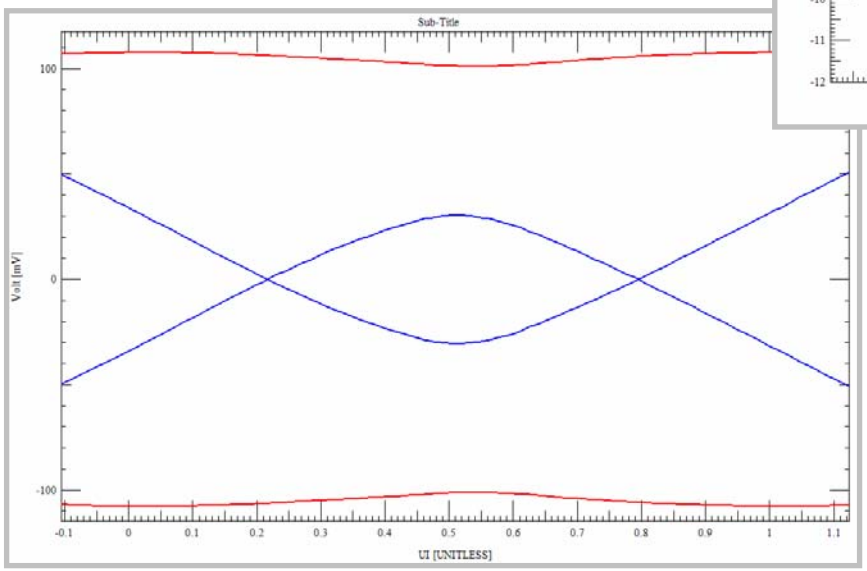


Test Cases and Results

Statistical Analysis

– DFE model with FFE coeff=1





Test Cases and Results

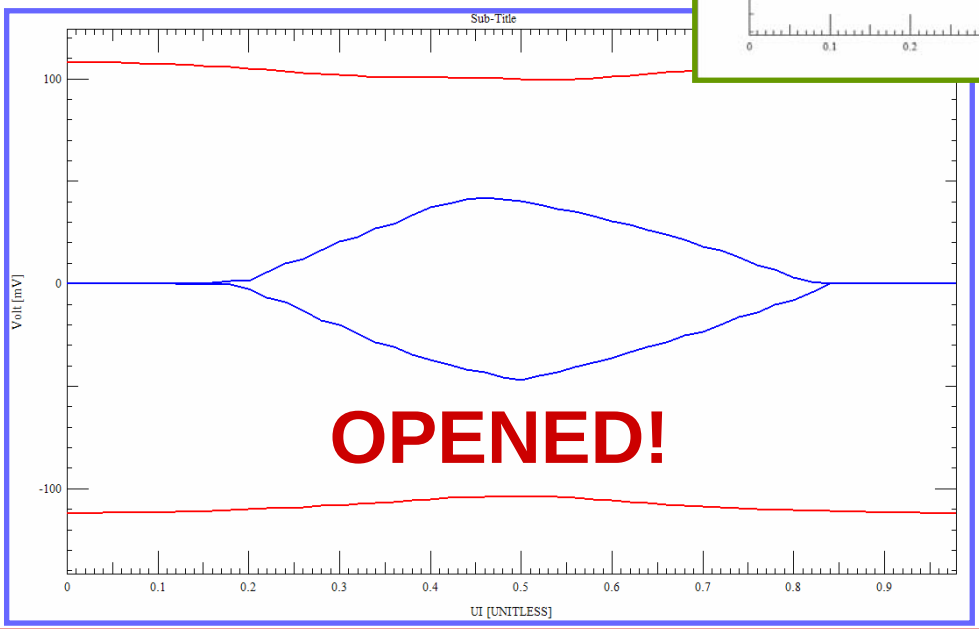
With Periodic Jitter (200ppm, 1MHz)

Case 1:

- DFE model with FFE coeff=1
- Add periodic jitter

Case2

- Add CDR model to previous test case



Next Steps

- Work with IBM for the real device correlations
- Work on the details for IBIS linkages
- Propose BIRD for API Standardization

We encourage more involvements from SERDES vendors, users and measurement vendors on this topic

Questions?