Proposal for modeling advanced SERDES

IBM, Cadence

June 2006

Presenters, Contributors

Presenters / Contributors

1. Joe Abler
   IBM Systems & Technology Group
   High Speed Serial Link Solutions
2. C. Kumar
   Architect
   Cadence Design Systems, Inc.

Supporting Contributors

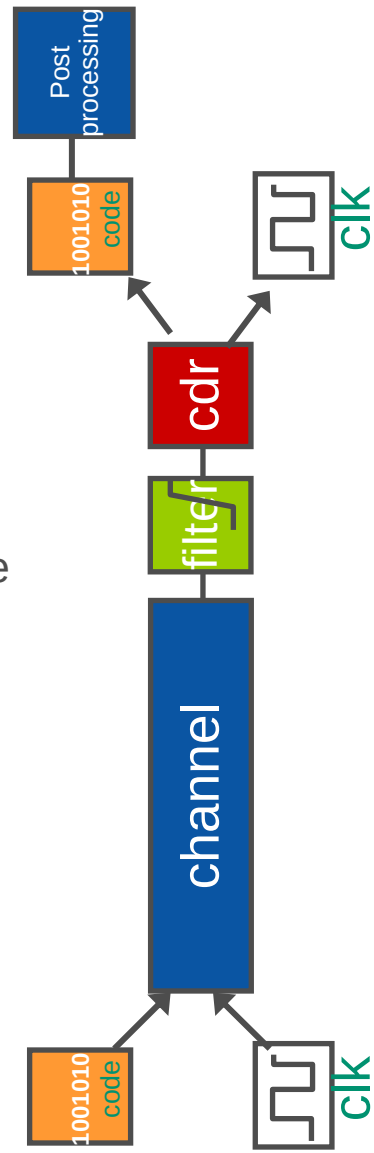
1. Richard Ward
   Texas Instruments
2. Hemant Shah
   Product Marketing
   Cadence Design Systems, Inc.

Agenda

- Introduction / Background
- The Advanced SERDES modeling challenge
- Current Approaches
- Need for system level modeling
  - Key modeling requirements
- Proposed solution
- Benefits

Advanced Serdes Modeling Challenges

- 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
- Architectural level exploration required
  - Algorithms are easy to represent and already exist at design level
  - They are typically modeled in higher level programming languages like C or Matlab
- These algorithms are very difficult to represent with traditional device modeling techniques
  - Long run times even if you can create them
- There is currently no industry-standard way to represent algorithms
  - IP suppliers have developed & distributed their own proprietary tools, increasing their support costs
  - No interoperability for systems company users



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

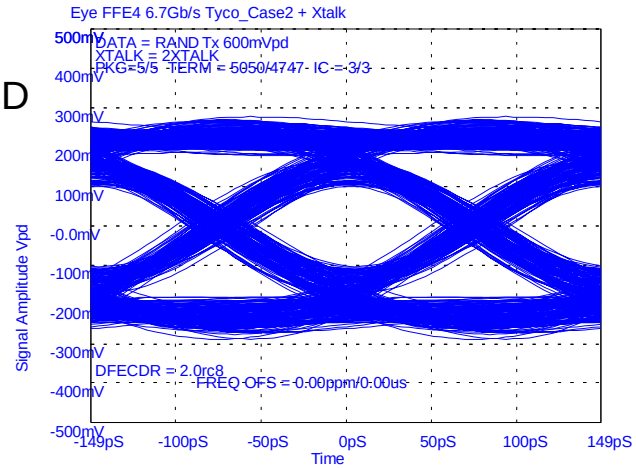
- **Adequate simulation time required to capture system level effects**
  - Data pattern dependencies
  - Crosstalk
- **Simulation results for example case**
  - Tyco Case 6
    - Includes Tyco xtalk channels
  - 10.3 Gbps operation
  - Random data
  - Only simulation length is varied
- **Modeling approach 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

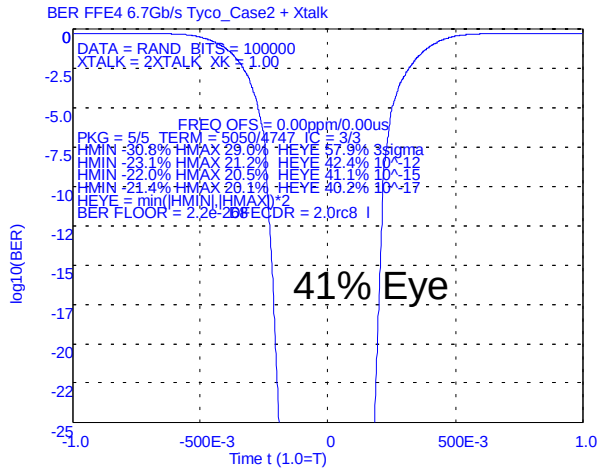
Sim length (bit times)	Sim time (minutes)	Heye margin (BER E-12)	Heye margin (BER E-15)	Heye margin (BER E-17)
100K	0.2	19.8%	17.9%	17.0%
1M	1.6	12.5%	11.3%	10.4%
10M	20.9	8.4%	5.8%	4.9%
100M	159	6.7%	2.9%	0.4%

Tx DCD Effects through Lossy Channel

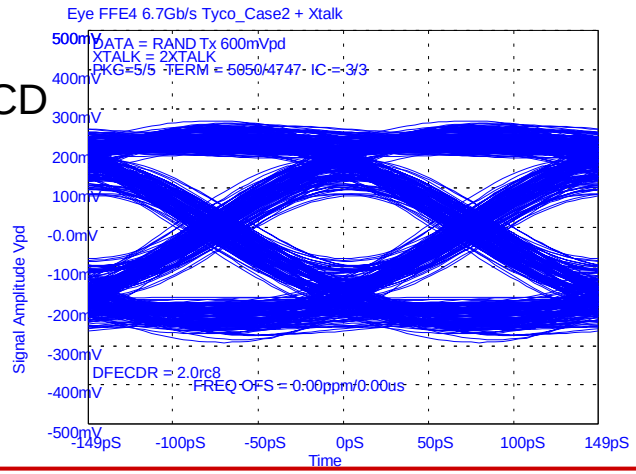
- Tx DCD (& resulting jitter amplification) can quickly close an eye...

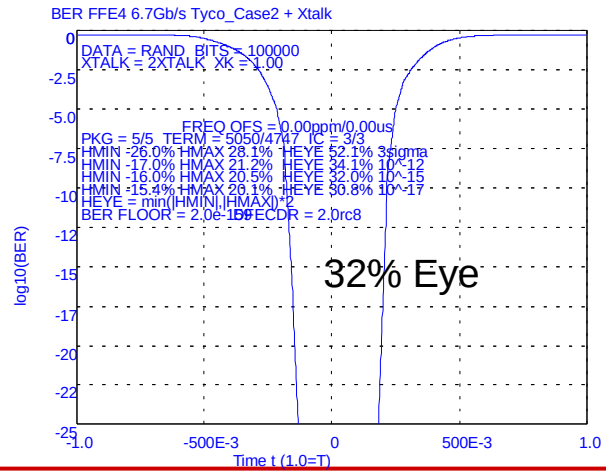
0% DCD





3.1% DCD



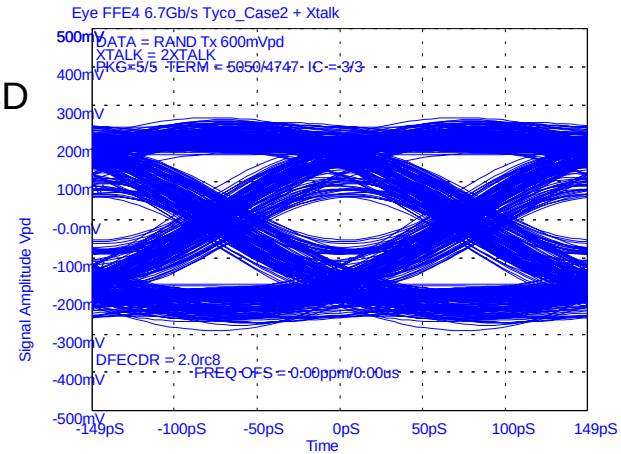


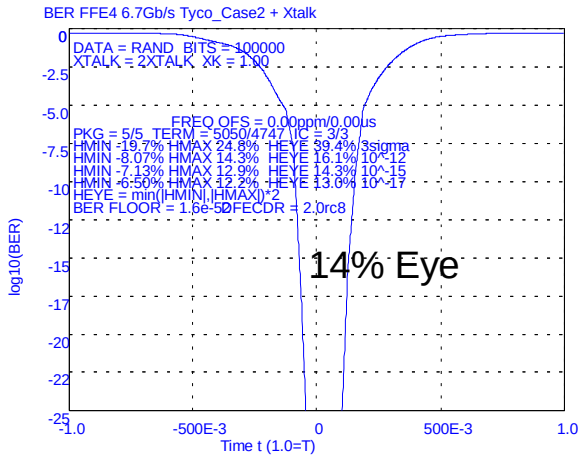
Tx DCD Effects through Lossy Channel

…Down to nothing!

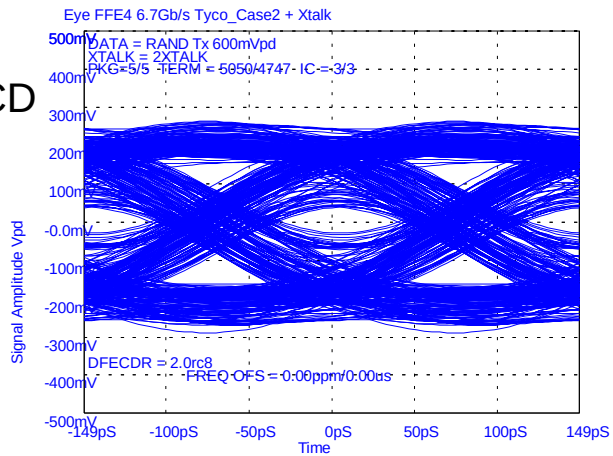
Control algorithms must be accurately modeled to analyze performance

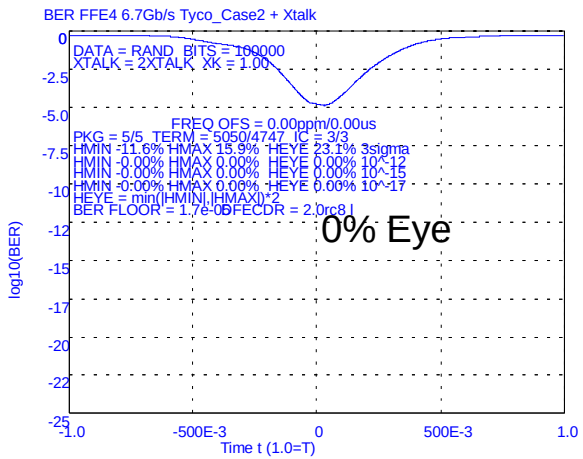
6.3% DCD





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, ..)	Limited	Yes
Filter Optimization	No	Yes
CDR	Partial	Yes
Jitter components	Partial	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

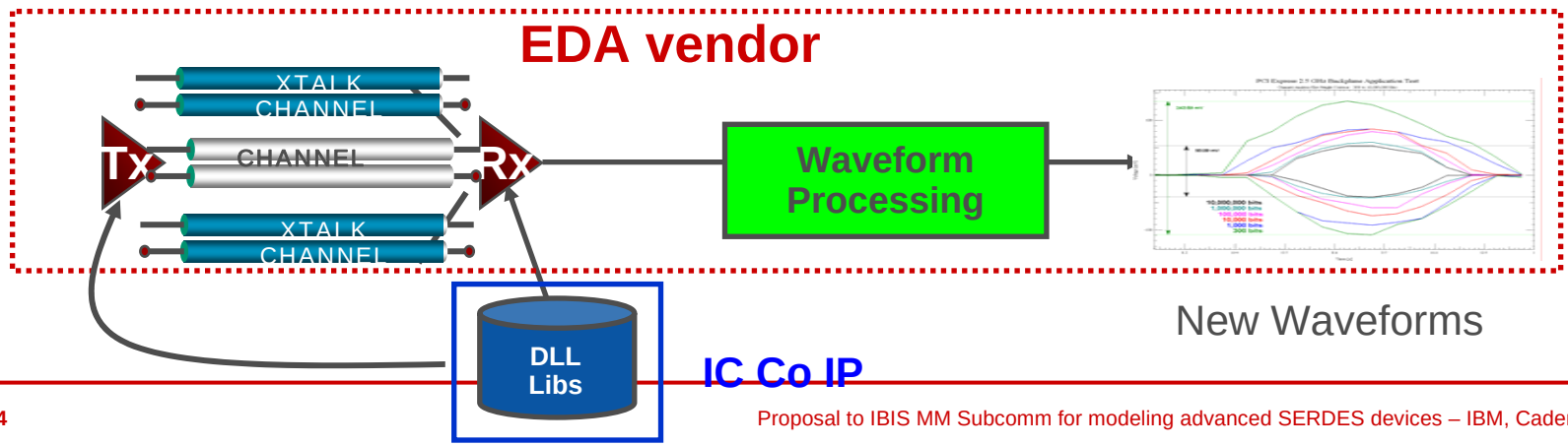
- Ability to capture complex ***algorithms***
  - DSP / Filter optimization: CDR, DFE, …
- Minimal model development time
- High accuracy (hardware correlated) with minimum simulation time
- Protection of IP (Silicon vendors)
- Architectural modeling
  - Ability to model & evaluate IP before silicon 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

Proposed Solution

- Allow IC companies to develop “executable” algorithm based models that plug into the system level simulator
- Algorithm methodology
  - Top Down design methodology
    - The level of modeling detail is controllable by the IP vendors
  - Prevalent in SERDES IP vendors companies
  - Enables architectural level exploration / experimentation
  - Enables compliance testing

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



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 config 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

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

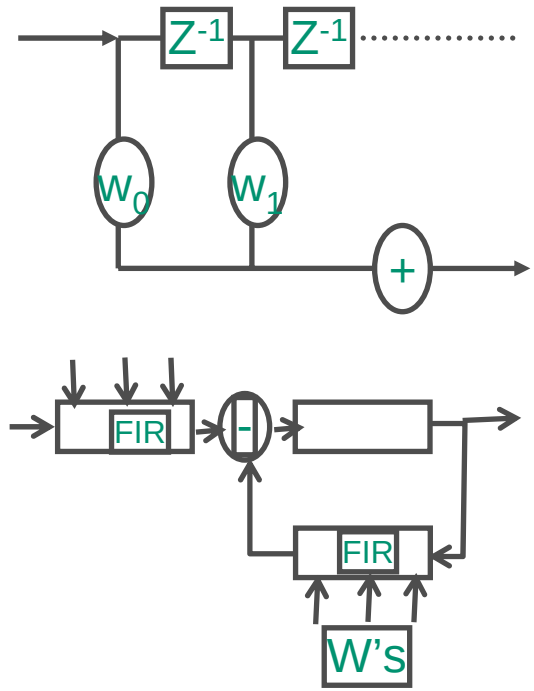
- long ***rx_getwave*** (double *wave_in, long size, double dt, double *clk, void *dll_server_param_obj, void *dll_client_param, [genchdllmsg_type **msg])
  - Input: Voltage at Rx input at specific times
  - Output: Modified Voltage, Clock tics, status

- long ***rx_close*** (void **ptr_2_dll_server_param_obj)
  - Clean up, exit

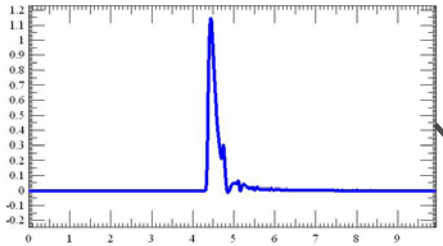
Note: items in [ ] are optional and can be 0(null)

Sample models

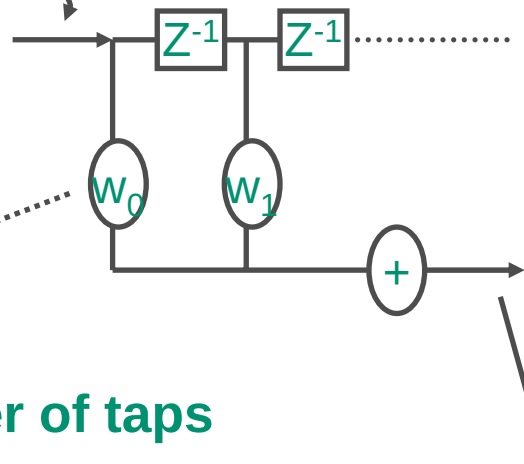
1. Chffefilt
   - Optimized Feed Forward Filter
2. Chdfefilt
   - Decision Feedback Filter
3. Chcdr
   - Clock and Data Recovery unit with Proportional Integral (pi) control



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))

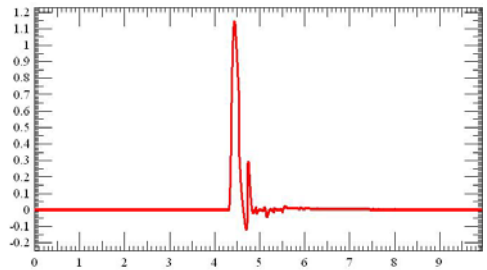
dll Name

Parameters

5 taps

Read pulse from ffein.txt

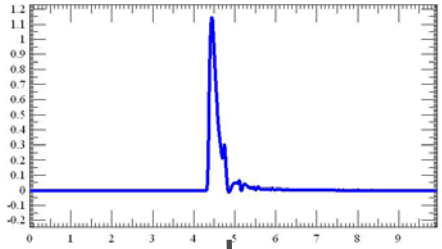
Write pulse from ffeout.txt

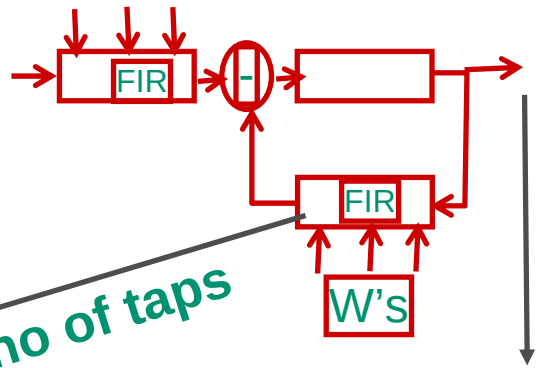


MMSE: Minimum Mean Square Error

Sample DFE Filter

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





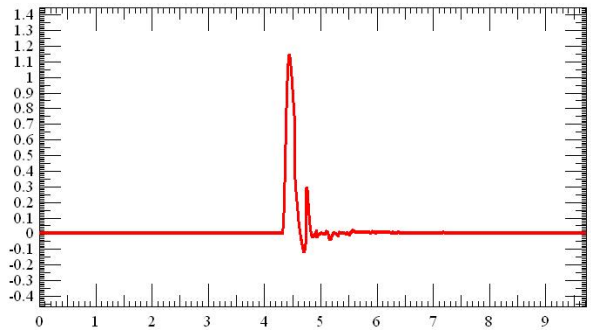
Adjustable no of taps

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

Dll Name

Parameters

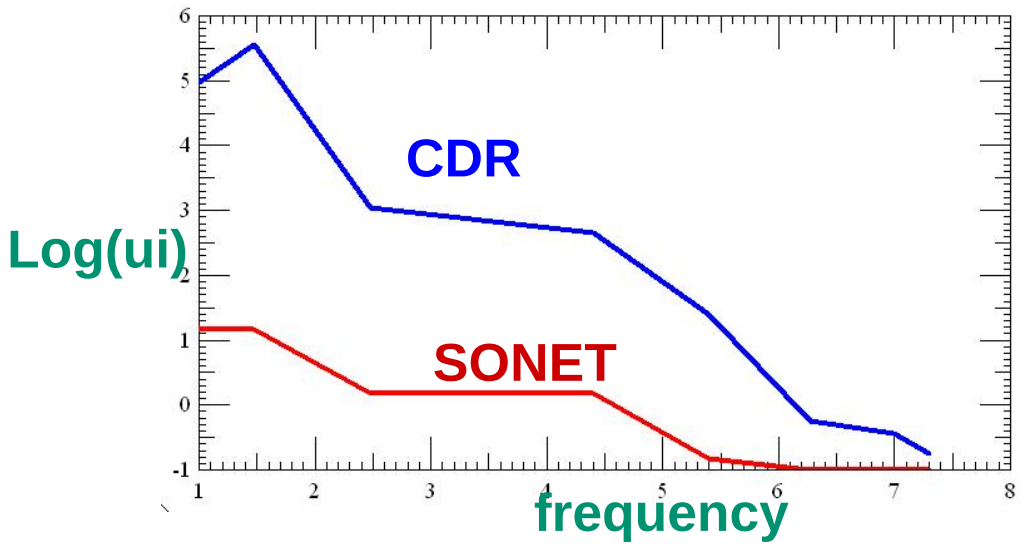
Backward # DFE taps



MMSE: Minimum Mean Square Error

Sample CDR model

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



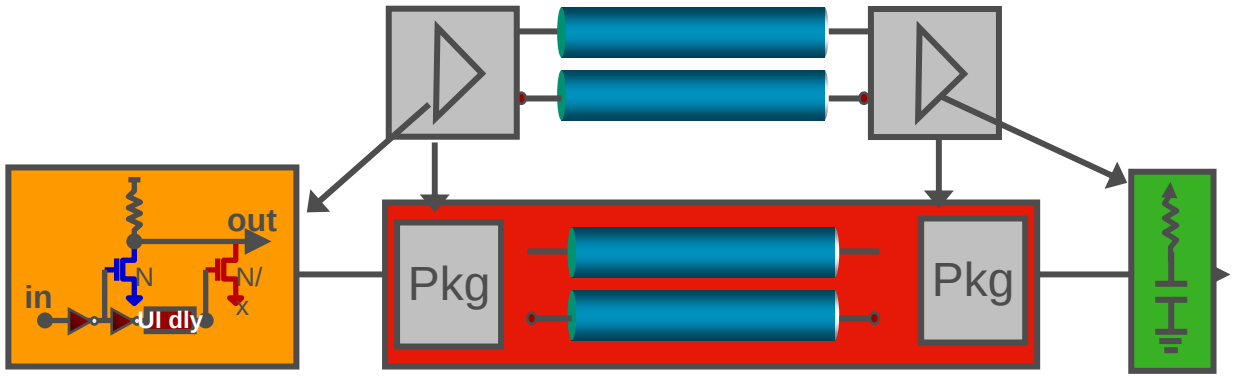
(chcdr (res ..) (corr_freq) (integ_corr_freq ..))

Parameters

dll Name

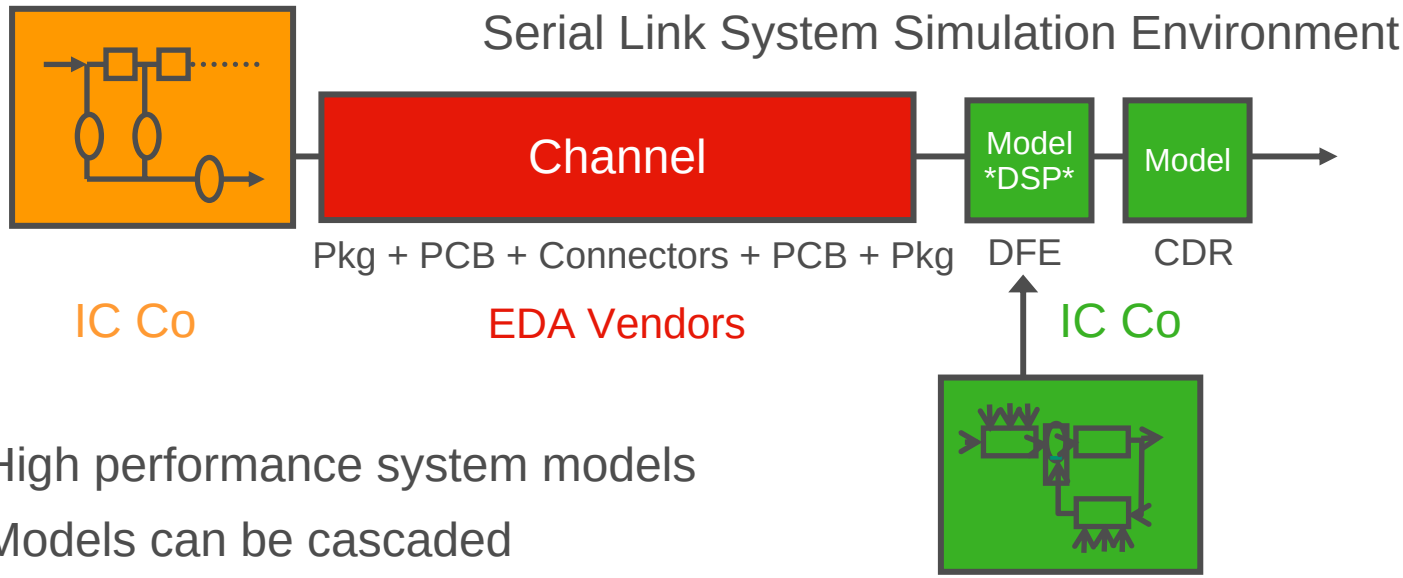
CDR resolution

Simulator – Model functional boundary
Present – traditional circuit models



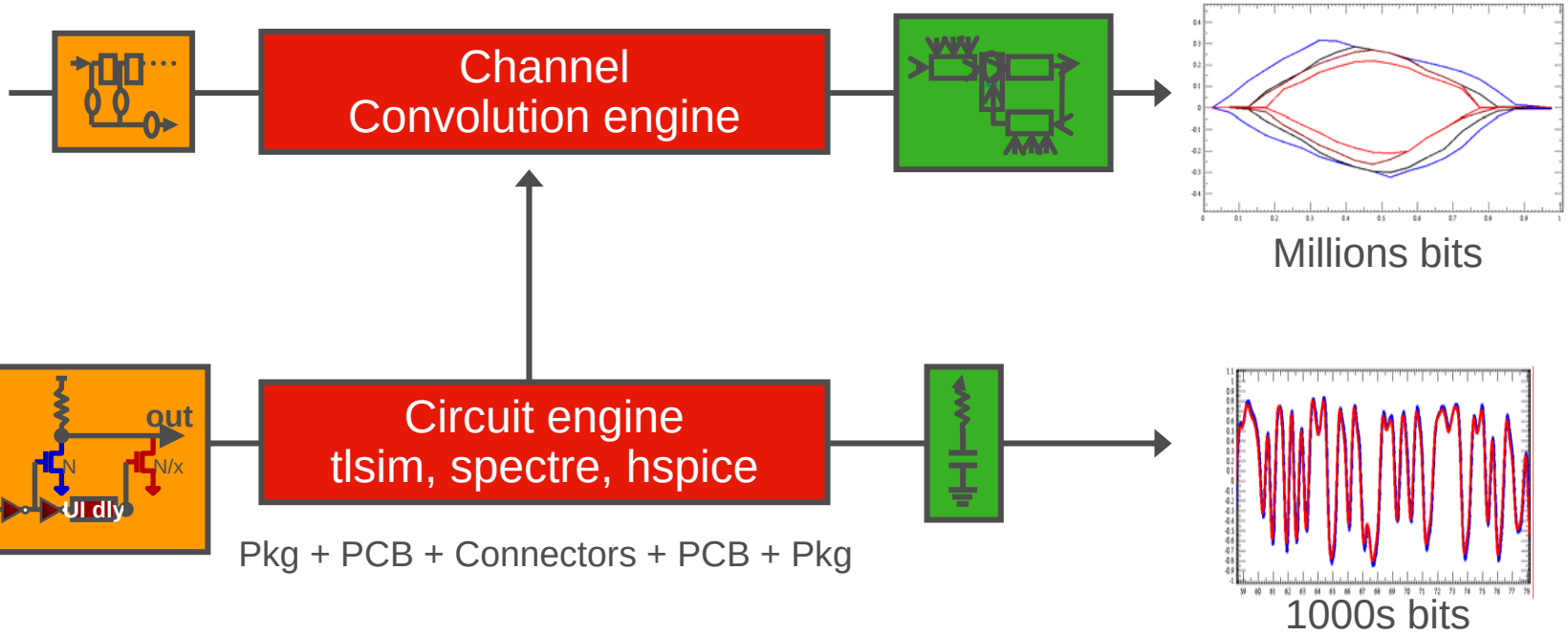
- Circuit models – behavioral, transistor, ..
- Simple models
  - Combination of Tx output stage with Rx load model
  - No jitter budget analysis
- Assumes open-eye system

Simulator – Model functional boundary
Proposed Algorithmic modeling environment

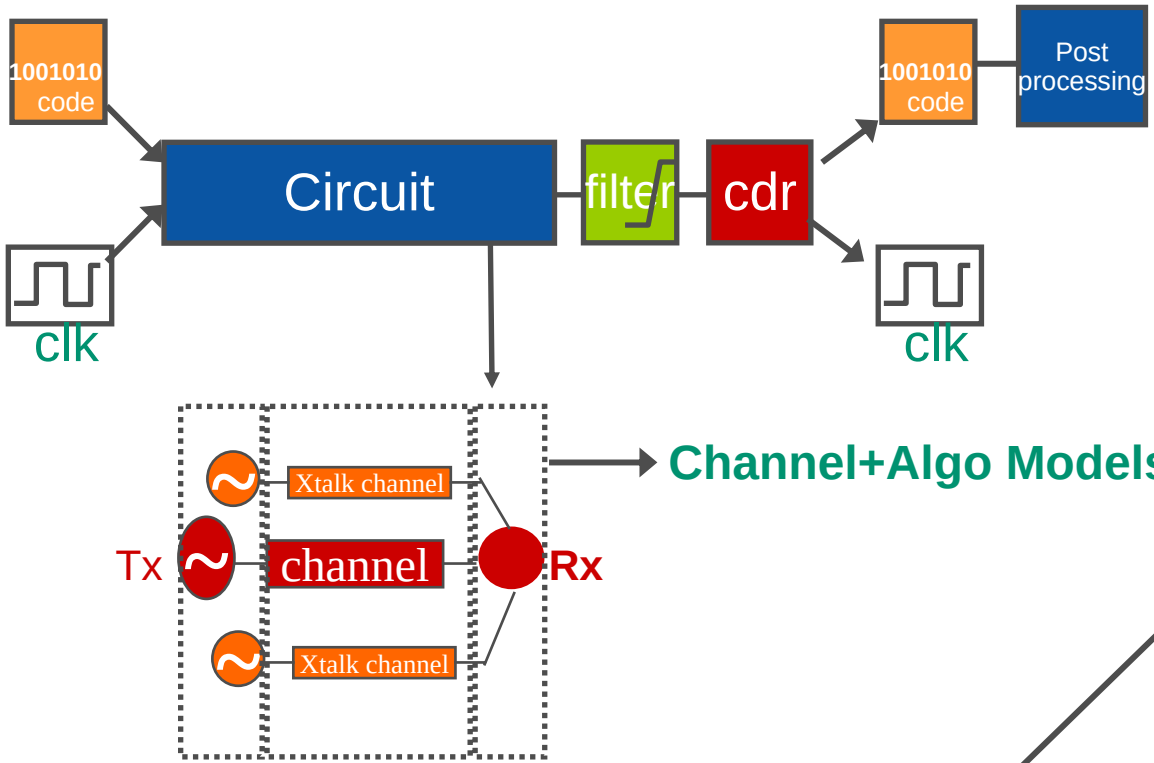


- High performance system models
- Models can be cascaded
- In-depth modeling of internal behavior
  - Filter blocks dynamically adapt to the channel
  - Optimal channel compensation (tap settings, ..)
  - Jitter injection, tolerance
- IP Protection

Simulator – Model functional boundary
Scalable engines, interoperable models



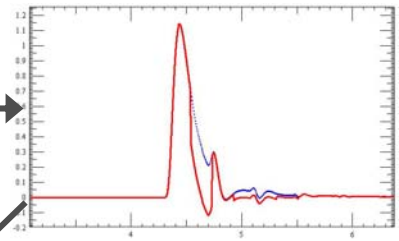
Channel – Bit by Bit simulation

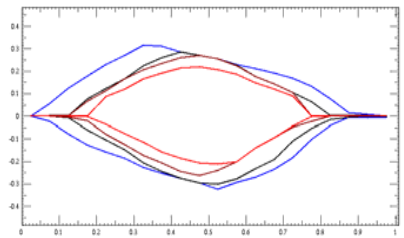


Channel
"Bit by Bit simulation"

characterize

Channel+Algo Models





input

$x(n)$

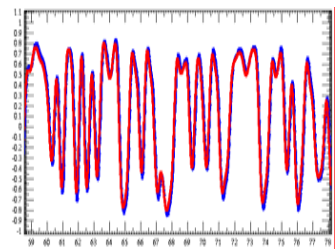
Convolution Engine

$h(n)$

output

$y(n)$

$y(n) = x(n) * h(n)$



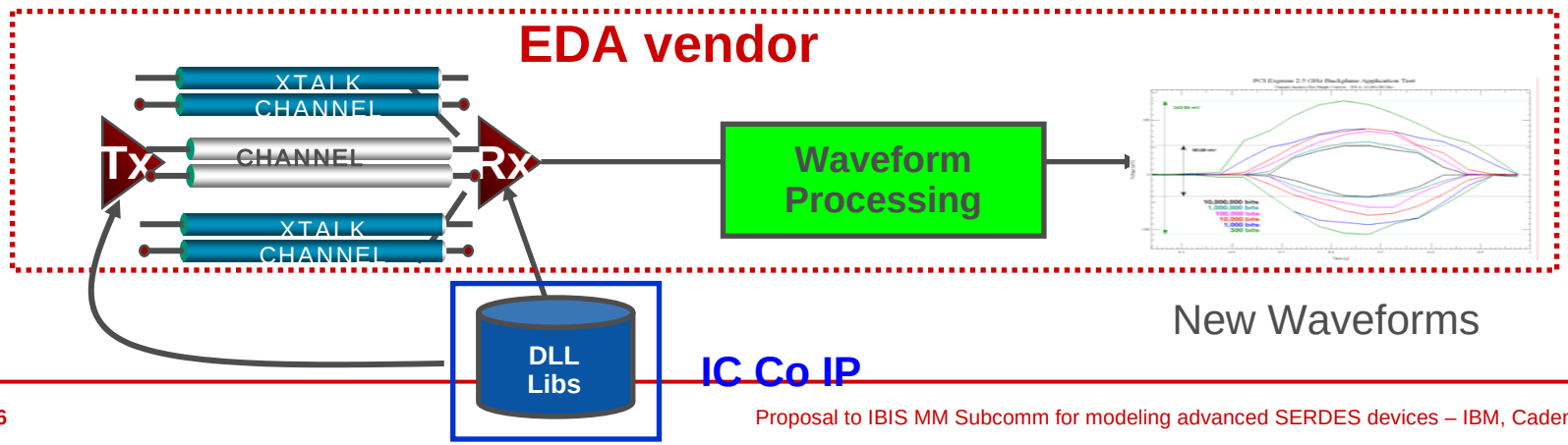
Circuit Correlation

Simulation tools – IP Vendor tasks

- Base Simulator functionality
  - Convolution engine
  - Bit Error Rate Prediction (BER)
  - GUI / Result Displays (Eye, BER Mask, Bathtub)
  - Jitter injection
  - Input bit generation
  - Input characterization
  - Support API
- IP Vendor
  - Wrap algorithms in API
  - Filter synthesis / Tap optimization
    - DFE / FFE architecture
    - Other equalization & signal processing techniques
  - CDR algorithm
  - Accept channel characterization from Simulator platform
  - Pass back modified channel characterization, waveform data

Proposed Solution - Benefits

- High level model development
  - Can easily capture complex ***algorithms*** (DSP / Filter optimization: CDR, DFE)
  - Models can be developed, distributed, evaluated at silicon architecture phase
- Provides interoperability of advanced models for systems company users
- Leverages IC Companies' model development methodology
- Provides Protection of IP
- Best possible performance
- Extensible for future needs



Summary

- Current device modeling techniques are inadequate for modeling advanced SERDES devices
- Behavioral modeling may provide short-term extension, but
  - Limitations remain
  - Why make a paradigm shift with limited potential to meet future or even current needs?
- Algorithmic modeling is a communication centric design paradigm
- We request the IBIS committee to consider defining a higher level algorithmic modeling approach
  - With a standardized API accessing compiled models via DLLs