INTRODUCTION TO IBIS-AMI

Todd Westerhoff, SiSoft

Mike LaBonte, SiSoft

Walter Katz, SiSoft

SPEAKERS

Todd Westerhoff

VP, Semiconductor Relations, SiSoft
twesterh@sisoft.com | www.sisoft.com

Todd has over 37 years of experience in electronic system modeling and simulation, including 20 years in signal integrity. He is responsible for SiSoft's activities working with semiconductor vendors to develop high-quality simulation models and has been heavily involved with the IBIS-AMI modeling specification since its inception. He has held senior technical and management positions for Cisco and Cadence and worked as an independent signal integrity consultant.

Mike LaBonte

Senior IBIS-AMI Specialist, SiSoft
mlabonte@sisoft.com | www.sisoft.com

An EDA software developer, Mike LaBonte has 29 years of signal integrity experience with 10 years of prior electronic thermal and reliability analysis experience. Since 2011 Mike has developed advanced IBIS-AMI model evaluation capabilities at SiSoft, as well as portions of the Quantum Channel Designer product line. Mike has held board positions in the IBIS Open Forum since 2009 and currently serves as its chairman.

Agenda

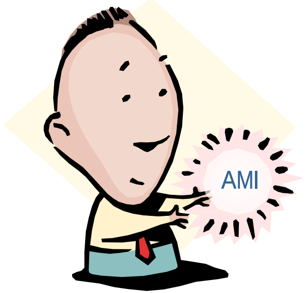
- Why IBIS-AMI?
- IBIS-AMI basics
- Optimizing Equalization
- Statistical simulation with AMI
- Time-Domain simulation with AMI
- IBIS-AMI flows
- Clocks and jitter
- Trusting simulation results
- Useful tips and tricks

Why IBIS-AMI?

Why Standardized SerDes IP Models?

- **Target serial link error rates < 1e-12**
- **Existing simulation tools won't work**
  - SPICE simulations ~100 bits, can't model complex EQ
  - "Traditional" IBIS can't model complex EQ
  - SerDes vendor simulators are proprietary
- **Need accurate and statistically significant way to model SerDes IP in commercial EDA simulators**

SerDes Modeling Goals



- **Interoperable: different vendor models work together**
- **Portable: one model runs in multiple simulators**
- **Flexible: support Statistical and Time-Domain simulation**
- **High Performance: simulate a million bits per CPU minute**
- **Accurate: high correlation to simulations / measurement**
- **Secure: represent IP behavior without exposing internal details**

IBIS Algorithmic Modeling Interface (IBIS-AMI)

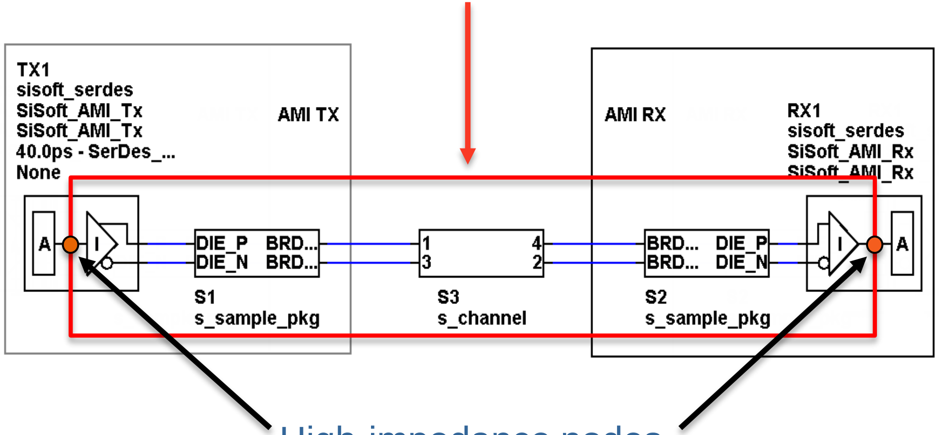
- Modeling specification maintained by the IBIS Open Forum
- Proposed in 2007, adopted as part of IBIS 5.0 in 2008
- Supported by most (if not all) commercial EDA simulators
- Supported by (some) proprietary vendor in-house simulators
- Multiple model development environments available
- Hundreds of different AMI models currently in use
- More info: www.ibis.org

IBIS-AMI Basics

IBIS-AMI Assumptions

- **SerDes channels can be broken into two parts for analysis:**
  - Analog (electrical) and Algorithmic
- **TX output driver & RX input termination are isolated from their respective equalization through a "high-impedance" node**
- **Analog channel can be considered linear and time-invariant (LTI)**

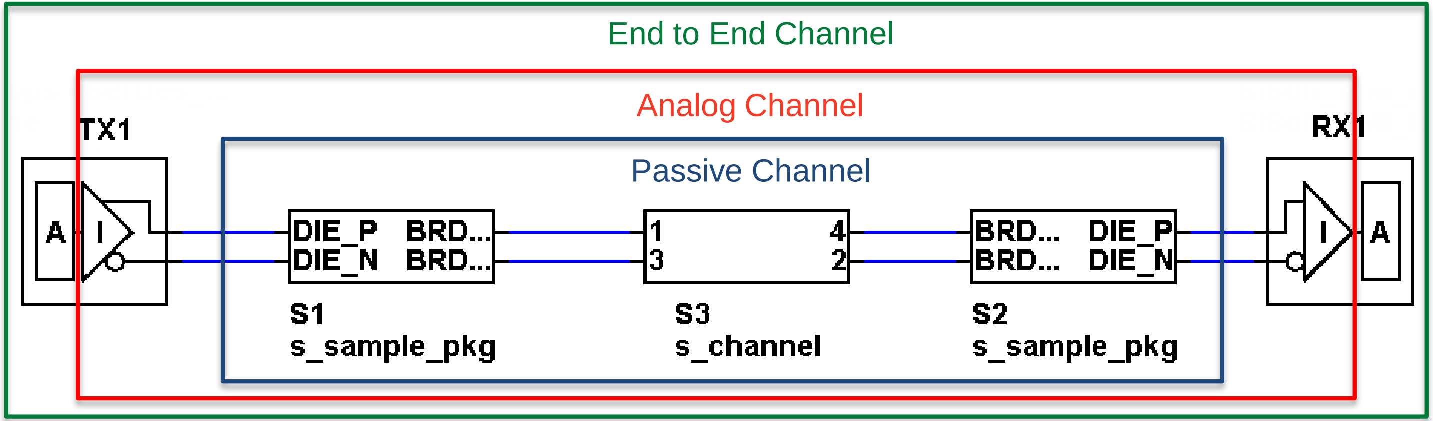
Analog channel



High-impedance nodes

http://ibis.org/ver6.1/ver6_1.pdf, page 170

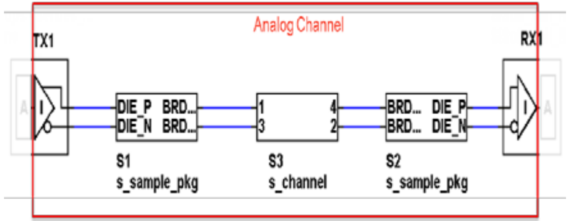
IBIS-AMI Channel Terminology



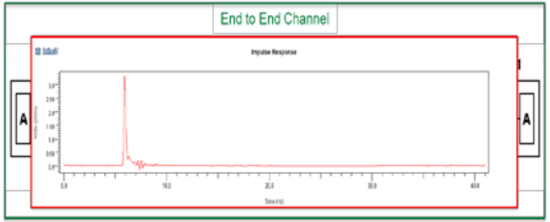
- Circuit simulation techniques are used for the analog channel
- Signal processing techniques are used for the end to end channel

IBIS-AMI Analysis Stages

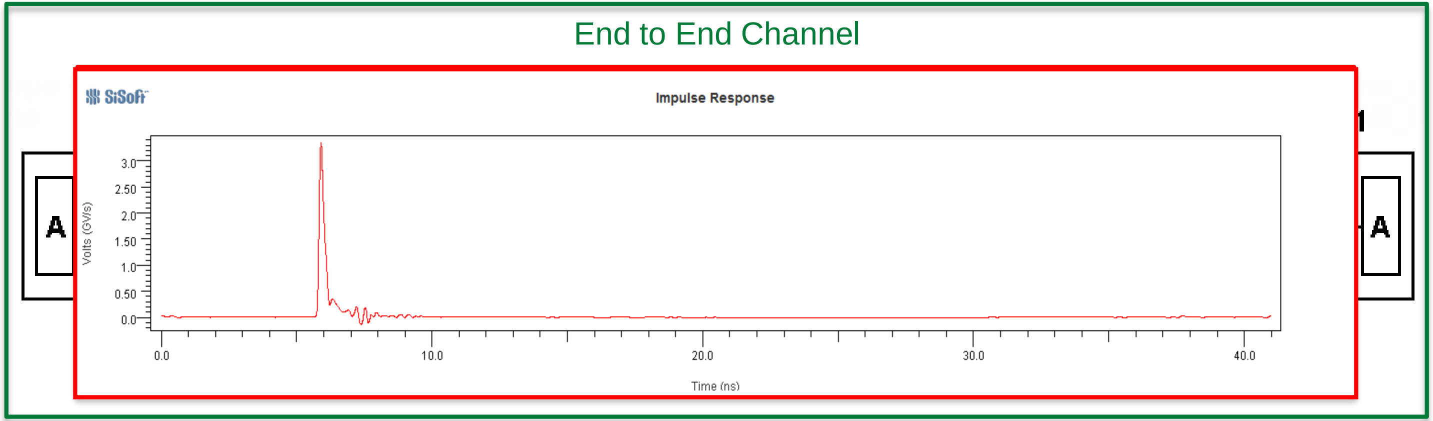
- **Network Characterization (Circuit Simulation)**
  - Inputs:
    - Passive network
    - IBIS analog models
  - Time-domain or frequency-domain
  - Derives analog channel impulse response
  - Analog effects (impedance, reflections) MUST be included in impulse response



- **Channel Simulation (Signal Processing)**
  - Inputs:
    - Analog channel impulse response
    - User settings for EQ & Clock Recovery
    - IBIS-AMI algorithmic models
  - Statistical and/or Time-Domain simulation depending on simulator & model capabilities



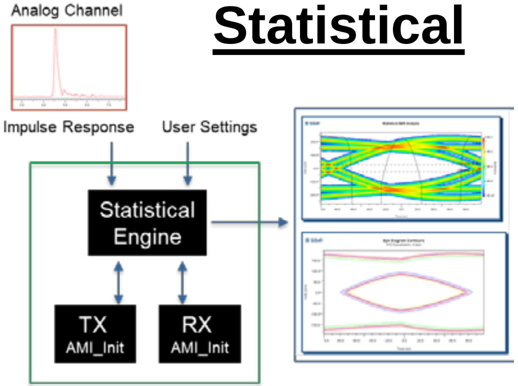
IBIS-AMI Analysis Stages



- **Network Characterization (Circuit Simulation)**
- **Channel Simulation (Signal Processing)**
  - Statistical Simulation
  - Time-Domain Simulation

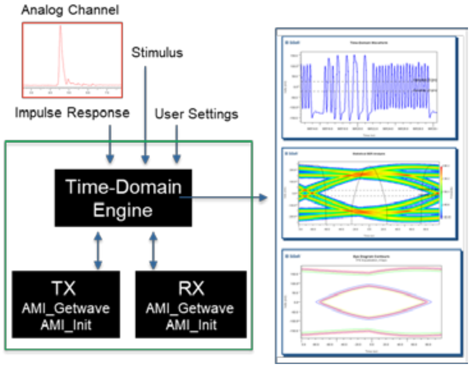
Channel Simulation Types

Statistical



- Computes eye directly from step/pulse response
- Probabilities <1e-45
- EQ is static (assumed LTI)

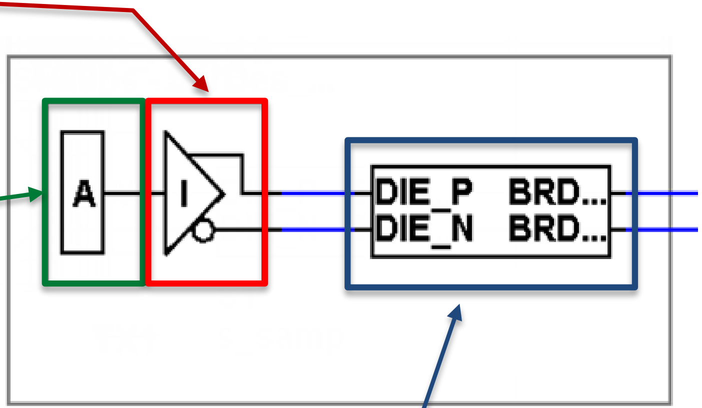
Time-Domain



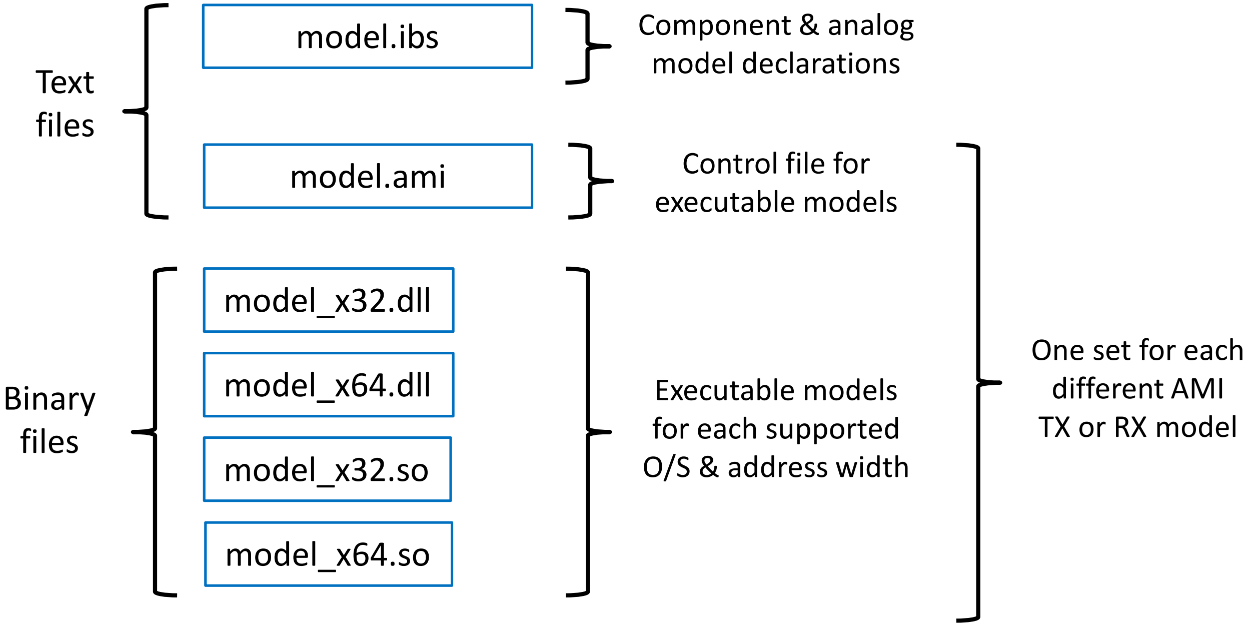
- Computes response based on specific input patterns
- Probabilities ~ 1e-6 to 1e-8
- EQ is adaptive (can be non-LTI)

IBIS-AMI Model Components

- **Analog model**
  - [Model] keyword in .ibs file
  - Tabular V/I & V/T data
  - Assumed to be LTI
- **Algorithmic model**
  - [Algorithmic Model] keyword in .ibs file
  - .ami file describes capabilities & inputs
- **Package model**
  - Can be described in .ibs file
  - Often supplied separately as .sNp file



IBIS-AMI File Set



Text files

model.ibs

Component & analog model declarations

model.ami

Control file for executable models

Binary files

model_x32.dll

model_x64.dll

model_x32.so

model_x64.so

Executable models for each supported O/S & address width

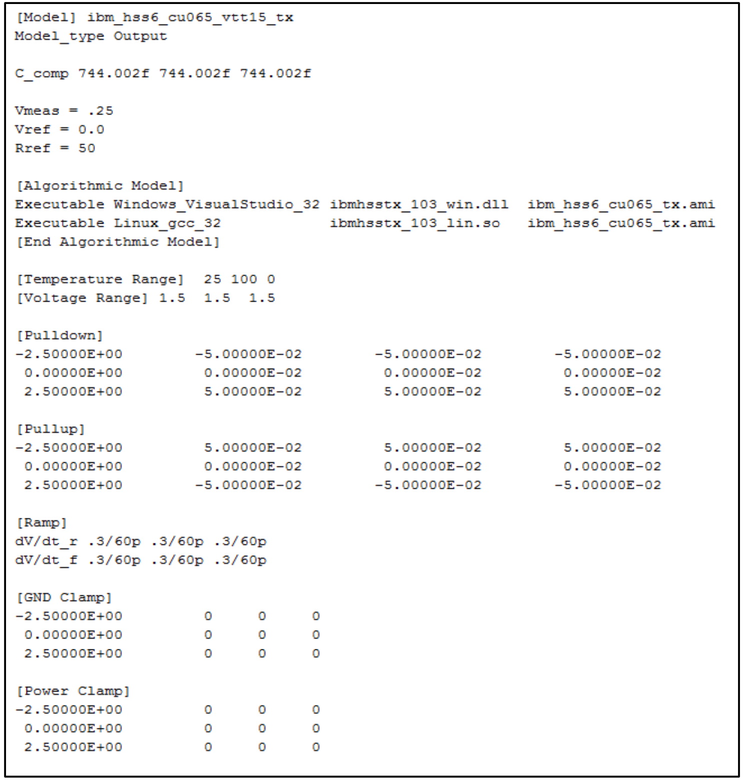
One set for each different AMI TX or RX model

.ibs File for AMI Model

Analog model declaration

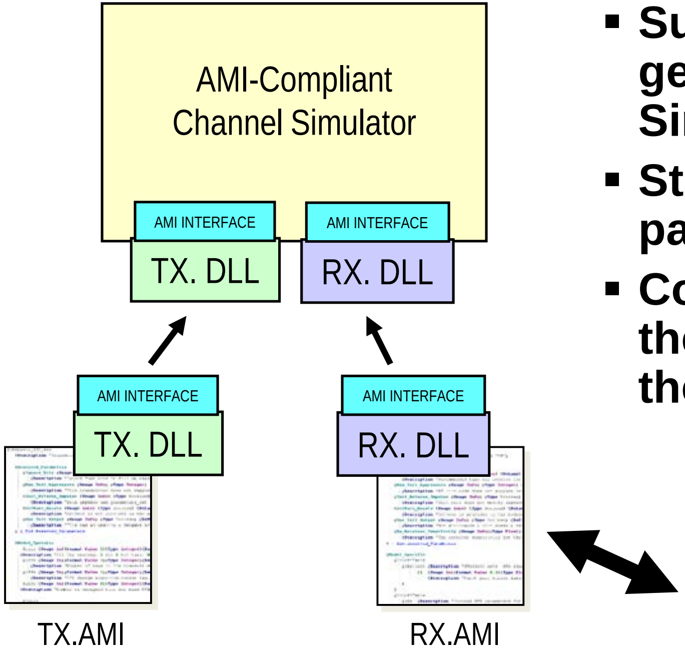
Executable models & control files

Analog model characteristics

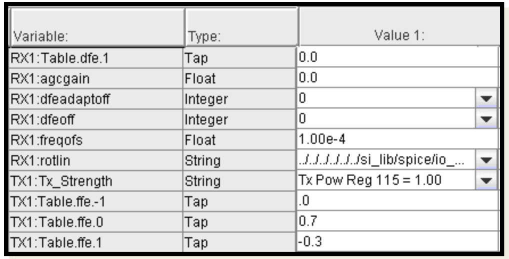


```
[Model] ibm_hss6_cu065_vtt15_tx
Model_type Output

C_comp 744.002f 744.002f 744.002f

Vmeas = .25
Vref = 0.0
Rref = 50

[Algorithmic Model]
Executable Windows_VisualStudio_32 ibmhsstx_103_win.dll  ibm_hss6_cu065_tx.ami
Executable Linux_gcc_32            ibmhsstx_103_lin.so   ibm_hss6_cu065_tx.ami
[End Algorithmic Model]

[Temperature Range]  25 100 0
[Voltage Range] 1.5  1.5  1.5

[Pulldown]
-2.50000E+00        -5.00000E-02        -5.00000E-02        -5.00000E-02
 0.00000E+00         0.00000E-02         0.00000E-02         0.00000E-02
 2.50000E+00         5.00000E-02         5.00000E-02         5.00000E-02

[Pullup]
-2.50000E+00         5.00000E-02         5.00000E-02         5.00000E-02
 0.00000E+00         0.00000E-02         0.00000E-02         0.00000E-02
 2.50000E+00        -5.00000E-02        -5.00000E-02        -5.00000E-02

[Ramp]
dV/dt_r .3/60p .3/60p .3/60p
dV/dt_f .3/60p .3/60p .3/60p

[GND Clamp]
-2.50000E+00         0     0     0
 0.00000E+00         0     0     0
 2.50000E+00         0     0     0

[Power Clamp]
-2.50000E+00         0     0     0
 0.00000E+00         0     0     0
 2.50000E+00         0     0     0
```

IBIS-AMI Algorithmic Models

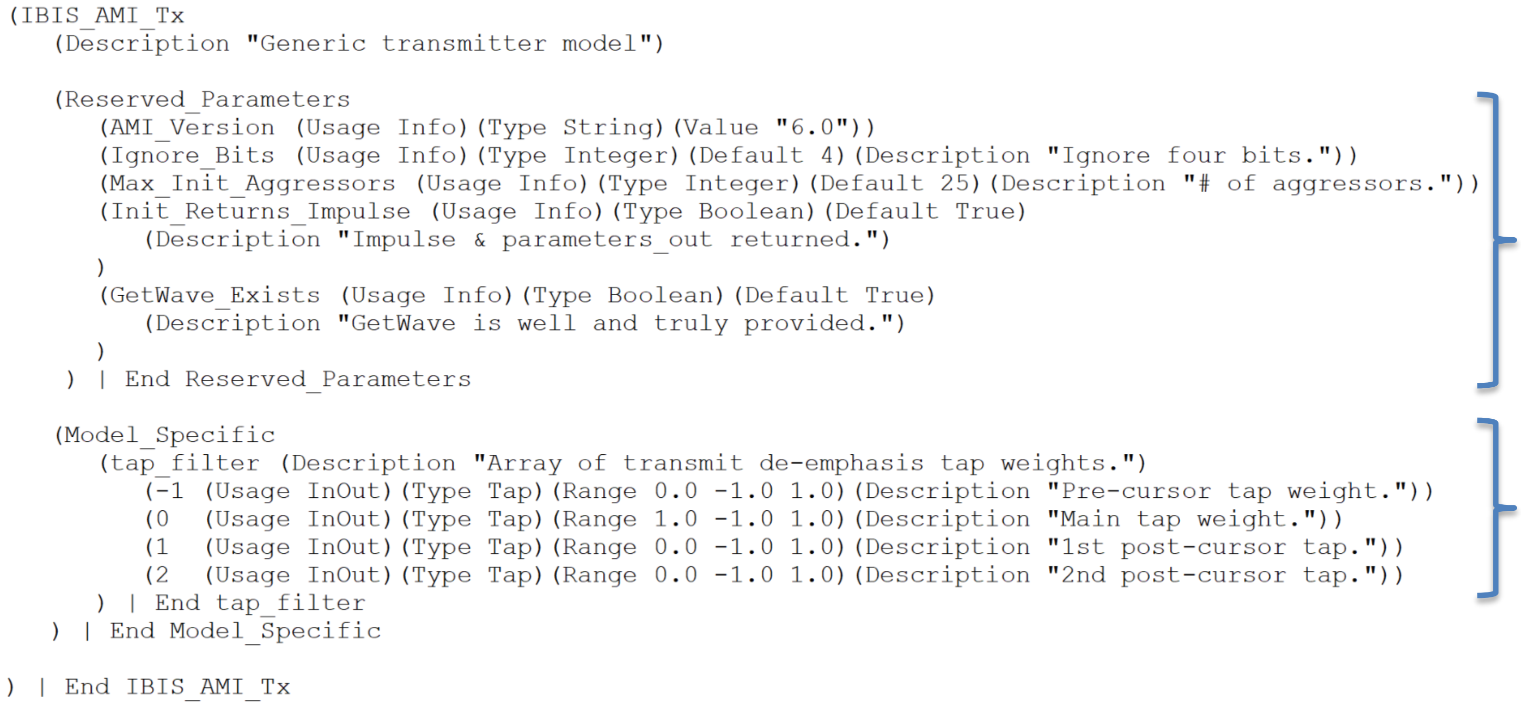


- **Supplied as binary code (.DLL) that gets linked into the Channel Simulator at runtime**
- **Standardized entry points and data passed to/from the model**
- **Control (.AMI) file lists what features the model supports & what controls the user can set**



Variable:	Type:	Value 1:
RX1:Table.dfe.1	Tap	0.0
RX1:agcgain	Float	0.0
RX1:dfeadaptoff	Integer	0
RX1:dfeoff	Integer	0
RX1:freqofs	Float	1.00e-4
RX1:rotlin	String	../../../../../../si_lib/spice/io_...
TX1:Tx_Strength	String	Tx Pow Reg 115 = 1.00
TX1:Table.ffe.-1	Tap	.0
TX1:Table.ffe.0	Tap	0.7
TX1:Table.ffe.1	Tap	-0.3

Example AMI File

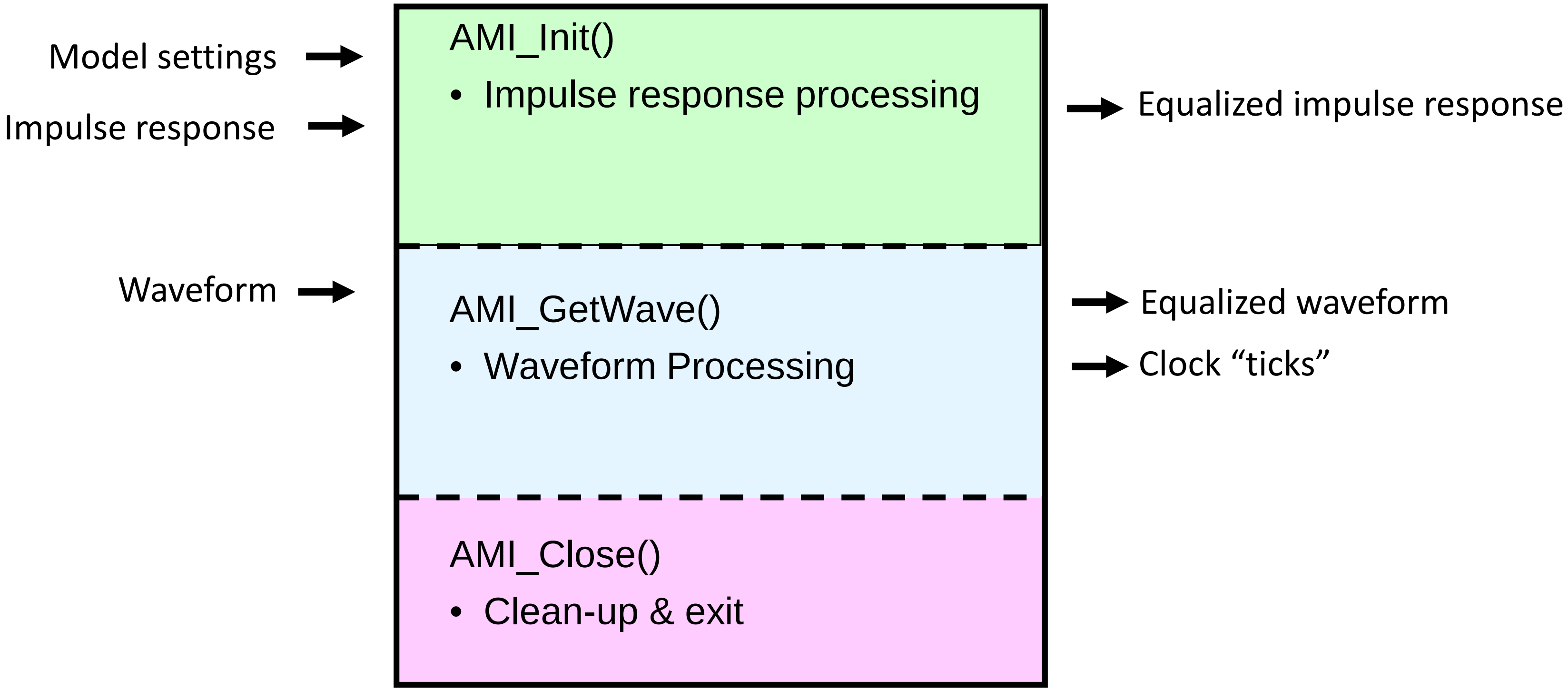


```
(IBIS_AMI_Tx
   (Description "Generic transmitter model")

   (Reserved_Parameters
      (AMI_Version (Usage Info)(Type String)(Value "6.0"))
      (Ignore_Bits (Usage Info)(Type Integer)(Default 4)(Description "Ignore four bits."))
      (Max_Init_Aggressors (Usage Info)(Type Integer)(Default 25)(Description "# of aggressors."))
      (Init_Returns_Impulse (Usage Info)(Type Boolean)(Default True)
         (Description "Impulse & parameters_out returned.")
      )
      (GetWave_Exists (Usage Info)(Type Boolean)(Default True)
         (Description "GetWave is well and truly provided.")
      )
    ) | End Reserved_Parameters

   (Model_Specific
      (tap_filter (Description "Array of transmit de-emphasis tap weights.")
         (-1 (Usage InOut)(Type Tap)(Range 0.0 -1.0 1.0)(Description "Pre-cursor tap weight."))
         (0  (Usage InOut)(Type Tap)(Range 1.0 -1.0 1.0)(Description "Main tap weight."))
         (1  (Usage InOut)(Type Tap)(Range 0.0 -1.0 1.0)(Description "1st post-cursor tap."))
         (2  (Usage InOut)(Type Tap)(Range 0.0 -1.0 1.0)(Description "2nd post-cursor tap."))
      ) | End tap_filter
   ) | End Model_Specific

) | End IBIS_AMI_Tx
```

Reserved Parameters

Model Specific Parameters

Executable Model Architecture



Model settings →

Impulse response →

AMI_Init()

- Impulse response processing

→ Equalized impulse response

Waveform →

AMI_GetWave()

- Waveform Processing

→ Equalized waveform

→ Clock “ticks”

AMI_Close()

- Clean-up & exit

Understanding the .DLL Interface

The AMI specification defines three standard entry points and calling signatures for .DLL models:

- **AMI_Init()**
  - REQUIRED. Called only once for each model at the start of each simulation run
  - Inputs: impulse response, model parameter string, memory pointers
  - Parses model parameter string and sets up model options
  - Allocates and manages any persistent memory used by the model
  - Optionally equalizes the impulse response and returns result in place in RAM
  - Optionally returns Parameters_Out data
  - Must be re-entrant, as multiple models and simulations are run simultaneously

Understanding the .DLL Interface

- **AMI_GetWave()**
  - OPTIONAL. When present, GetWave_exists is set to TRUE in the .AMI file
  - Inputs: time-domain waveform passed in as individual blocks of data
  - Called multiple times during time-domain analysis
  - Sliding window algorithm used to optimize simulation performance and memory requirements, must be supported by compliant .DLL’s
  - Returns equalized waveform and (in the case of RX) array of clock tick times
  - Optionally returns Parameters_Out data
  - Must be re-entrant, as multiple models and simulations are run simultaneously

- **AMI_Close()**
  - REQUIRED. Called only once for each model at the end of each simulation run
  - Inputs: none
  - Responsible for releasing memory allocated by model

Summary: IBIS-AMI Basics

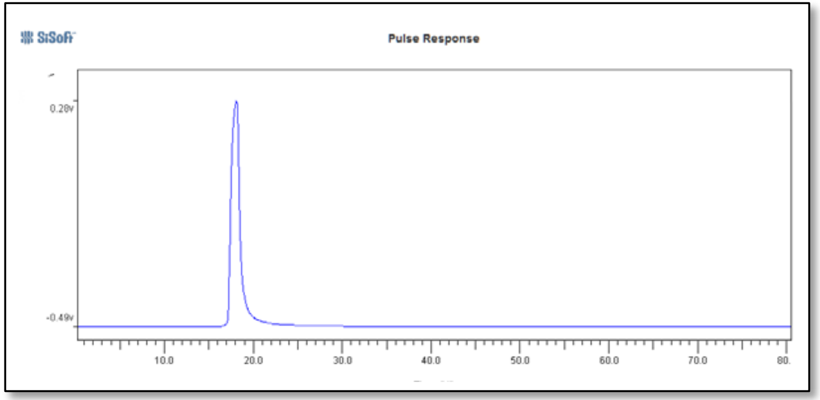
- AMI assumes serial links can be separated into analog and algorithmic portions that can be analyzed sequentially
- Models & analysis have two stages: analog & algorithmic
- Two types of channel simulation: Statistical & Time-Domain
- Executable models are supplied as DLLs linked into the simulator at runtime, with an associated .AMI control file

Summary: IBIS-AMI Basics

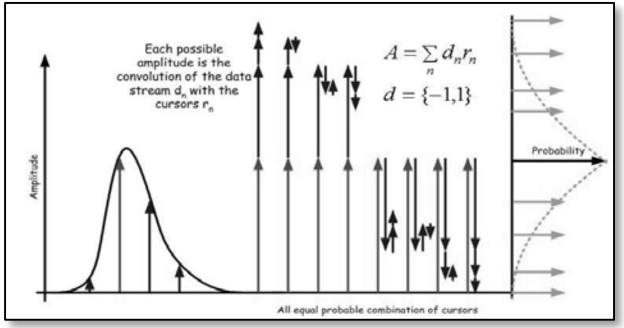
- The AMI specification defines the programming interface that governs how DLL models interact with the host EDA simulator
- AMI models have two modeling methods: impulse and waveform
- AMI files tell the simulator what features the DLL supports
- AMI have file two sections: reserved and model-specific

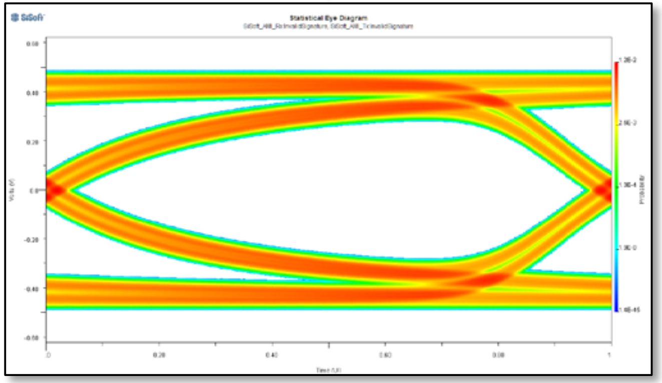
Optimizing Equalization

Statistical Simulation



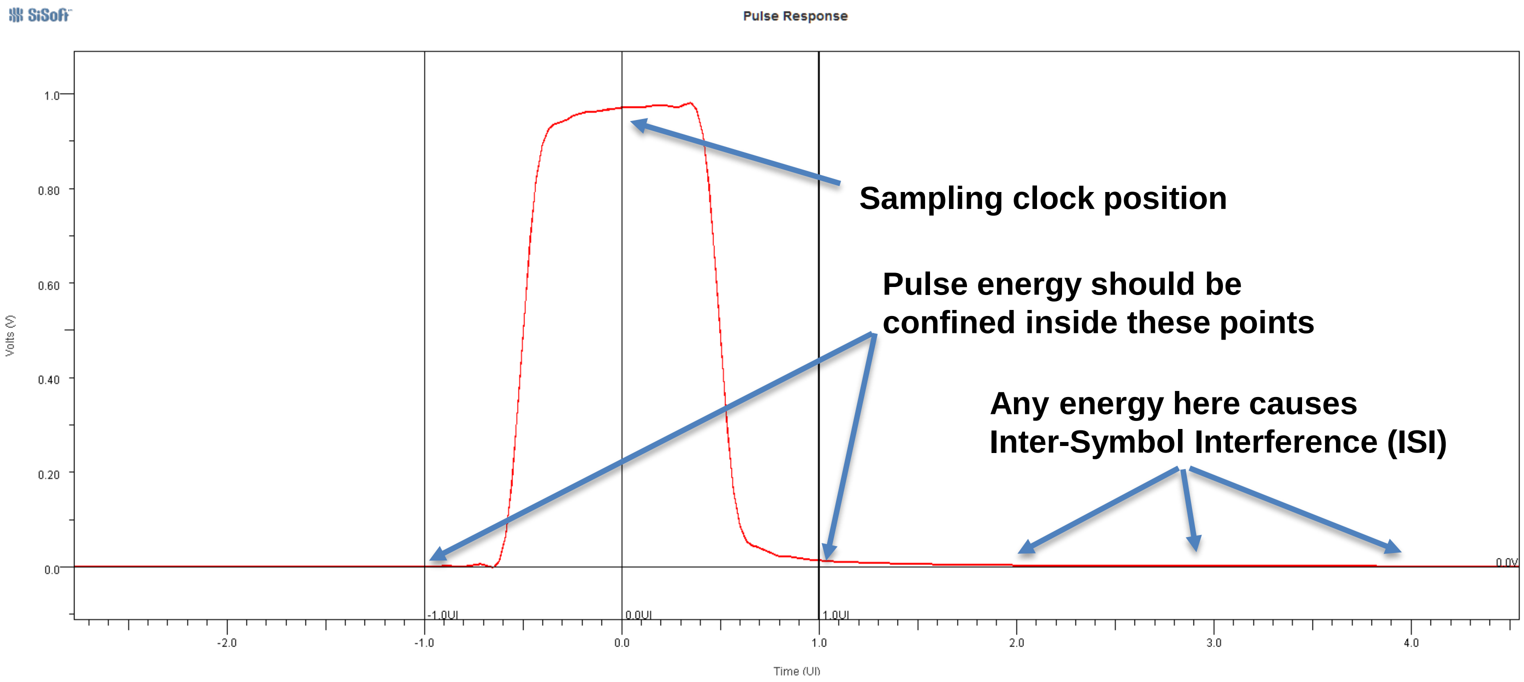
Pulse response


Each possible amplitude is the convolution of the data stream d_n with the cursors r_n
$A = \sum_n d_n r_n$
$d = \{-1,1\}$
Amplitude
Probability
All equal probable combination of cursors



- Computes eye diagram directly from pulse response
- What pulse response characteristics are best for open eyes?

Inter-Symbol Interference (ISI)

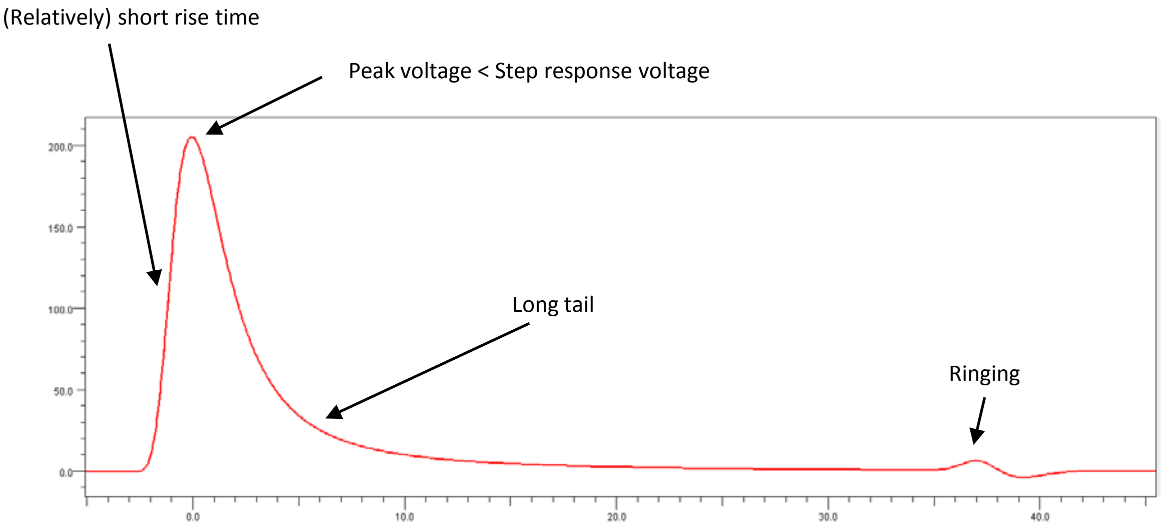


Sampling clock position

Pulse energy should be confined inside these points

Any energy here causes Inter-Symbol Interference (ISI)

Channel Pulse Response


(Relatively) short rise time

Peak voltage < Step response voltage

Long tail

Ringing

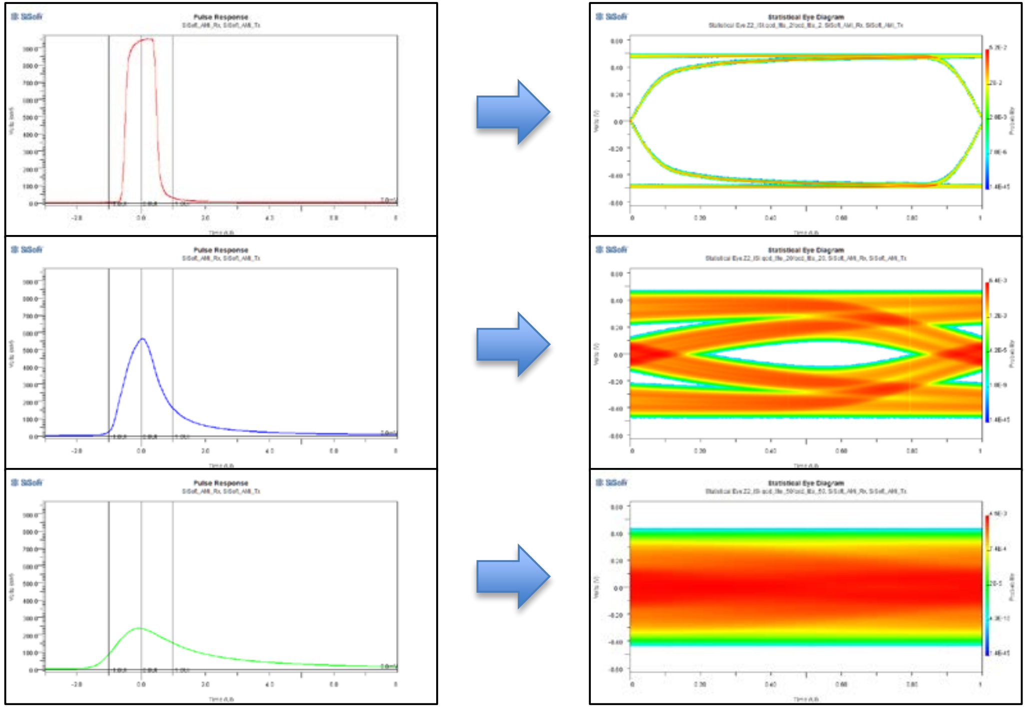
- **Need accurate models to correctly predict loss and reflections**
- **Analog Tx/Rx models are often overlooked**

Channels, Pulses and Statistical Eyes

Short channel,
Minimal ISI

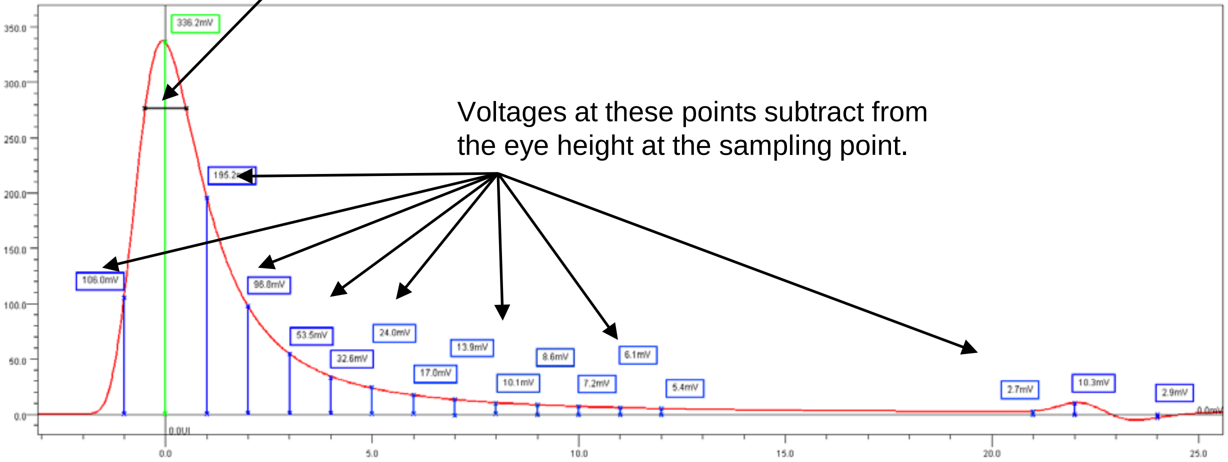
Medium channel,
Moderate ISI

Long channel,
Extreme ISI



Pulse Response, ISI and Eye Height

Hula hoop algorithm determines clock sampling time and main cursor height.
This is the maximum possible inner eye height.



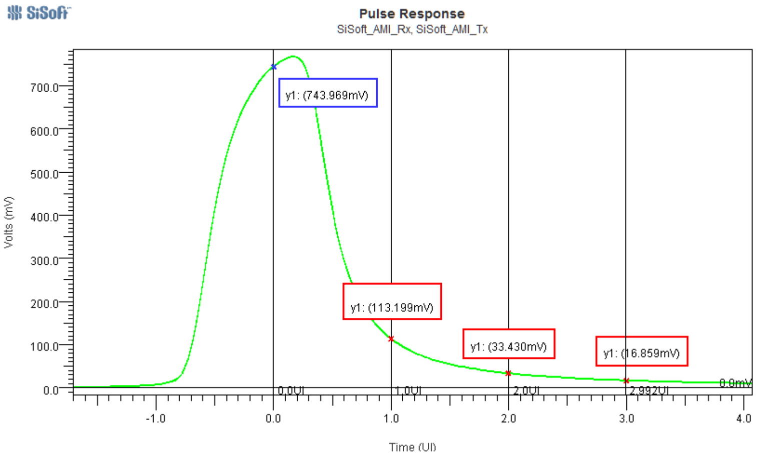
Voltages at these points subtract from
the eye height at the sampling point.

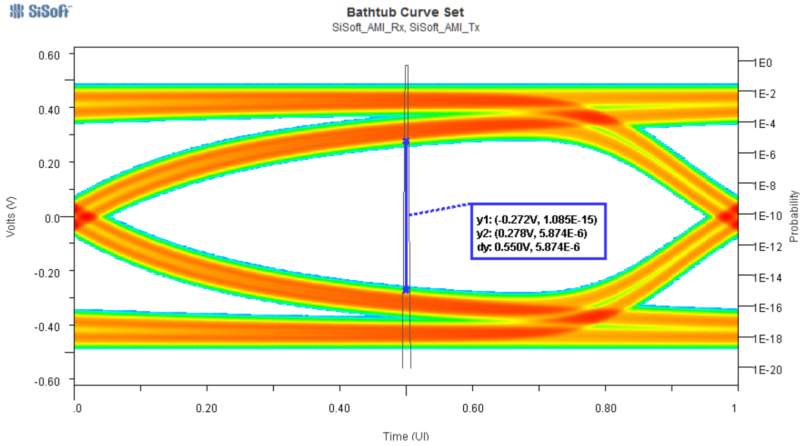
$$\mathit{Inner\ Eye\ Height} =$$

$$\text{main\_cursor} - \Sigma\ |\text{ISI\_voltages}|$$

- Voltage and time scales show ISI contributions
- Useful in evaluating EQ & predicting eye opening

Calculating Inner Eye Height





Prediction: 580mV

Simulated Actual: 550mV

$$\textit{Inner Eye Height} = \text{main\_cursor} - \Sigma\,|\text{ISI\_voltages}|$$

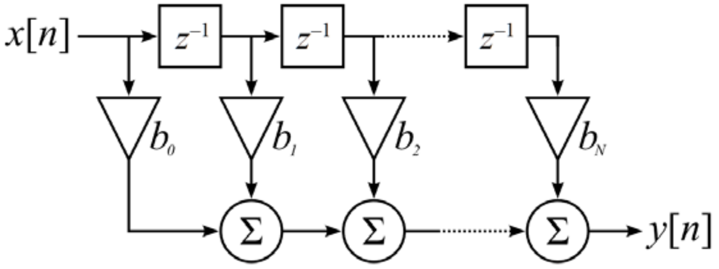
A quick calculation gets us close, but small amounts of energy in the tail add up

The Role of Equalization

- **Some things can be compensated for, some things can't:**
  - Compensate (within limits):
    - Channel loss
    - Reflections due to channel discontinuities
  - Can't compensate:
    - Random noise (that is, truly random noise)
    - Effectively random noise (that is, crosstalk & power noise)

- **The signal really only matters <u>at the sampling point</u>**
  - More on this later

Tx Feed-Forward Equalization (FFE)



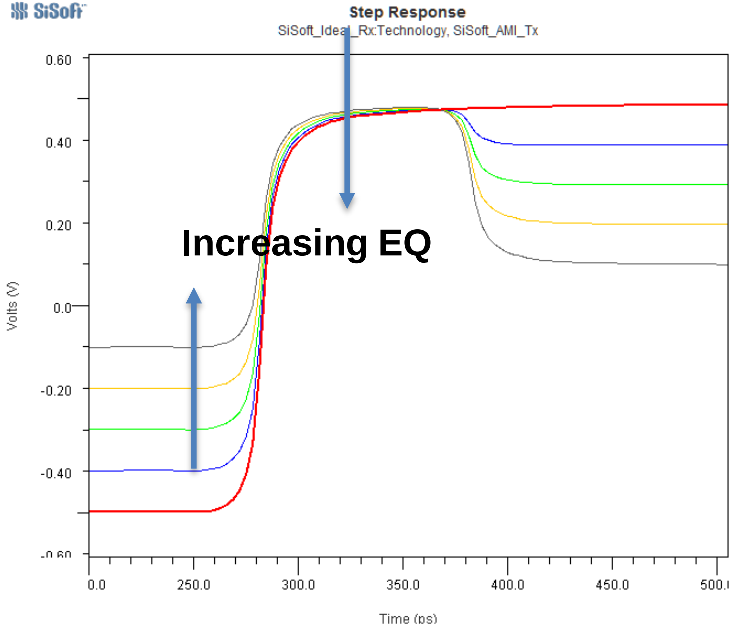
- **Typically implemented as taps spaced 1 UI apart**

- **Can precede the signal (pre-cursor), follow the signal (post-cursor), or both**

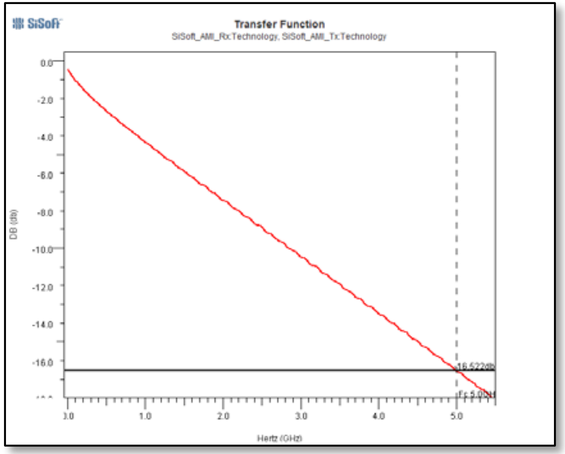
- **Common configuration is 1 pre-cursor, 2 post-cursor taps**

TX FFE Equalization (1st post-cursor)

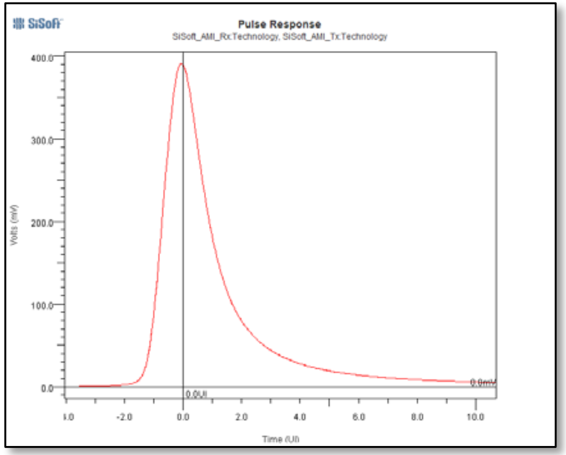
- **Goal: reduce disparity between high and low frequency channel losses**
- **TX EQ is usually implemented as de-emphasis**
  - Transition occurs at full strength, then driver “pulls back” for subsequent bits
  - Reduces the energy sent into the channel



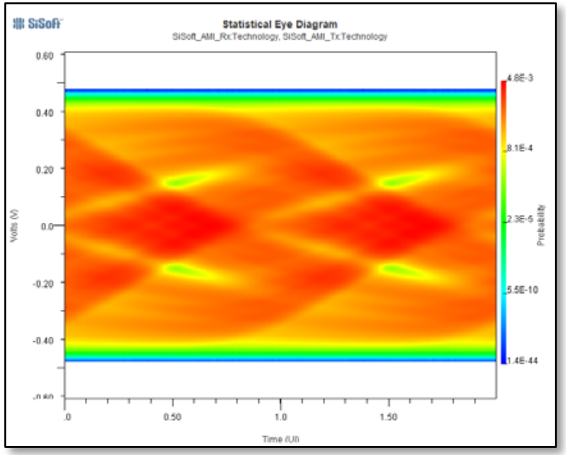
Example: 20 Inch Channel, 10 Gb/s



16.5 dB loss



12+ bits of ISI

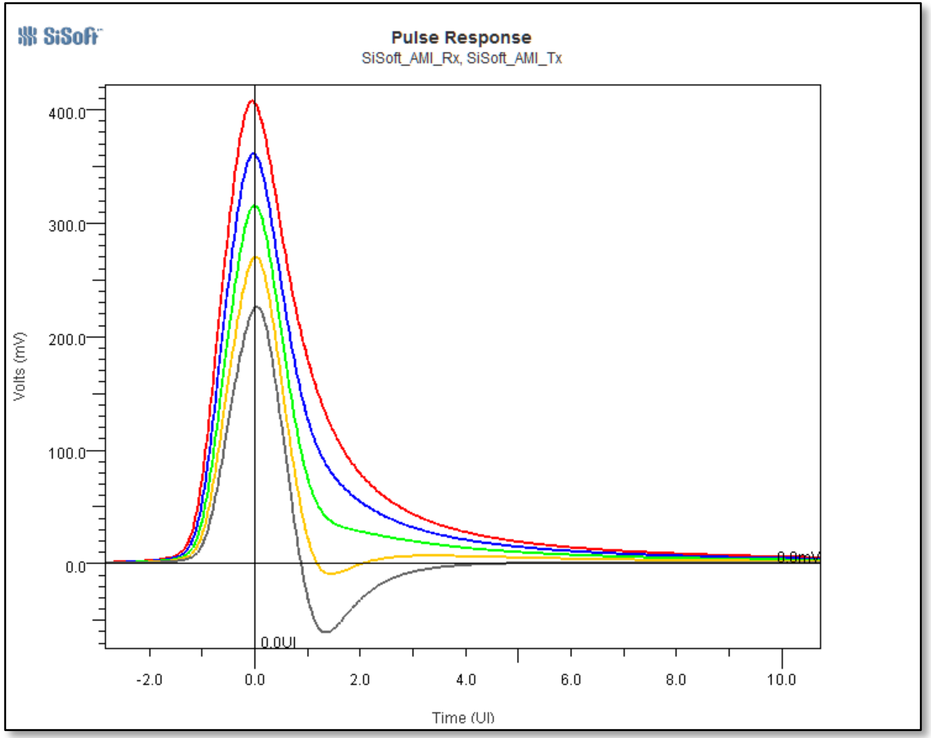


No EQ = No eye

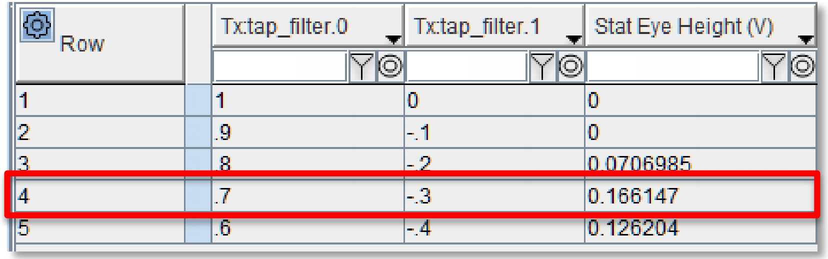
Optimizing TX Equalization

Case	Cursor	1st Post
1	1.0	0.0
2	0.9	-0.1
3	0.8	-0.2
4	0.7	-0.3
5	0.6	-0.4

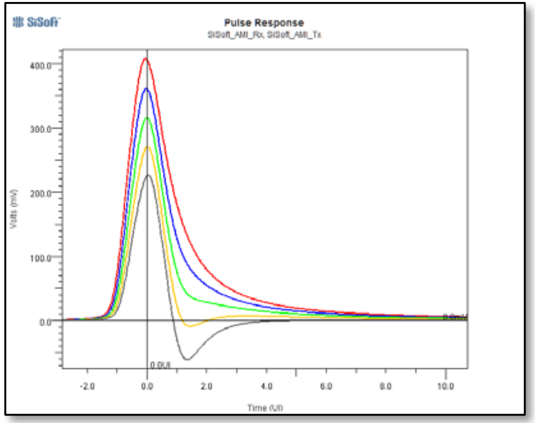
- **Which case will provide the best eye?**

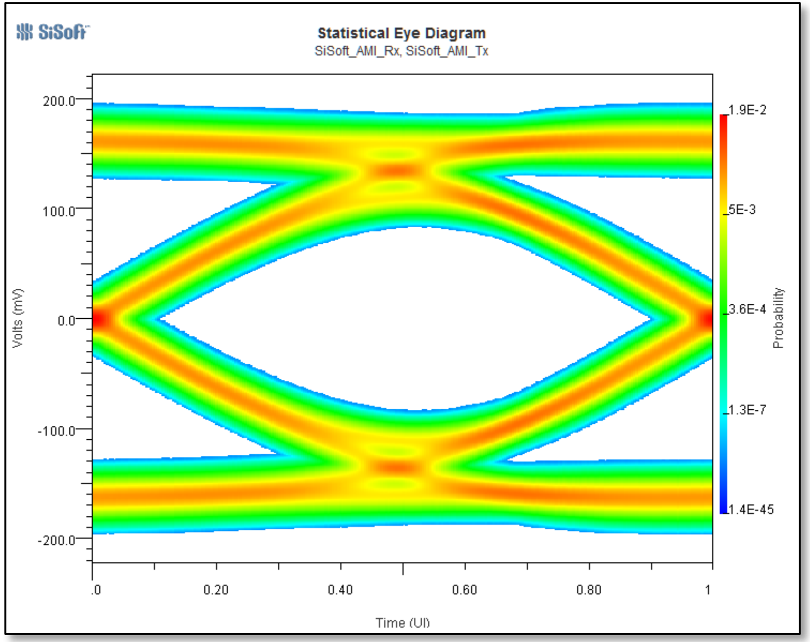


Optimizing TX Equalization

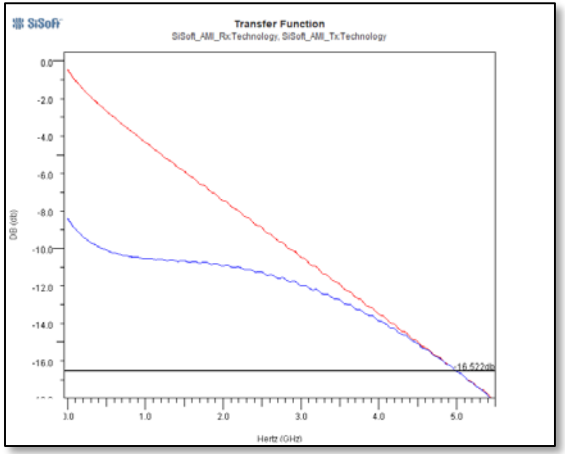


Row	Tx:tap_filter.0	Tx:tap_filter.1	Stat Eye Height (V)
1	1	0	0
2	.9	-.1	0
3	.8	-.2	0.0706985
4	.7	-.3	0.166147
5	.6	-.4	0.126204

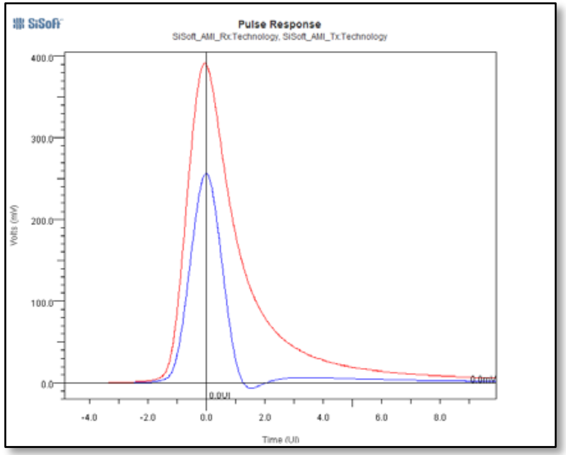




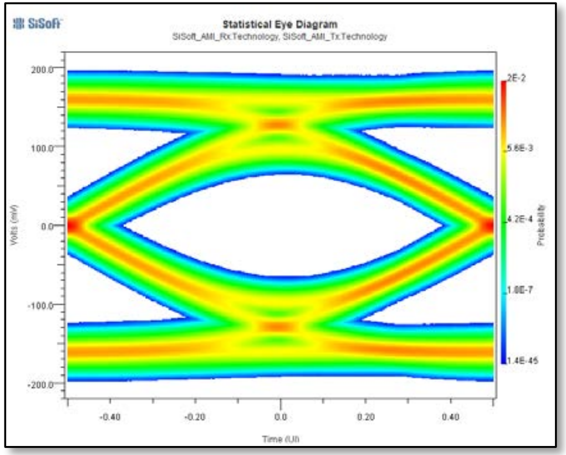
Effect of Tx Equalization



Flattened loss curve



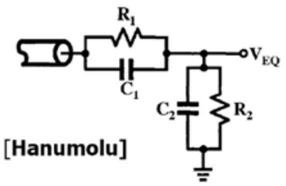
Reduced ISI



Open eye

Rx Continuous Time Linear Filter (CTLE)

- Also called a “Peaking Filter”
- Typically analog circuitry designed to flatten system insertion loss curve
- Typically found in the “front end” of SerDes receivers
- Can be passive or active



[Hanumolu]

$$H(s)=\frac{R_2}{R_1+R_2}\frac{1+R_1C_1s}{1+\frac{R_1R_2}{R_1+R_2}(C_1+C_2)s}$$

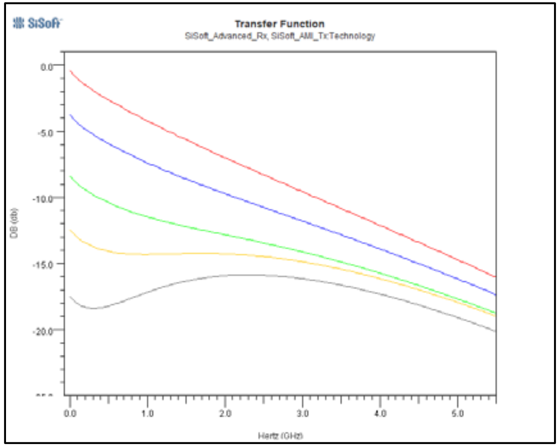
$$\omega_z=\frac{1}{R_1C_1},\qquad \omega_p=\frac{1}{\frac{R_1R_2}{R_1+R_2}(C_1+C_2)}$$

$$\textbf{DC gain}=\frac{R_2}{R_1+R_2},\ \textbf{HF gain}=\frac{C_1}{C_1+C_2}$$

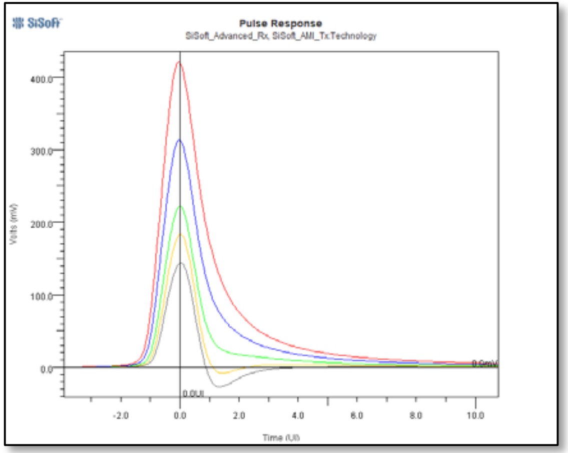
$$\textbf{Peaking}=\frac{\textbf{HF gain}}{\textbf{DC gain}}=\frac{\omega_p}{\omega_z}=\frac{R_1+R_2}{R_2}\frac{C_1}{C_1+C_2}$$

From Texas A&M, ECEN72, Lecture 8, Sam Palermo
http://www.ece.tamu.edu/~spalermo/ecen689/lecture8_ee720_rxeq.pdf

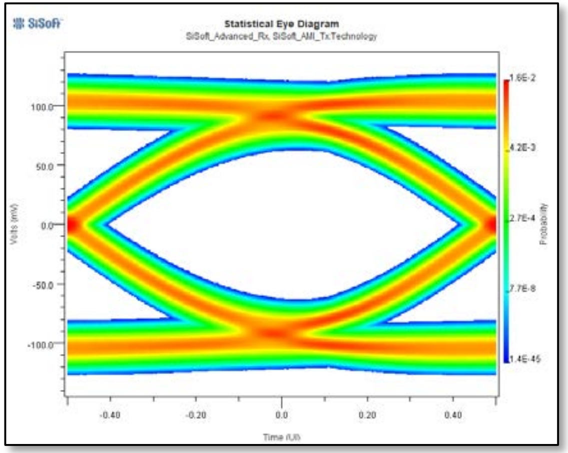
Rx CTLE (Same 20” Channel)



Insertion loss

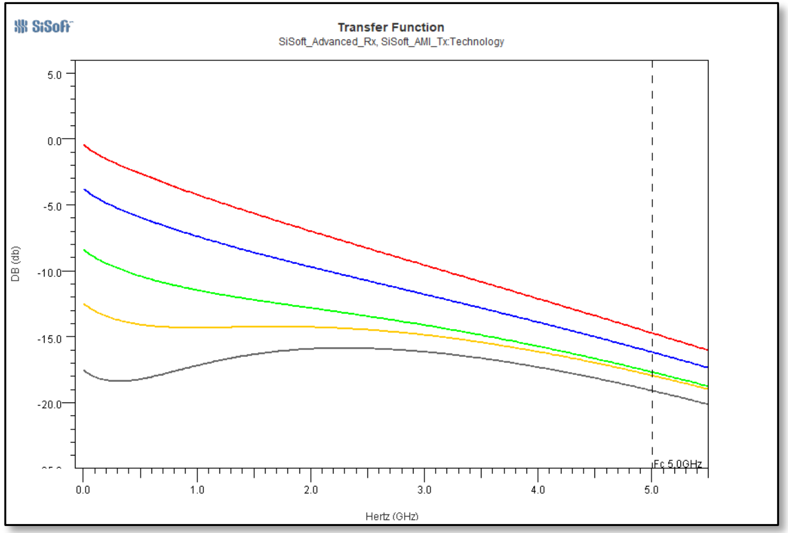


Pulse responses

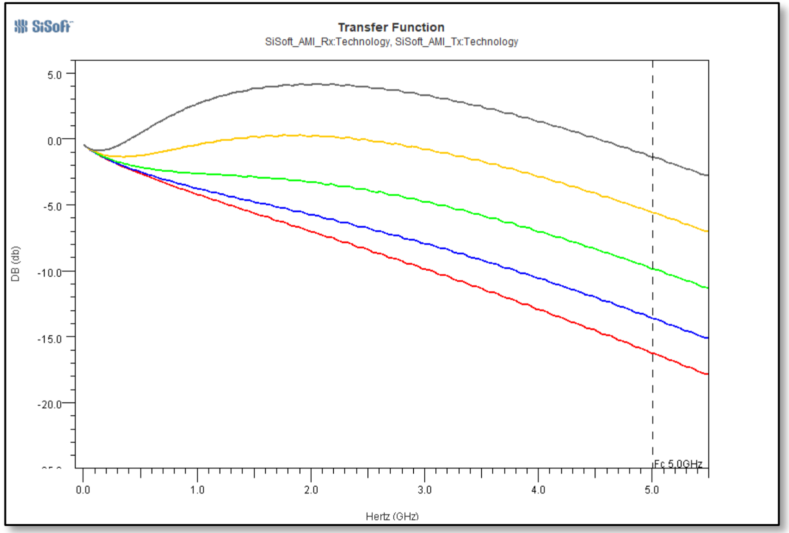


Best case eye

Rx CTLEs and Gain @ Nyquist



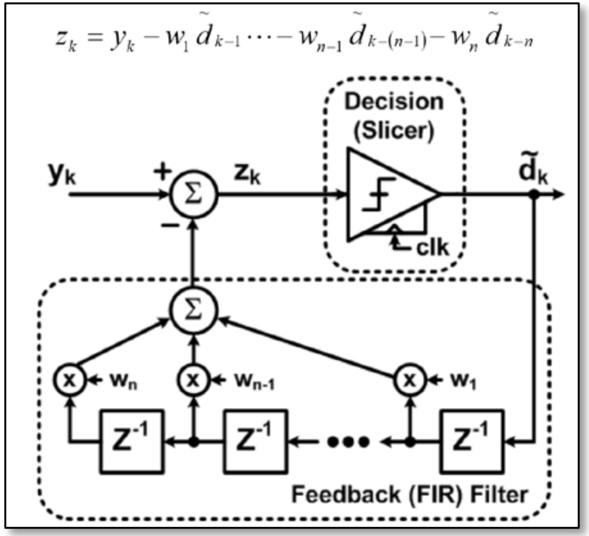
Passive



Active

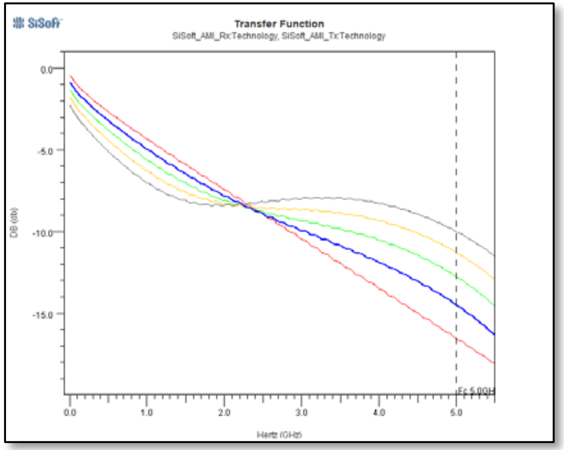
Rx Decision Feedback Equalizer (DFE)

- **Active, power-hungry non-linear equalization**
- **Slicer makes symbol decisions and uses them to cancel out ISI from previously detected bits**
- **Adjustments are intended to cancel out ISI at the instant the signal is sampled**
- **Fixed length tap array, taps only affect a single bit**

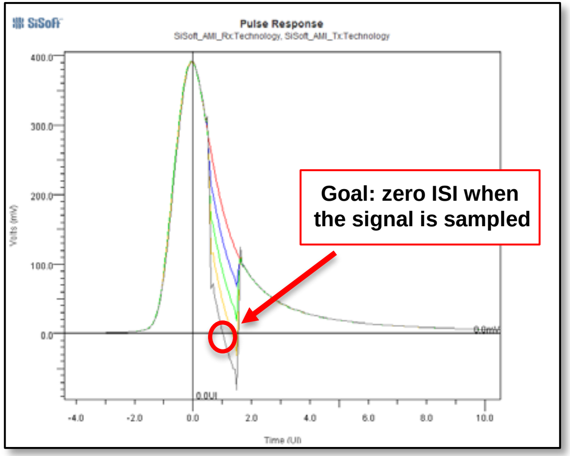


From Texas A&M, ECEN72, Lecture 8, Sam Palermo
http://www.ece.tamu.edu/~spalermo/ecen689/lecture8_ee720_rxeq.pdf

Rx DFE (Single Tap Example)

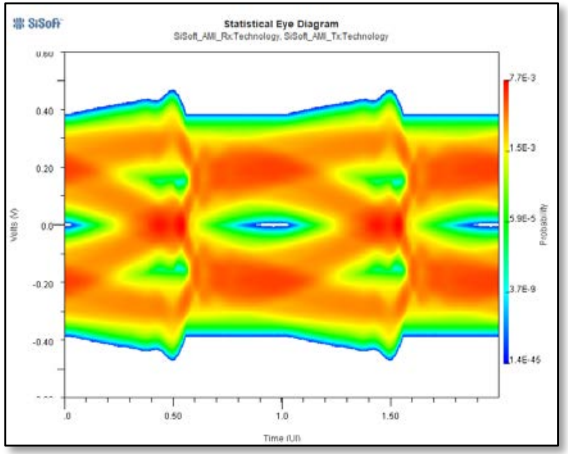


Insertion loss



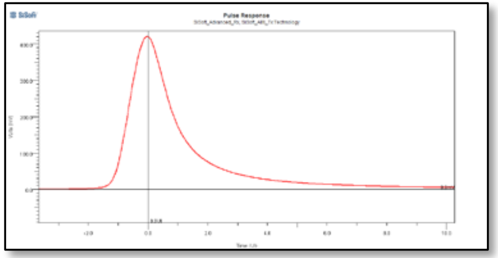
Goal: zero ISI when the signal is sampled

Pulse responses

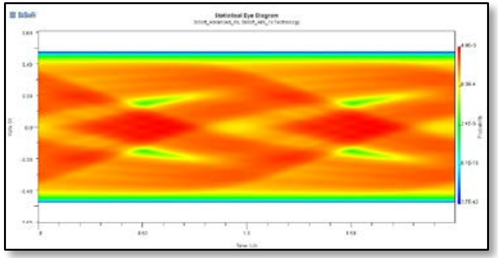


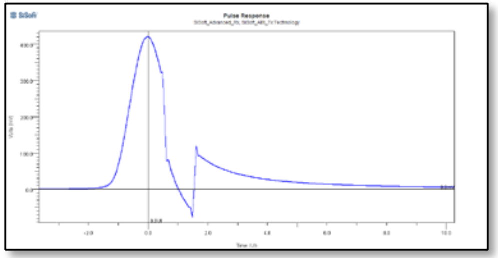
Best case eye

Rx DFE and Number of Taps

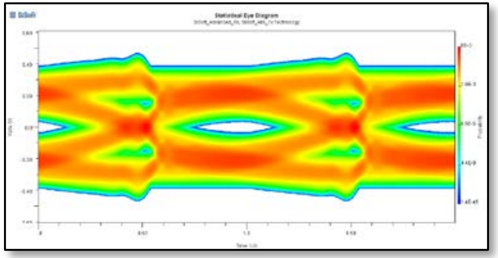


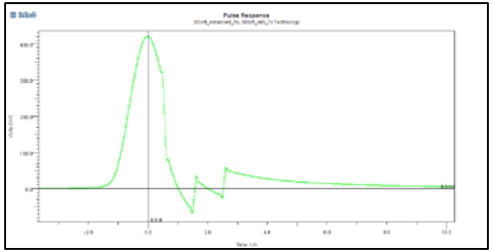
No taps



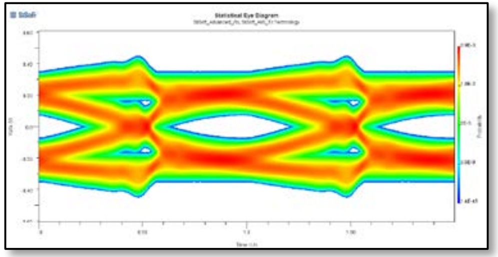


1 tap

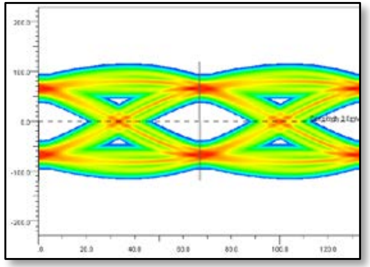




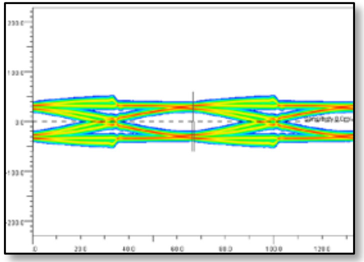
2 taps



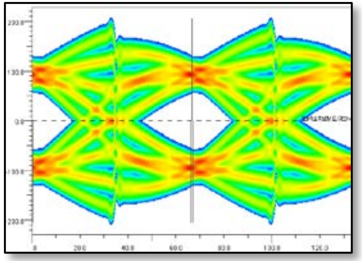
Evaluating EQ Tradeoffs



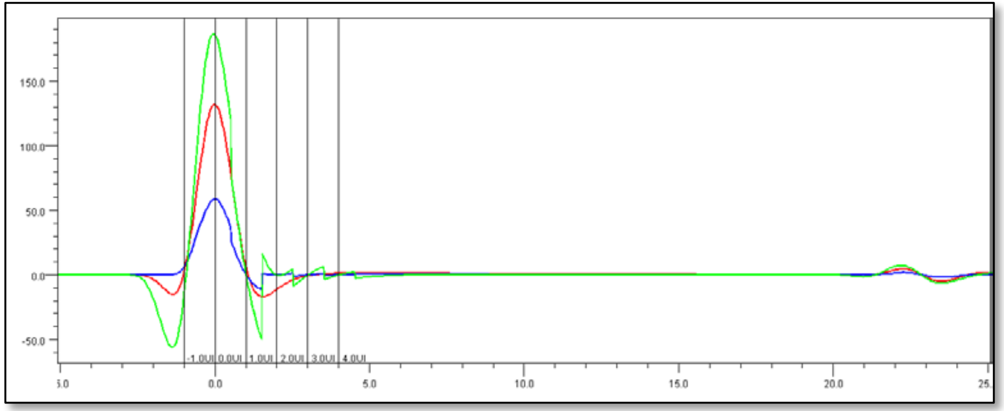
TX Only



RX Only

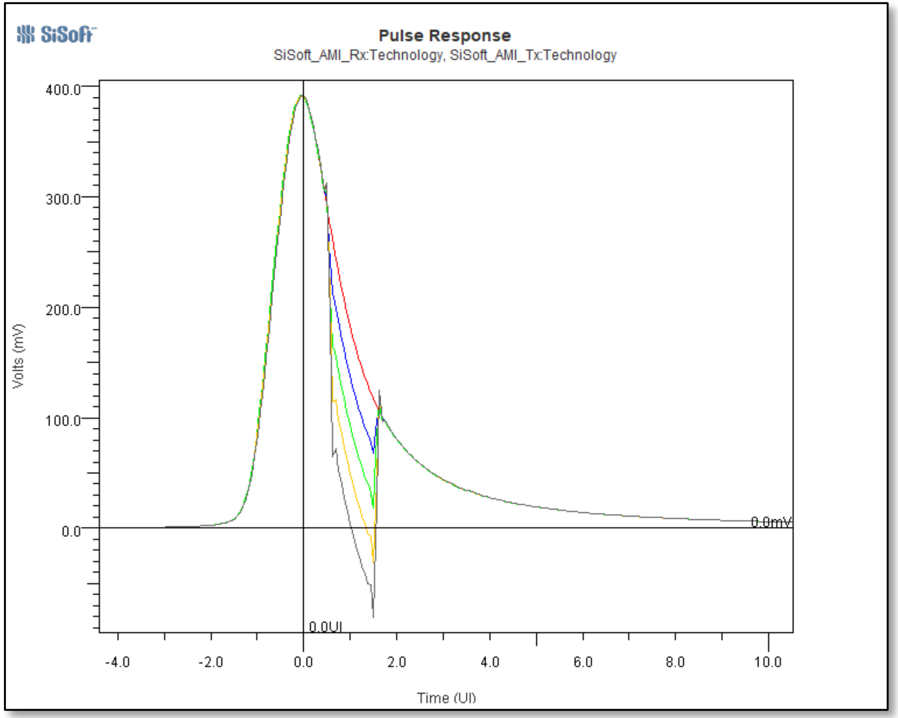


Tx & Rx



- Tx EQ trades cursor amplitude for reduced ISI
- Tx and Rx CTLE both address pre- & post-cursor ISI
- Tx and Rx CTLE best suited for channel loss (not ringing)
- DFE does not reduce cursor height but only corrects single bits
- DFE can correct for loss, if enough taps are present
- DFE best suited for correcting for ringing *if* taps can cover the corresponding bit time

Of AMI Models and Pulse Responses …



- **Channel pulse responses can be obtained from**
    - Statistical simulation
    - Time-Domain simulation
- **Isolating pulse responses in Time-Domain can affect the channel's operating point**
    - Statistical simulations are preferred
- **AMI models require "Init" to support statistical simulations**

Summary: Optimizing Equalization

- Tx/Rx EQ compensates for pattern-based channel ISI
- Primary causes of ISI are high frequency loss and reflections
- Pulse responses show what equalization might be effective
- Tx/Rx equalization methods have specific signatures and uses
- Maximizing margin involves balancing equalization methods
- AMI models need “Init” support for pulse response analysis

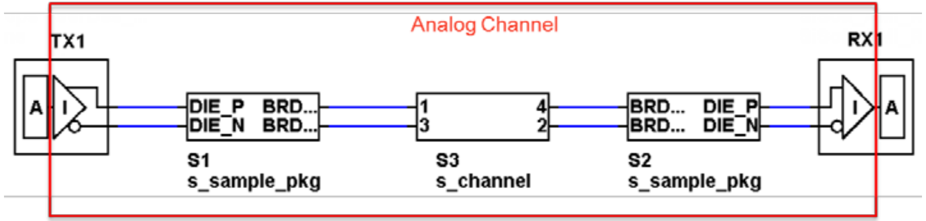
Statistical Simulations with IBIS-AMI Models

Network Characterization

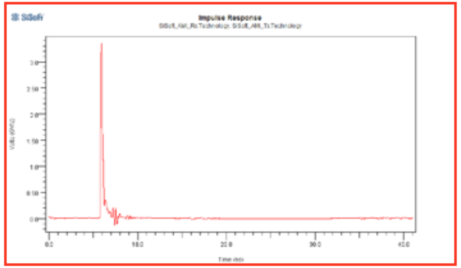
- **Inputs:**
  - Analog sections of .ibs file
  - Passive topology elements

- **Analysis Method:**
  - Not specified by IBIS
  - Time-domain (step response)
  - Frequency-domain (transfer function)

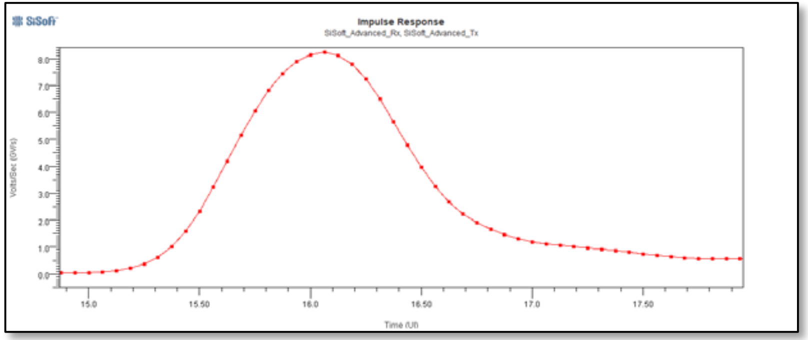
- **Outputs:**
  - Impulse response
  - Fixed time steps
  - Long enough for signal to settle



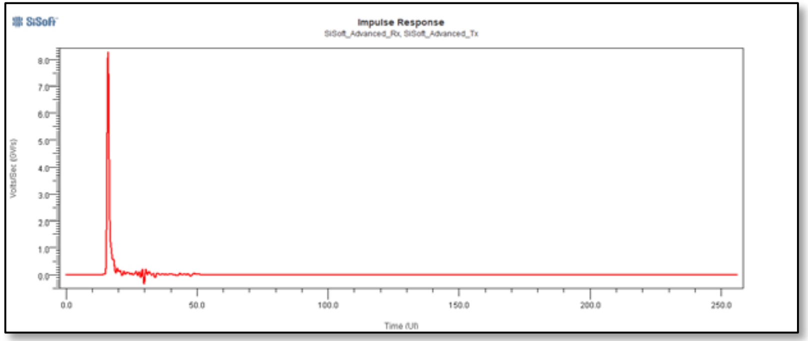
Time-Domain or Frequency-Domain



Analog Channel Impulse Response



Fixed time steps



Long enough for reflections to settle

- **Impulse response should include accurate Tx/Rx impedance models**
  - If not, reflections / ringing will be wrong
- **Impulse response has fixed time steps**
  - Ratio of sample time step (sample_interval) to UI is oversampling or “samples per bit” ratio
- **AMI channel simulations use this same samples per bit setting**
- **Impulse response should be long enough for all reflections to settle out**

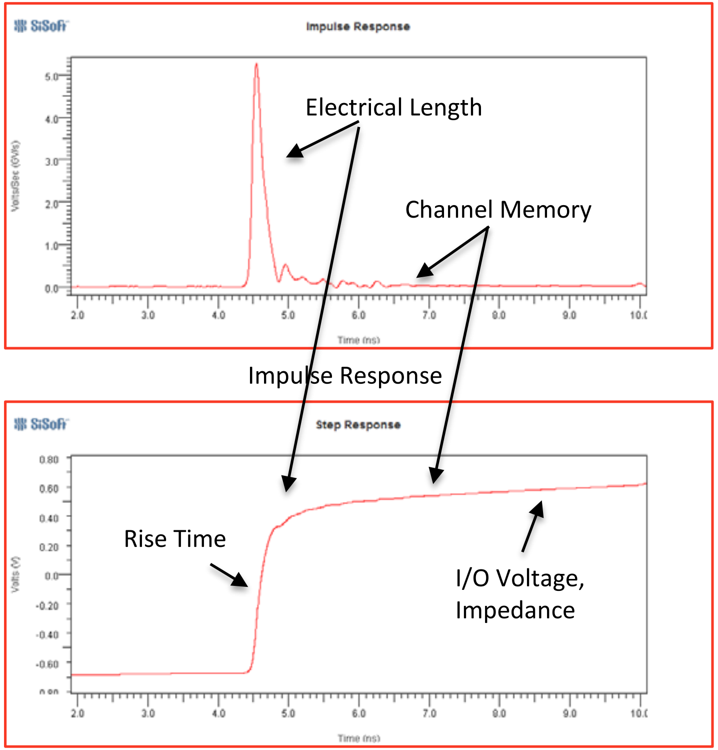
About the Channel Impulse Response …

- **<u>Only</u> the impulse response goes forward from Network Characterization …**
  - If the impulse response is bad, running Channel Simulation ***<u>is a waste of time</u>***

- **<u>Verify</u> impulse response before running channel analysis**

- **Step response is easier to interpret**
  - Voltage levels
  - Rise time
  - Network delay
  - Reflections and settling behavior

- **Remember – channel impulse response does <u>not</u> include TX or RX equalization**



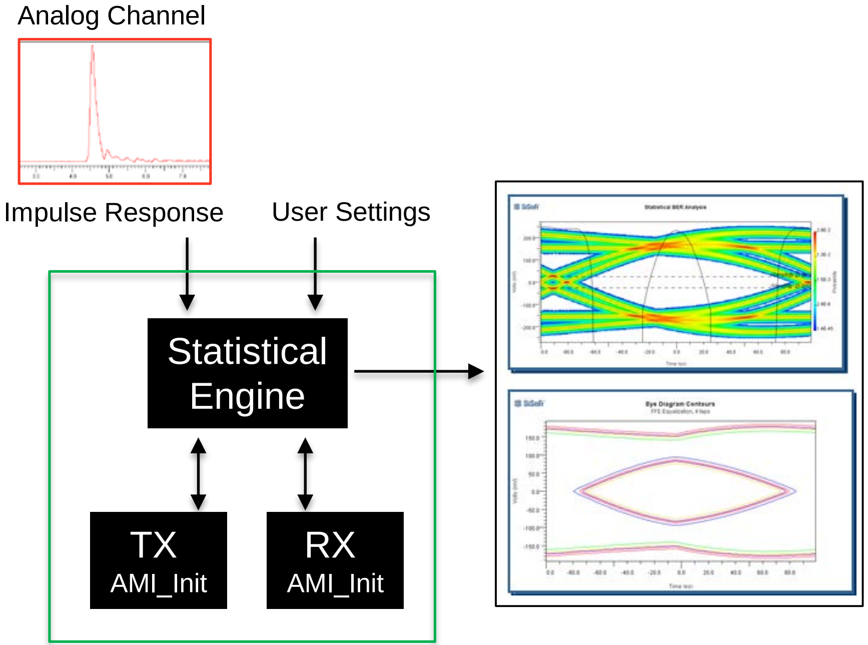
Impulse Response

Step Response

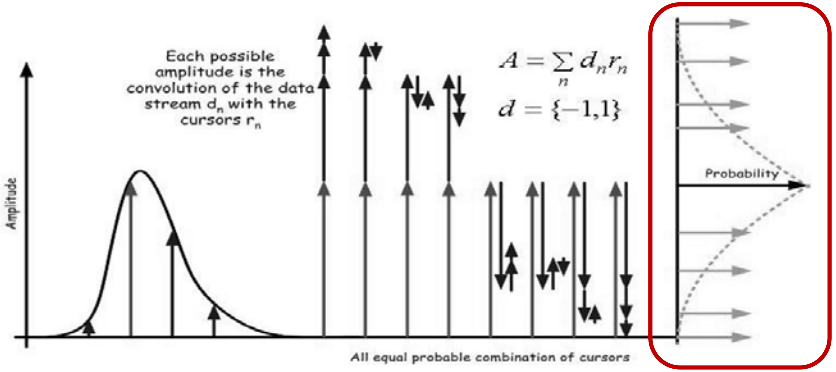
Statistical Simulation

- **Inputs:**
  - Analog channel impulse response
  - User selections for AMI model parameters
  - Algorithmic models
    (AMI_Init / impulse response processing)

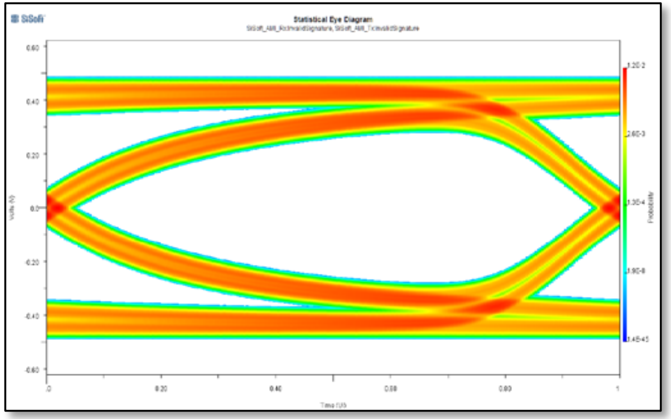
- **Outputs:**
  - Not specified by IBIS
  - Statistical eye diagrams
  - Eye height / width measurements
  - Eye contours @ probabilities
  - Equalized / unequalized responses



All Possible LTI Combinations Evaluated

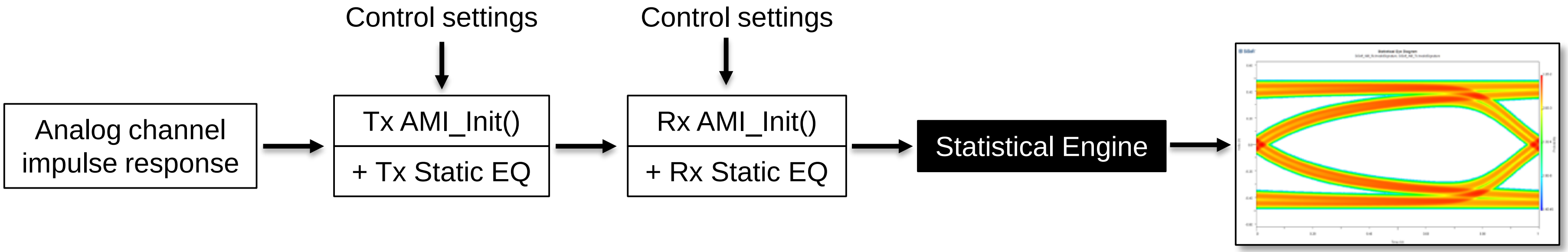


Amplitude
Each possible amplitude is the convolution of the data stream d_n with the cursors r_n
$A = \sum_n d_n r_n$
$d = \{-1,1\}$
Probability
All equal probable combination of cursors



- Eye diagram represents (nearly) infinitely long, random pattern
- Algorithm runs fast, typically a few seconds
- Statistically rich, represents probabilities < 1e-50

Statistical Simulation Flow



Control settings

Control settings

Analog channel impulse response

Tx AMI_Init()

+ Tx Static EQ

Rx AMI_Init()

+ Rx Static EQ

Statistical Engine

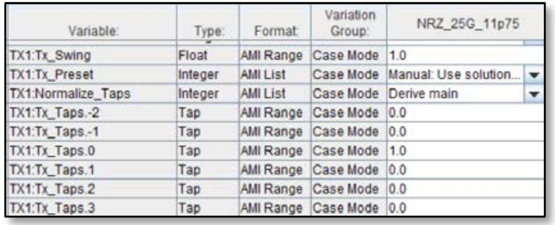
- **Statistical flow is constant, not dependent on AMI model type**
- **If a model does not support “Init”, its behavior is absent**
  - Tx but no Rx Init ➔ eye represents eye at Rx pad
  - Rx but no Tx Init ➔ no physical correspondence
  - No Tx or Rx Init ➔ eye represents channel only
- **Eye centering is performed by simulator (no clock from model)**

AMI Parameter Passing

```
(Model_Specific
   (Tx_Swing (Usage In)(Type Float)(Range 1.0 0.3 2.0)
     (Description "Peak differential output voltage.")
   )
   (Tx_Preset (Usage In)(Type Integer)(List 11 1 2 3 4 5 6 7 8 9 10)
     (Default 11)(Description "Presets 1-10, use 11 for manual mode.")
   )
   (Normalize_Taps (Usage In)(Type Integer)(List 1 2 3)(Default 3)
     (Description "1:Disable, 2:Scale all, 3:Derive main.")
   )
   (Tx_Taps
     (-2 (Usage InOut)(Type Tap)(Range 0.0 -0.5 0.5)(Description "2nd Pre Tap"))
     (-1 (Usage InOut)(Type Tap)(Range 0.0 -0.5 0.5)(Description "1st Pre Tap"))
     (0  (Usage InOut)(Type Tap)(Range 1.0  0.1 1.0)(Description "Main Tap"))
     (1  (Usage InOut)(Type Tap)(Range 0.0 -0.5 0.5)(Description "1st Post Tap"))
     (2  (Usage InOut)(Type Tap)(Range 0.0 -0.5 0.5)(Description "2nd Post Tap"))
     (3  (Usage InOut)(Type Tap)(Range 0.0 -0.5 0.5)(Description "3rd Post Tap"))
   )
 )
```

1. Model’s .AMI file

2. User selections



Variable:	Type:	Format:	Variation Group:	NRZ_25G_11p75
TX1:Tx_Swing	Float	AMI Range	Case Mode	1.0
TX1:Tx_Preset	Integer	AMI List	Case Mode	Manual: Use solution... ▼
TX1:Normalize_Taps	Integer	AMI List	Case Mode	Derive main ▼
TX1:Tx_Taps.-2	Tap	AMI Range	Case Mode	0.0
TX1:Tx_Taps.-1	Tap	AMI Range	Case Mode	0.0
TX1:Tx_Taps.0	Tap	AMI Range	Case Mode	1.0
TX1:Tx_Taps.1	Tap	AMI Range	Case Mode	0.0
TX1:Tx_Taps.2	Tap	AMI Range	Case Mode	0.0
TX1:Tx_Taps.3	Tap	AMI Range	Case Mode	0.0

3. Control string passed to AMI_Init()

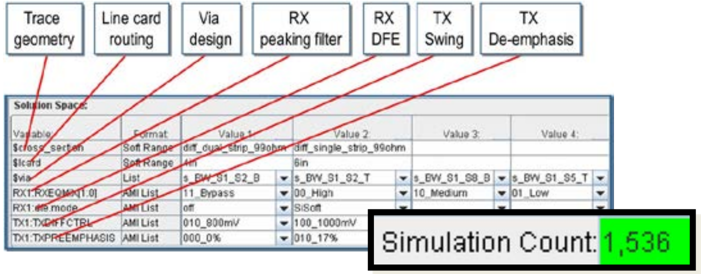
```
(IBIS_AMI_Tx(Tx_Swing 1.0)(Tx_Preset 11)(Normalize_Taps 3)
(Tx_Taps(-2 0.0)(-1 0.0)(0 1.0)(1 0.0)(2 0.0)(3 0.0)))
```

AMI_Init() and Equalization

- **Modeling Linear, Time-Invariant (LTI) equalization is straightforward**
  - Tx FIR (FFE) filters and Rx CTLE
  - Supported, proven, portable among EDA tools
- **Modeling Nonlinear, Time-Varying (DFE) equalization is possible**
  - Proven and portable among EDA tools even though not consistent with definitions of AMI_Init() modeling
- **Self-optimizing models are possible**
  - For example, Rx models can optimize CTLE or DFE tap settings
  - Adaptation cannot be modeled literally, but the endpoint can
- **Complex modeling is controversial**
  - For example, saturation can be modeled in a limited manner, but portability among EDA tools is questionable

Neat Statistical Simulation Tricks (YMMV)

Quick design space search

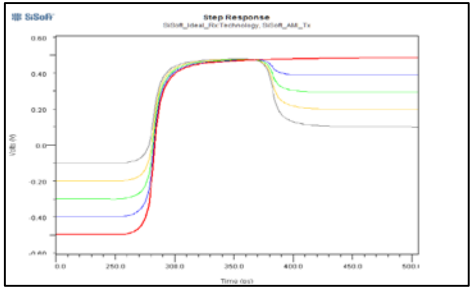


Solution Space:

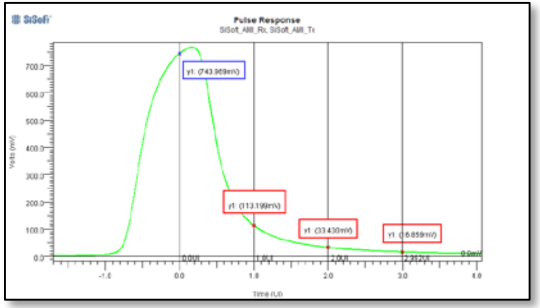
Variable:	Format:	Value 1:	Value 2:	Value 3:	Value 4:
$cross_section	Soft Range	diff_dual_strip_99ohm	diff_single_strip_99ohm		
$lcard	Soft Range	4in	6in		
$via	List	s_BW_S1_S2_B	s_BW_S1_S2_T	s_BW_S1_S8_B	s_BW_S1_S5_T
RX1:RXEQMAN[1:0]	AMI List	11_Bypass	00_High	10_Medium	01_Low
RX1:dfe.mode	AMI List	off	SiSoft		
TX1:TXDIFFCTRL	AMI List	010_800mV	100_1000mV		
TX1:TXPREEMPHASIS	AMI List	000_0%	010_17%		

Simulation Count: 1,536

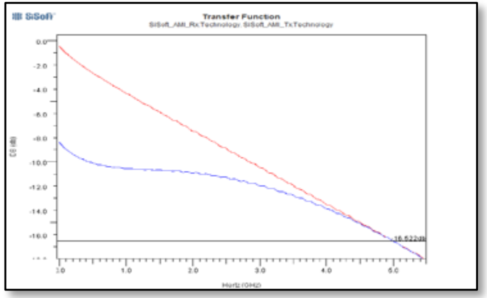
Characterize EQ using step response



Estimate eye height from EQ pulse response



EQ effect on channel transfer function



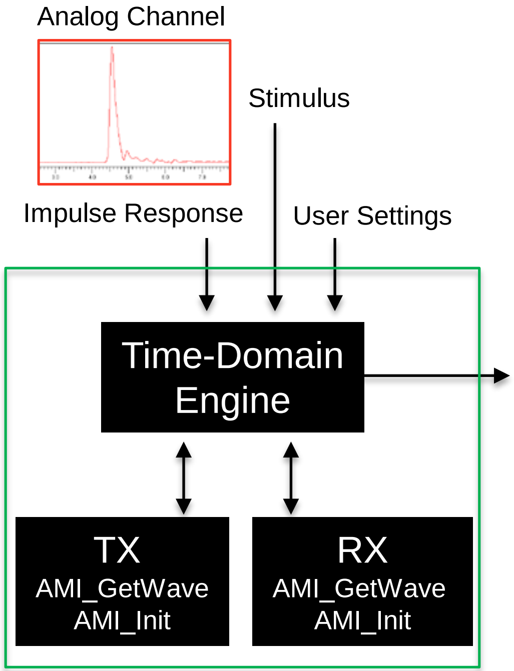
Summary: Statistical Simulations with AMI

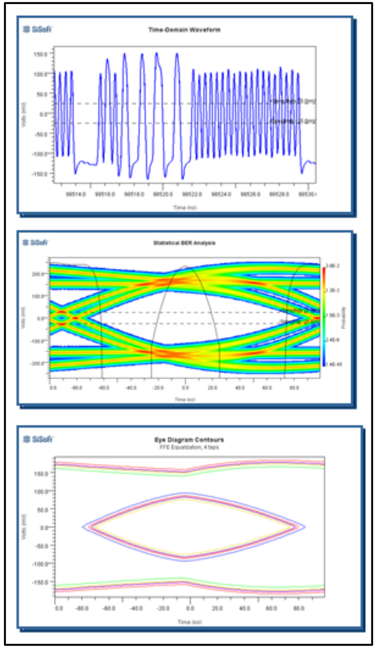
- **Generates eye directly from a pulse response**
- **Statistically rich; random pattern, probabilities < 1e-50**
- **Fast analysis; typically 1 - 4 seconds**
- **Static equalization; can optimize coefficients but cannot model adaptation sequence**
- **AMI models:**
  - Require Init_Returns_Impulse = True in .AMI file (impulse response processing)
  - Eye diagram can be “missing” effects of Tx or Rx EQ (or both)
  - Models use control settings passed in at runtime
  - Sampling clock prediction is performed by the simulator

Time Domain Simulations with IBIS-AMI Models

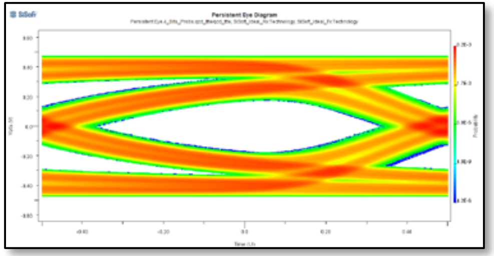
Time-Domain Simulation

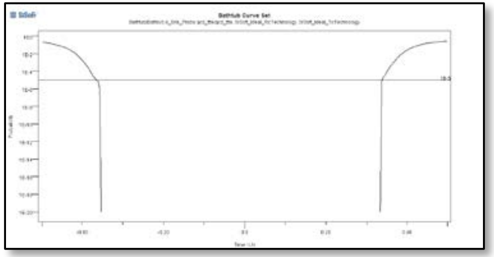
- **Inputs:**
  - Impulse responses from prior steps
  - User-defined input stimulus
  - Algorithmic models (AMI_GetWave / waveform processing)
- **Analysis Method:**
  - Waveform processing & convolution
- **Outputs:**
  - Not specified by IBIS
  - Time domain waveforms and clock times
  - Persistent eye diagrams
  - Eye height / width measurements
  - Eye contours @ probabilities
  - Equalized / unequalized responses

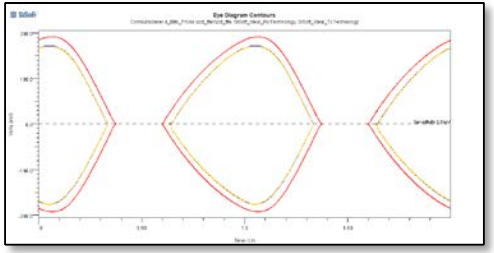




# Bits Simulated and Probabilities



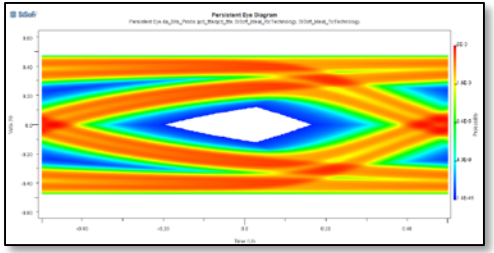


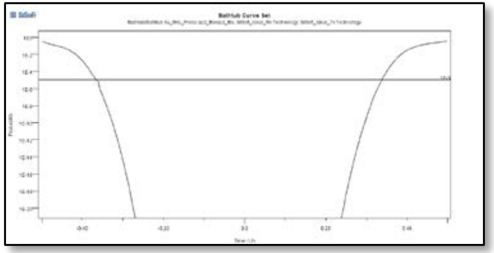


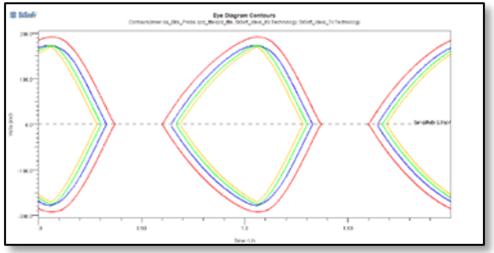
- Time-Domain simulations are typically 1e5~1e7 bits long
- Results and probabilities are limited by the number of bits simulated
- “Ignore_Bits” setting throws bits away at the start of simulation while equalization stabilizes, subtracts from bits available to compute probabilities

200,000 bits simulated, 10,000 ignored.
190,000 bits available for post-processing

Extrapolation







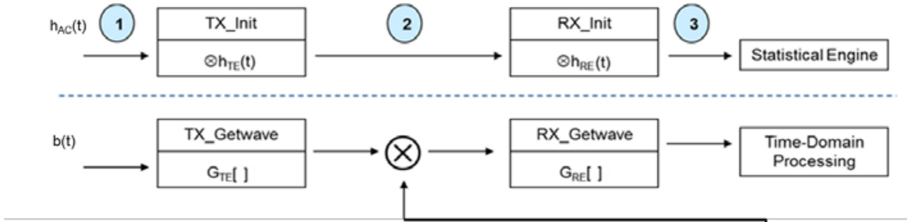
- EDA simulators can extrapolate results to predict margins at low probability levels
- Extrapolation required for
  - Tx jitter
  - ISI
  - Crosstalk
- Extrapolation methods and results are EDA tool-specific

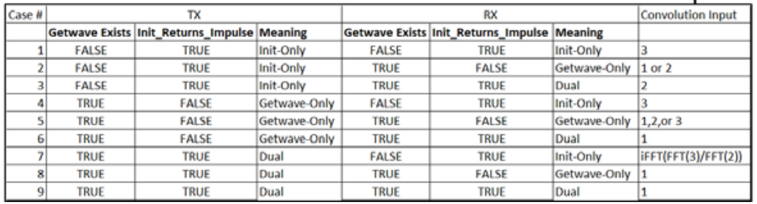
200,000 bits simulated, 10,000 ignored.
190,000 bits available for post-processing

Time-Domain and Equalization

- Time-Domain simulations always include the effects of both Tx and Rx equalization
- A model’s EQ contribution in Time-Domain simulation can come from impulse response processing (“Init”) or waveform processing (“GetWave”), but not both
- Init processing is “static” and does not vary from bit to bit
- GetWave processing is “dynamic” and can vary from bit to bit, allowing control loops and adaptation to be modeled
- Clock “ticks” can only be returned by GetWave models

Time-Domain Simulation Flow

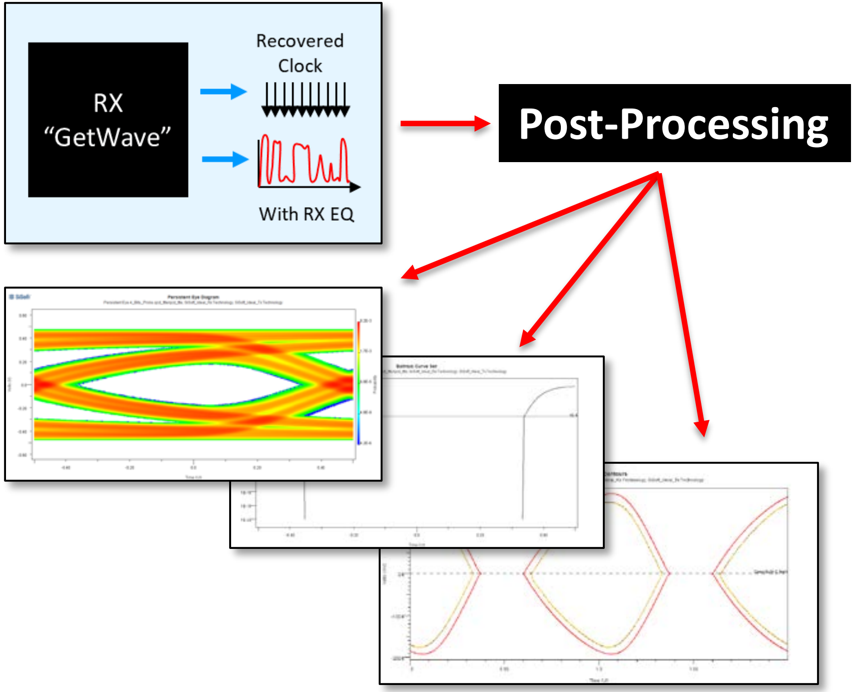




Case #	TX			RX			Convolution Input
	Getwave Exists	**Init_Returns_Impulse**	**Meaning**	**Getwave Exists**	**Init_Returns_Impulse**	**Meaning**	
1	FALSE	TRUE	Init-Only	FALSE	TRUE	Init-Only	3
2	FALSE	TRUE	Init-Only	TRUE	FALSE	Getwave-Only	1 or 2
3	FALSE	TRUE	Init-Only	TRUE	TRUE	Dual	2
4	TRUE	FALSE	Getwave-Only	FALSE	TRUE	Init-Only	3
5	TRUE	FALSE	Getwave-Only	TRUE	FALSE	Getwave-Only	1,2,or 3
6	TRUE	FALSE	Getwave-Only	TRUE	TRUE	Dual	1
7	TRUE	TRUE	Dual	FALSE	TRUE	Init-Only	iFFT(FFT(3)/FFT(2))
8	TRUE	TRUE	Dual	TRUE	FALSE	Getwave-Only	1
9	TRUE	TRUE	Dual	TRUE	TRUE	Dual	1

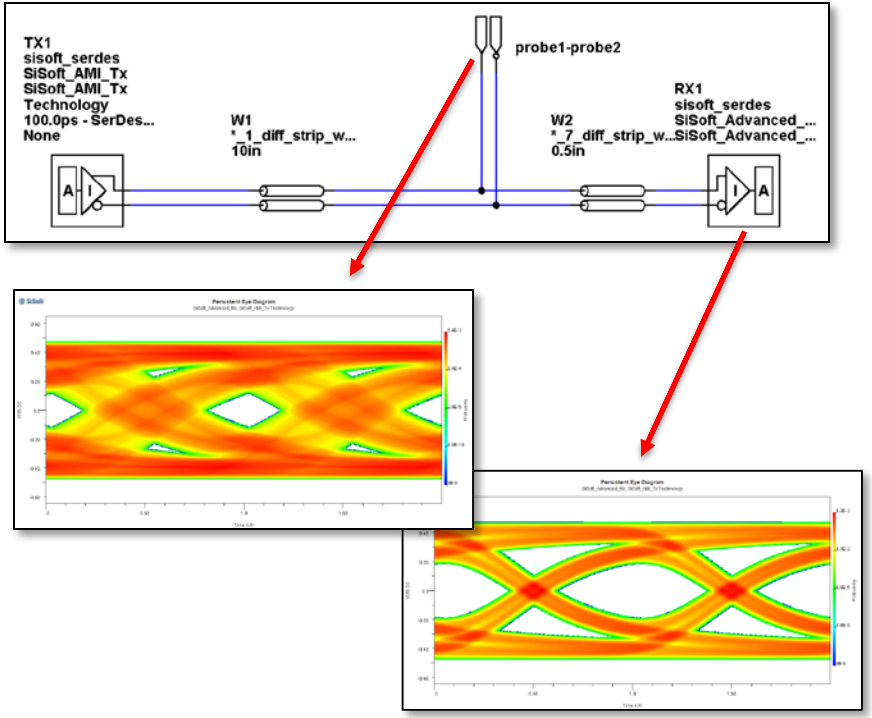
- Time-Domain simulations must accounts for differences in how Init and GetWave models process data
- The simulation flow used depends on the AMI model types involved
- AMI model types are determined by looking at the corresponding .AMI files

AMI Models and Clock “Ticks”



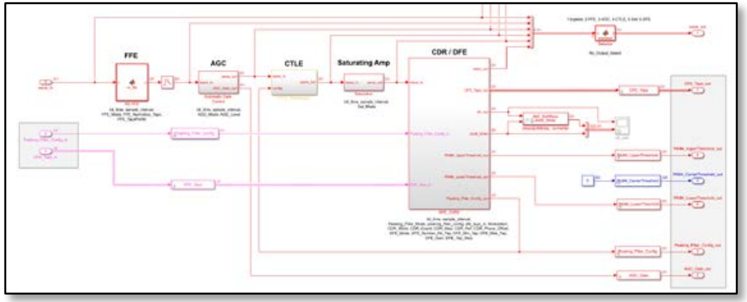
- Rx “GetWave” models return equalized waveforms and clock “ticks” to the simulator
- Clock “ticks” are the output of a CDR modeling loop and represent the start of the UI (not the sampling time)
- “Init” models do not output clocks; clock estimation is performed by the simulator

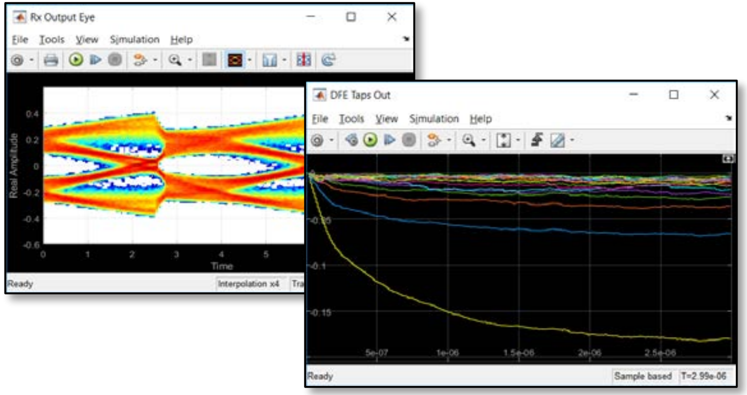
Jitter Tracking



- CDR loops in “GetWave” models can open eyes by tracking out low frequency jitter
- Jitter in AMI models isn’t guaranteed: if it’s there to track out, someone put it there to begin with
- Remember: waveform / clock processing and eye diagram generation is tool-specific

Modeling Adaptive Behavior





- AMI "GetWave" models process waveform data in blocks
- GetWave models can output internal state information as AMI "Output Parameters"
- This can be used to expose key internal state information
  - How DFE taps adapt with time
  - Clock recovery loop behavior
  - Other internal control loop info

Summary: Time-Domain with AMI

- **Simulates channel response to specific input patterns**
  - # bits simulated determines probabilities predicted
  - Simulation performance: ~1M bits / minute
  - Extending to lower probabilities involves tool-specific extrapolation
- **Equalization can be static or dynamic (adaptive)**
- **Can model clock recovery loop and jitter tracking**
- **AMI models**
  - Tx and Rx always have EQ (no “missing” effects)
  - EQ can be either static (“Init”) or dynamic (“GetWave”)
  - Dynamic (“GetWave”) Rx models return waveforms and clock “ticks”
  - Models can output internal state variables as they change
  - Results post-processing and presentation is simulator-specific

IBIS-AMI Simulation Flows

Algorithmic Model Types

3 types of algorithmic models:

1.Impulse response (Init) only

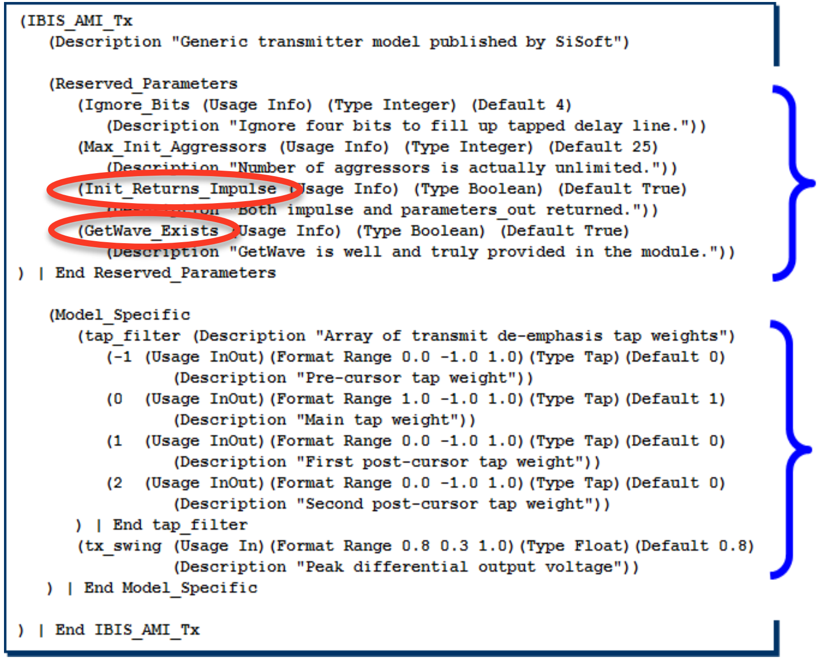
- Init_Returns_Impulse = TRUE
- GetWave_Exists = FALSE

2.Waveform (GetWave) only

- Init_Returns_Impulse = FALSE
- GetWave_Exists = TRUE

3.Dual

- Init_Returns_Impulse = TRUE
- GetWave_Exists = TRUE



```
(IBIS_AMI_Tx
   (Description "Generic transmitter model published by SiSoft")

   (Reserved_Parameters
      (Ignore_Bits (Usage Info) (Type Integer) (Default 4)
         (Description "Ignore four bits to fill up tapped delay line."))
      (Max_Init_Aggressors (Usage Info) (Type Integer) (Default 25)
         (Description "Number of aggressors is actually unlimited."))
      (Init_Returns_Impulse (Usage Info) (Type Boolean) (Default True)
         (Description "Both impulse and parameters_out returned."))
      (GetWave_Exists (Usage Info) (Type Boolean) (Default True)
         (Description "GetWave is well and truly provided in the module."))
) | End Reserved_Parameters

   (Model_Specific
      (tap_filter (Description "Array of transmit de-emphasis tap weights")
         (-1 (Usage InOut)(Format Range 0.0 -1.0 1.0)(Type Tap)(Default 0)
                (Description "Pre-cursor tap weight"))
         (0  (Usage InOut)(Format Range 1.0 -1.0 1.0)(Type Tap)(Default 1)
                (Description "Main tap weight"))
         (1  (Usage InOut)(Format Range 0.0 -1.0 1.0)(Type Tap)(Default 0)
                (Description "First post-cursor tap weight"))
         (2  (Usage InOut)(Format Range 0.0 -1.0 1.0)(Type Tap)(Default 0)
                (Description "Second post-cursor tap weight"))
      ) | End tap_filter
      (tx_swing (Usage In)(Format Range 0.8 0.3 1.0)(Type Float)(Default 0.8)
                (Description "Peak differential output voltage"))
   ) | End Model_Specific

) | End IBIS_AMI_Tx
```

Reserved_Parameters
Provide information on model function and control analysis flow
[Info for the simulator]

Model_Specific
Parameters used by this specific model, legal and default values
[Info for providing inputs to the model]

.AMI file

Static and Dynamic Equalization

- **Static equalization**
  - Impulse response processing (Init)
  - Happens once - does not vary from bit to bit
  - Treated as LTI by simulation engine
  - Can be used to generate Statistical and Time-Domain results

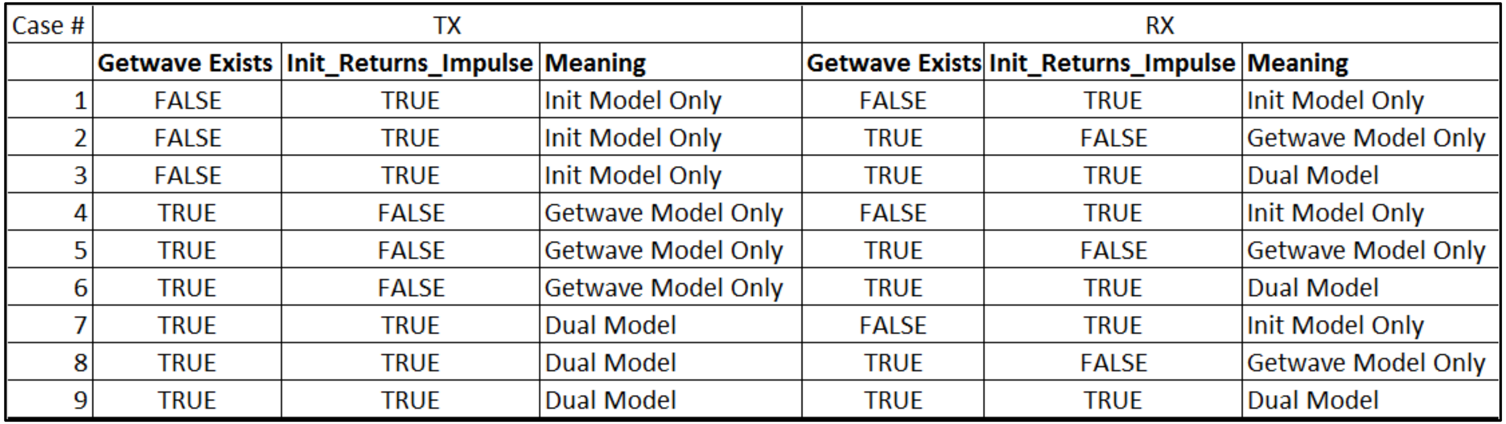
- **Dynamic equalization**
  - Waveform processing (GetWave)
  - Can vary from bit to bit
  - Includes equalization and clock recovery
  - Only used to generate Time-Domain results

Model Type	Equalization
Init-Only	Static
GetWave-Only	Dynamic
Dual	Static & Dynamic

The 9 AMI Simulation Cases

- **The method an AMI simulator uses to create Time-Domain results is based on the types of TX and RX algorithmic models involved.**

{ Init-Only
GetWave-Only
Dual } x { Init-Only
GetWave-Only
Dual } = 9 Cases

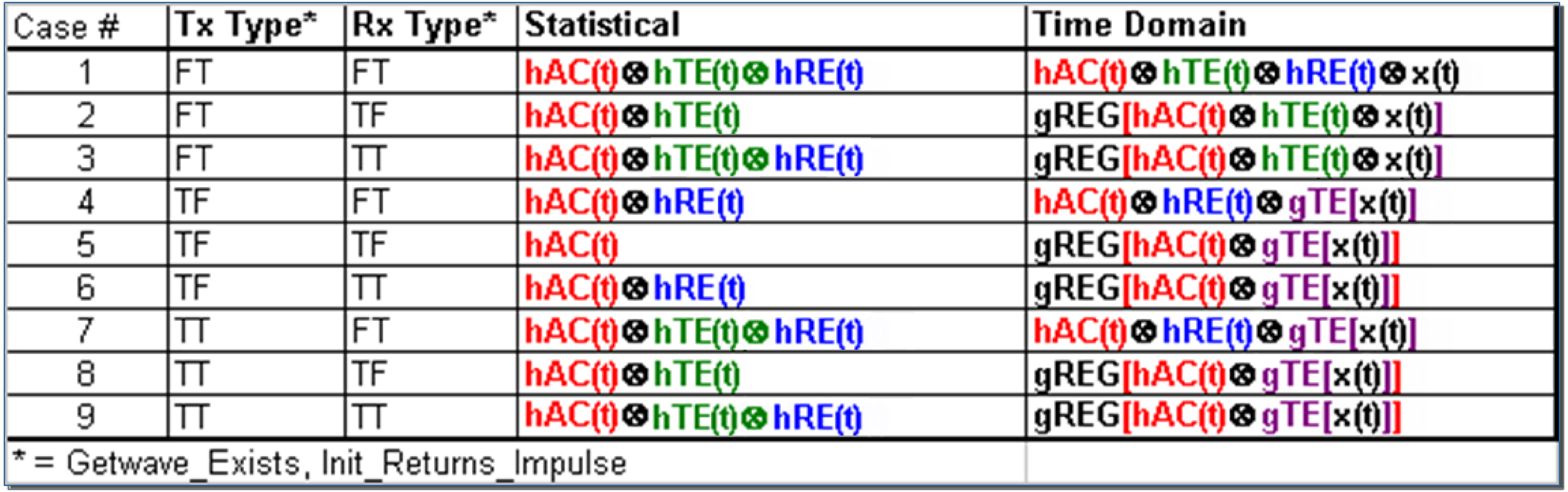


Case #	TX			RX		
	Getwave Exists	**Init_Returns_Impulse**	**Meaning**	**Getwave Exists**	**Init_Returns_Impulse**	**Meaning**
1	FALSE	TRUE	Init Model Only	FALSE	TRUE	Init Model Only
2	FALSE	TRUE	Init Model Only	TRUE	FALSE	Getwave Model Only
3	FALSE	TRUE	Init Model Only	TRUE	TRUE	Dual Model
4	TRUE	FALSE	Getwave Model Only	FALSE	TRUE	Init Model Only
5	TRUE	FALSE	Getwave Model Only	TRUE	FALSE	Getwave Model Only
6	TRUE	FALSE	Getwave Model Only	TRUE	TRUE	Dual Model
7	TRUE	TRUE	Dual Model	FALSE	TRUE	Init Model Only
8	TRUE	TRUE	Dual Model	TRUE	FALSE	Getwave Model Only
9	TRUE	TRUE	Dual Model	TRUE	TRUE	Dual Model

IBIS-AMI Simulation Terminology

- $h_{AC}(t)$ – Analog channel impulse response
- $p(t)$ – Unit pulse at target data rate
- $b(t)$ – Data bit stream suitable for convolution processing
- $h_{TE}(t)$ – Impulse response of TX AMI_Init equalization
- $h_{RE}(t)$ – Impulse response of RX AMI_Init equalization
- $g_{TE}[x(t)]$ – Waveform output of TX GetWave processing
- $g_{RE}[x(t)]$ – Waveform output of RX GetWave processing

AMI Equations for 9 TX/RX Cases

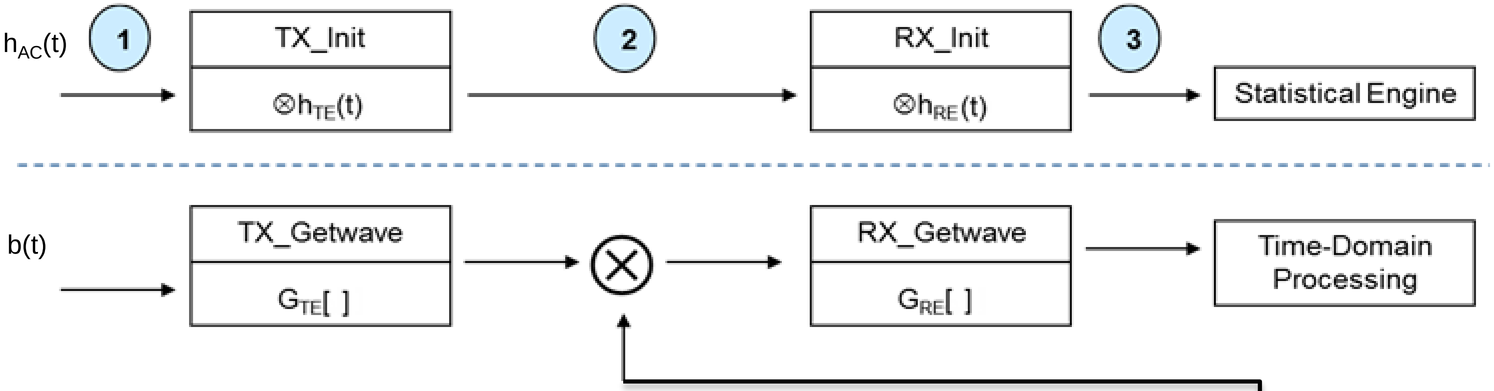


Case #	Tx Type*	Rx Type*	Statistical	Time Domain
1	FT	FT	hAC(t)⊗hTE(t)⊗hRE(t)	hAC(t)⊗hTE(t)⊗hRE(t)⊗x(t)
2	FT	TF	hAC(t)⊗hTE(t)	gREG[hAC(t)⊗hTE(t)⊗x(t)]
3	FT	TT	hAC(t)⊗hTE(t)⊗hRE(t)	gREG[hAC(t)⊗hTE(t)⊗x(t)]
4	TF	FT	hAC(t)⊗hRE(t)	hAC(t)⊗hRE(t)⊗gTE[x(t)]
5	TF	TF	hAC(t)	gREG[hAC(t)⊗gTE[x(t)]]
6	TF	TT	hAC(t)⊗hRE(t)	gREG[hAC(t)⊗gTE[x(t)]]
7	TT	FT	hAC(t)⊗hTE(t)⊗hRE(t)	hAC(t)⊗hRE(t)⊗gTE[x(t)]
8	TT	TF	hAC(t)⊗hTE(t)	gREG[hAC(t)⊗gTE[x(t)]]
9	TT	TT	hAC(t)⊗hTE(t)⊗hRE(t)	gREG[hAC(t)⊗gTE[x(t)]]

* = Getwave_Exists, Init_Returns_Impulse

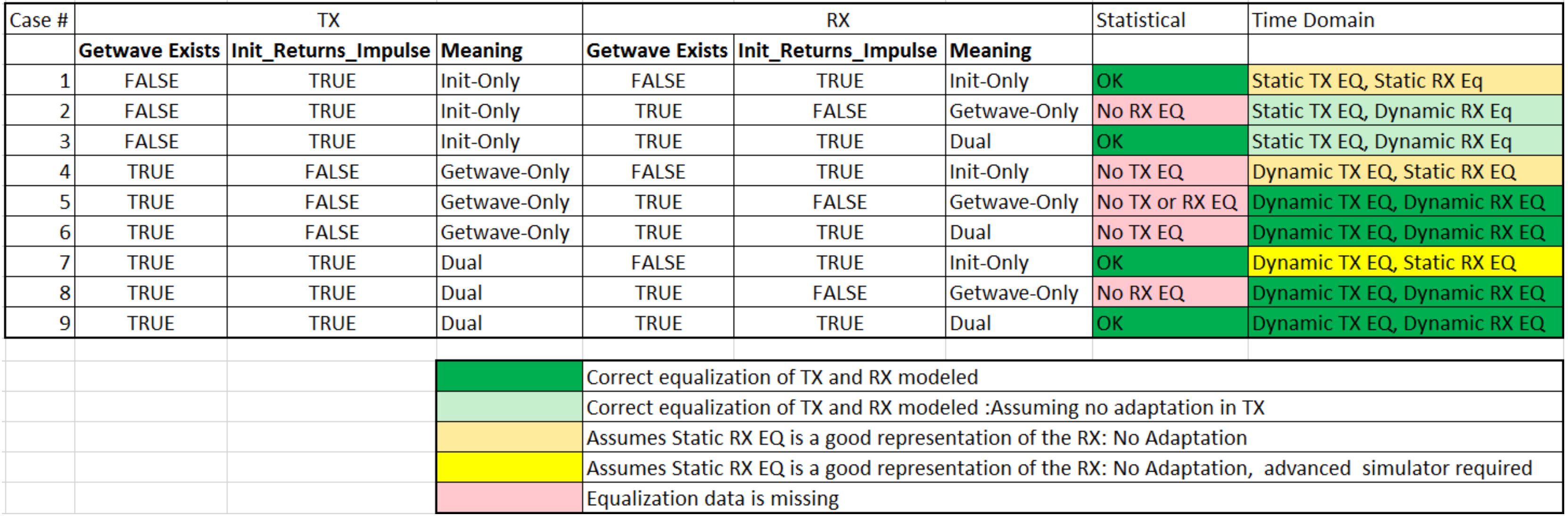
- **Allows us to efficiently & unambiguously define what simulation results are expected**

IBIS-AMI Reference Flow



Case #	TX			RX			Convolution Input
	Getwave Exists	**Init_Returns_Impulse**	**Meaning**	**Getwave Exists**	**Init_Returns_Impulse**	**Meaning**	
1	FALSE	TRUE	Init-Only	FALSE	TRUE	Init-Only	3
2	FALSE	TRUE	Init-Only	TRUE	FALSE	Getwave-Only	1 or 2
3	FALSE	TRUE	Init-Only	TRUE	TRUE	Dual	2
4	TRUE	FALSE	Getwave-Only	FALSE	TRUE	Init-Only	3
5	TRUE	FALSE	Getwave-Only	TRUE	FALSE	Getwave-Only	1,2,or 3
6	TRUE	FALSE	Getwave-Only	TRUE	TRUE	Dual	1
7	TRUE	TRUE	Dual	FALSE	TRUE	Init-Only	iFFT(FFT(3)/FFT(2))
8	TRUE	TRUE	Dual	TRUE	FALSE	Getwave-Only	1
9	TRUE	TRUE	Dual	TRUE	TRUE	Dual	1

Interpreting Simulation Results



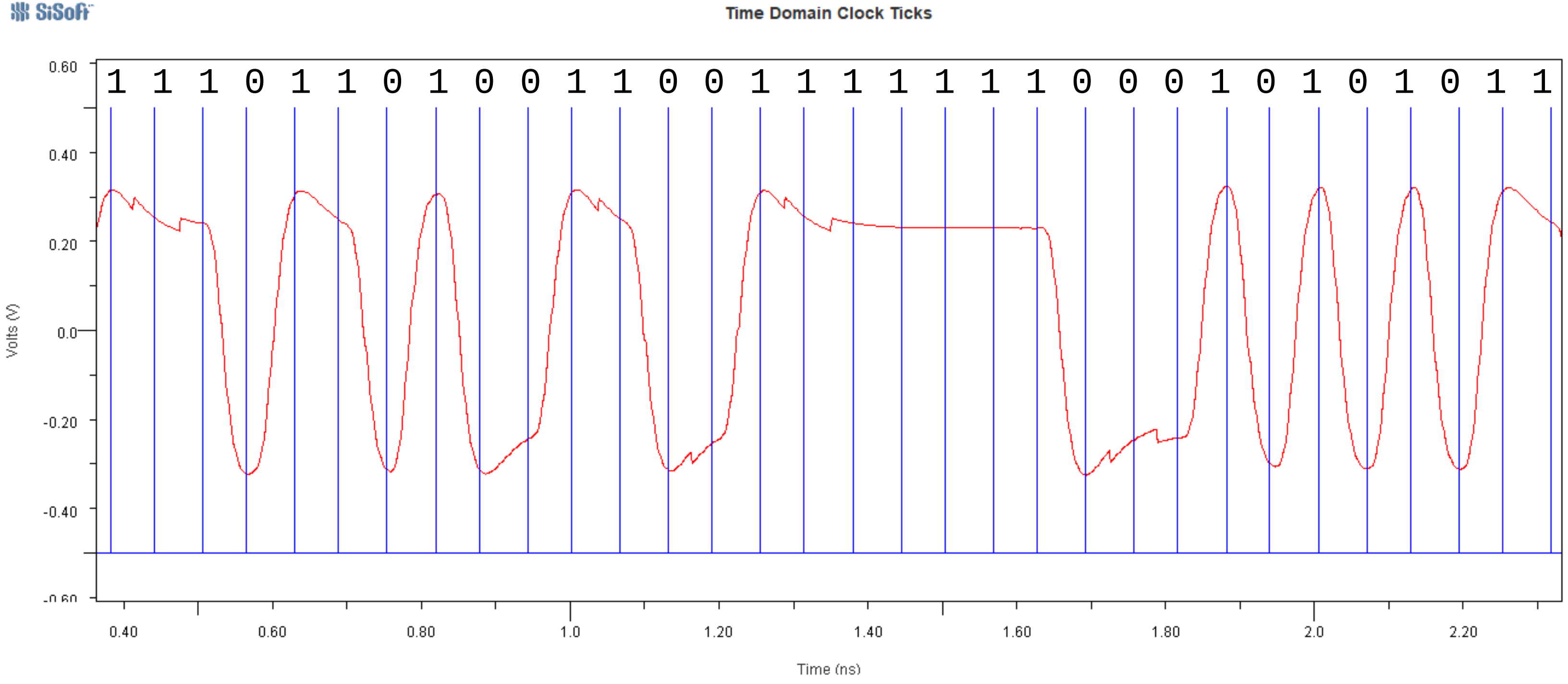
Case #	TX			RX			Statistical	Time Domain
	Getwave Exists	Init_Returns_Impulse	Meaning	Getwave Exists	Init_Returns_Impulse	Meaning		
1	FALSE	TRUE	Init-Only	FALSE	TRUE	Init-Only	OK	Static TX EQ, Static RX Eq
2	FALSE	TRUE	Init-Only	TRUE	FALSE	Getwave-Only	No RX EQ	Static TX EQ, Dynamic RX Eq
3	FALSE	TRUE	Init-Only	TRUE	TRUE	Dual	OK	Static TX EQ, Dynamic RX Eq
4	TRUE	FALSE	Getwave-Only	FALSE	TRUE	Init-Only	No TX EQ	Dynamic TX EQ, Static RX EQ
5	TRUE	FALSE	Getwave-Only	TRUE	FALSE	Getwave-Only	No TX or RX EQ	Dynamic TX EQ, Dynamic RX EQ
6	TRUE	FALSE	Getwave-Only	TRUE	TRUE	Dual	No TX EQ	Dynamic TX EQ, Dynamic RX EQ
7	TRUE	TRUE	Dual	FALSE	TRUE	Init-Only	OK	Dynamic TX EQ, Static RX EQ
8	TRUE	TRUE	Dual	TRUE	FALSE	Getwave-Only	No RX EQ	Dynamic TX EQ, Dynamic RX EQ
9	TRUE	TRUE	Dual	TRUE	TRUE	Dual	OK	Dynamic TX EQ, Dynamic RX EQ

🟩	Correct equalization of TX and RX modeled
🟢	Correct equalization of TX and RX modeled :Assuming no adaptation in TX
🟧	Assumes Static RX EQ is a good representation of the RX: No Adaptation
🟨	Assumes Static RX EQ is a good representation of the RX: No Adaptation, advanced simulator required
🟥	Equalization data is missing

- **Statistical simulations can be missing TX and/or RX equalization, depending on case**
  - Some partial statistical results are useful, others are not
- **Time-Domain simulations ALWAYS include TX & RX equalization**
  - Equalization can be either static or dynamic, depending on the case
- **Case 9 fully supports both Statistical & Time-Domain simulation**

Clocks and Jitter

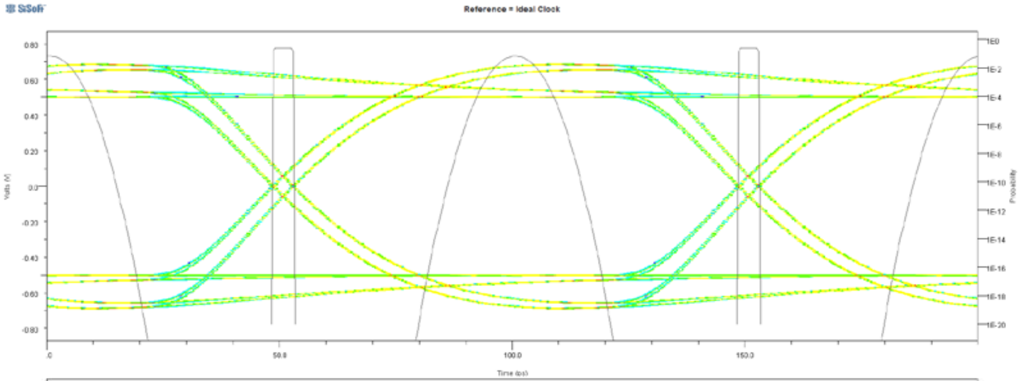
Data Latching Driven by Clock Ticks

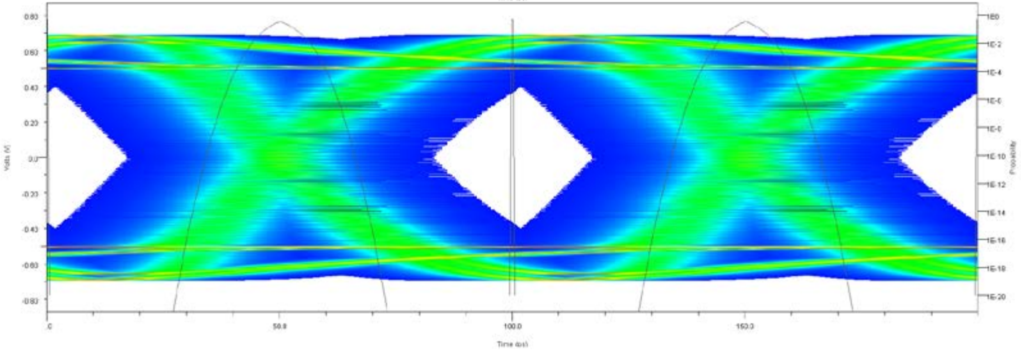


Model Outputs Can Be Viewed Different Ways

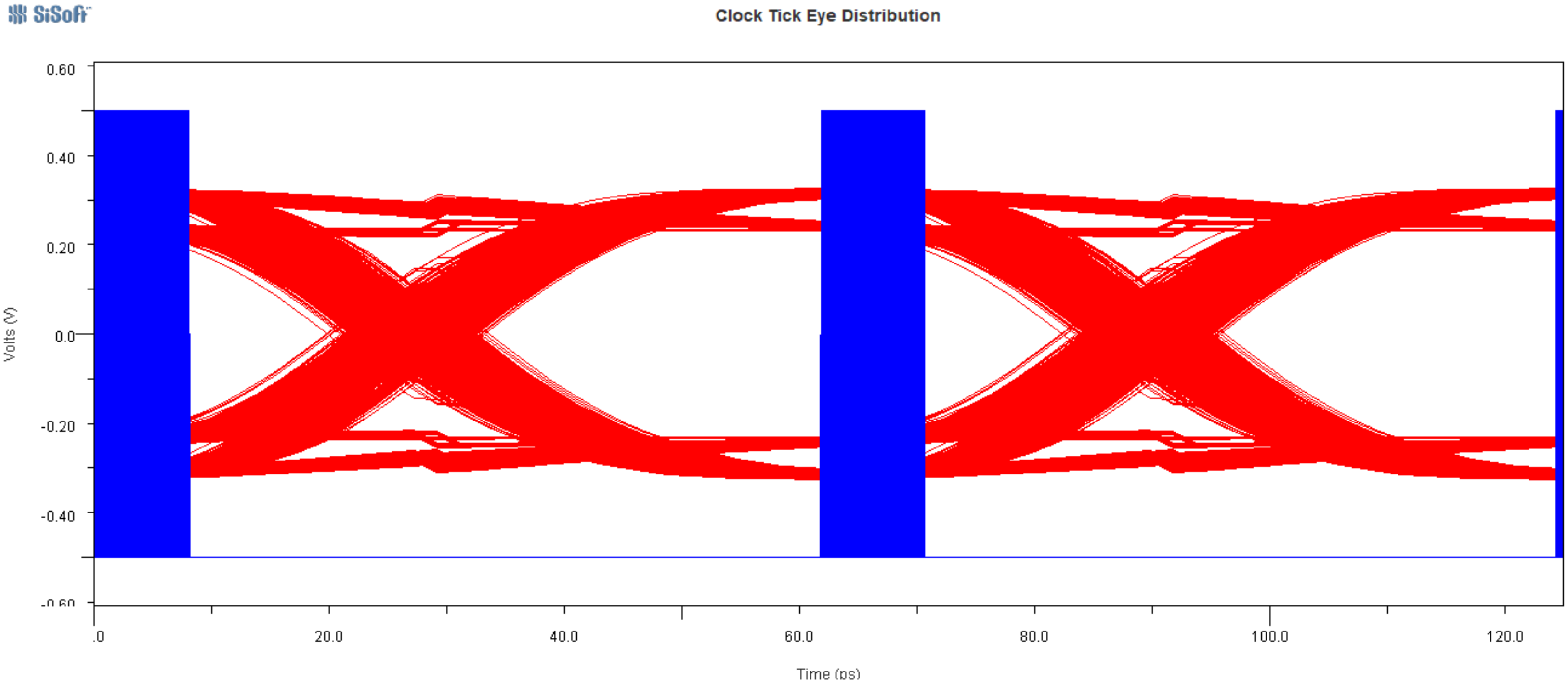
- **Model clock and data output probabilities plotted against an ideal 1 UI clock**

- **Data plotted against model clock output**





Clock Ticks are Not Perfectly Regular



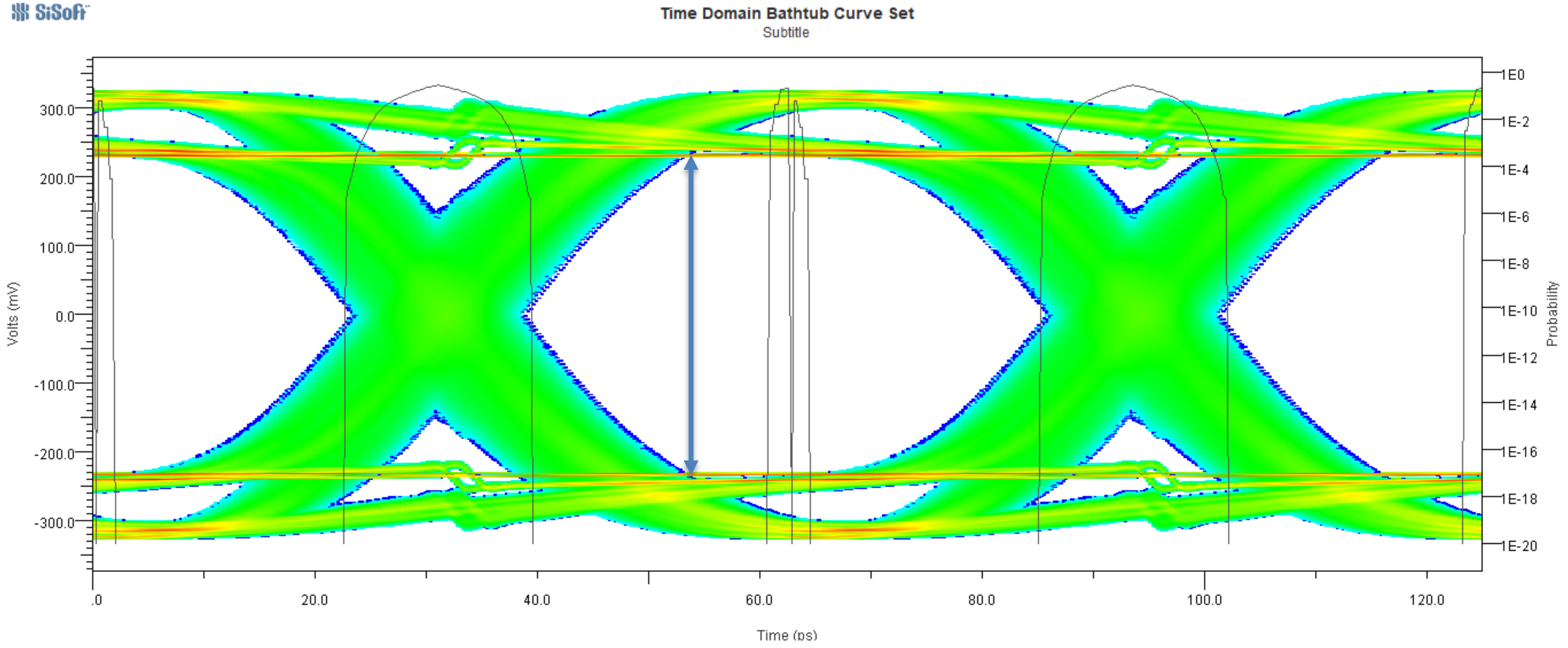
AMI_GetWave Outputs Clock Time Values

```
long AMI_GetWave(
  double *wave_in,
  long wave_size,
  double *clock_times,
  char **AMI_parameters_out,
  void *AMI_memory );
```

Clock Ticks are Not Perfectly Regular

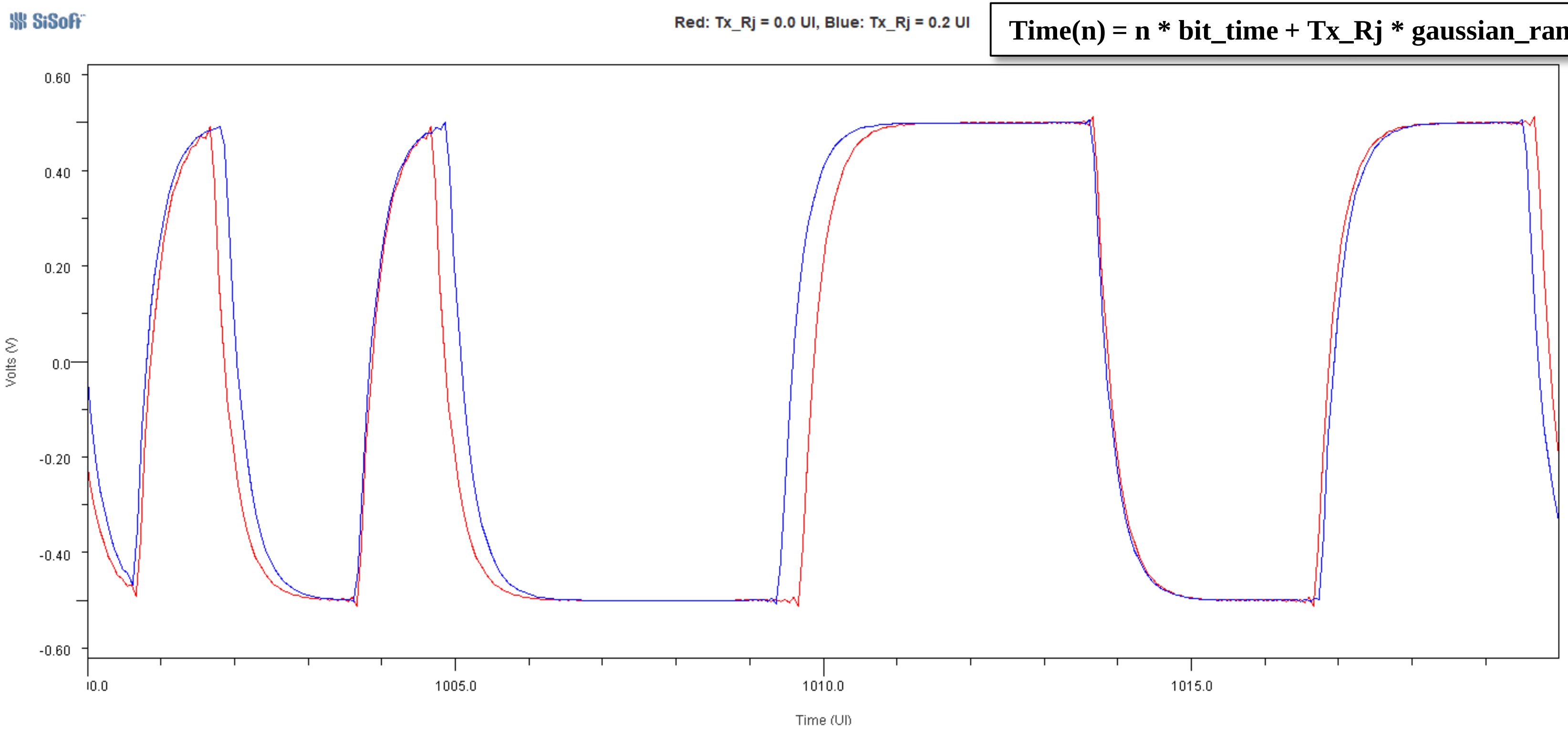
UI#	clock_times	period
997,510	62,344,398.5 ps	62.5 ps
997,511	62,344,461.0 ps	62.5 ps
997,512	62,344,523.5 ps	62.5 ps
997,513	62,344,586.0 ps	62.5 ps
997,514	62,344,648.5 ps	62.5 ps
997,515	62,344,711.0 ps	62.5 ps
997,516	62,344,773.5 ps	62.5 ps
997,517	62,344,836.5 ps	63.0 ps
997,518	62,344,899.0 ps	62.5 ps
997,519	62,344,961.5 ps	62.5 ps
997,520	62,345,024.0 ps	62.5 ps

Clocks Are Not Always at the Greatest Eye Height

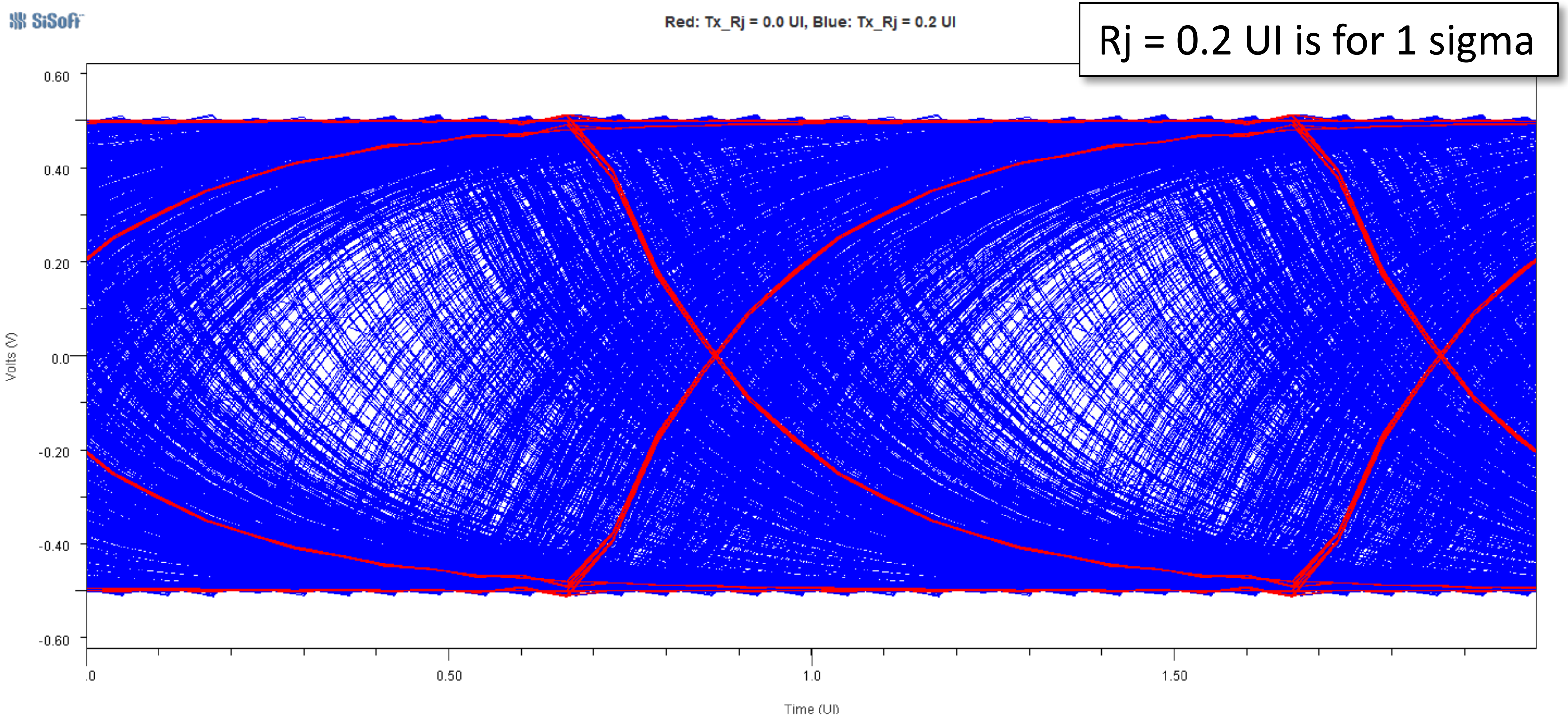


Tx_Rj Jitter Modulating the Tx Output

Time(n) = n * bit_time + Tx_Rj * gaussian_rand()



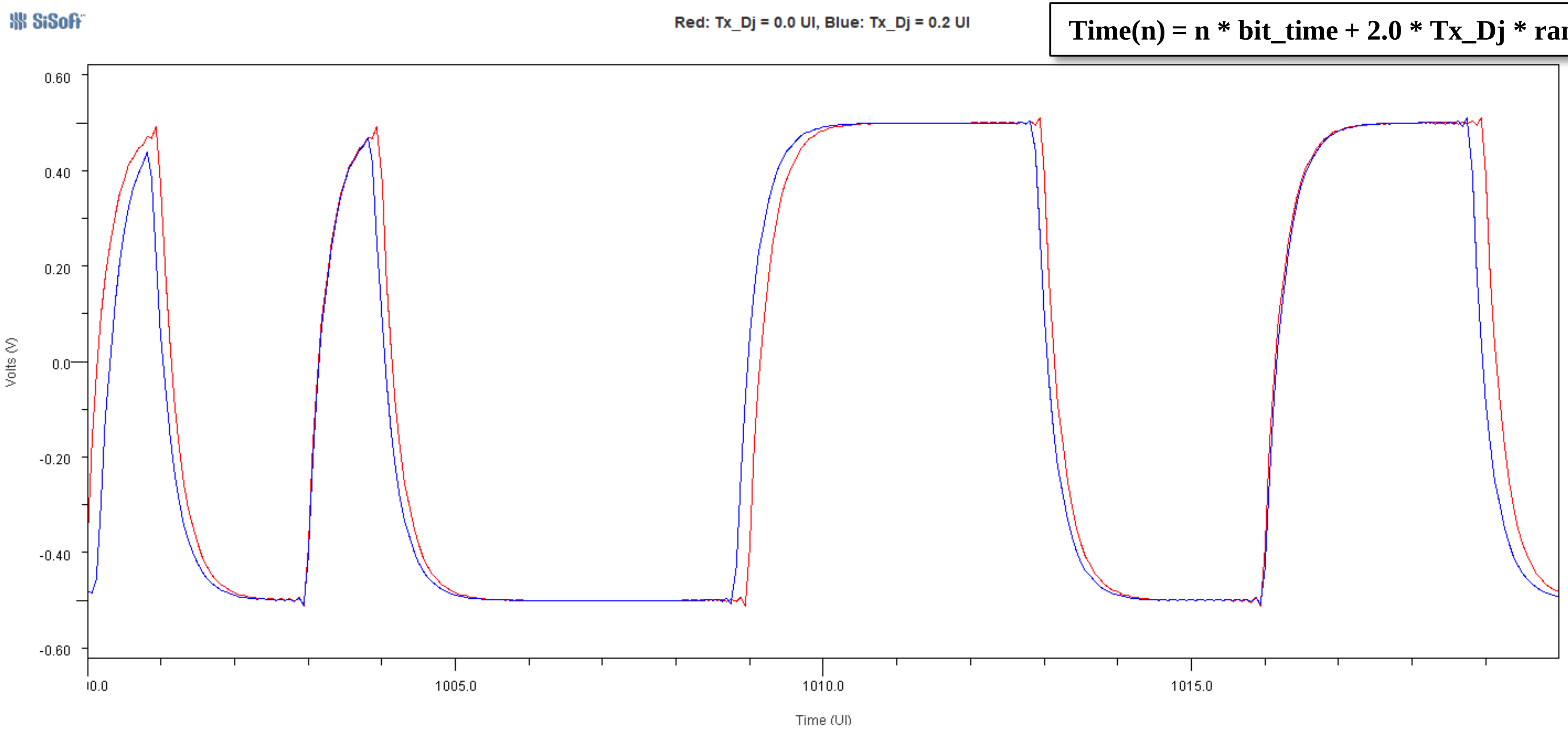
Tx_Rj Jitter Modulating the Tx Output



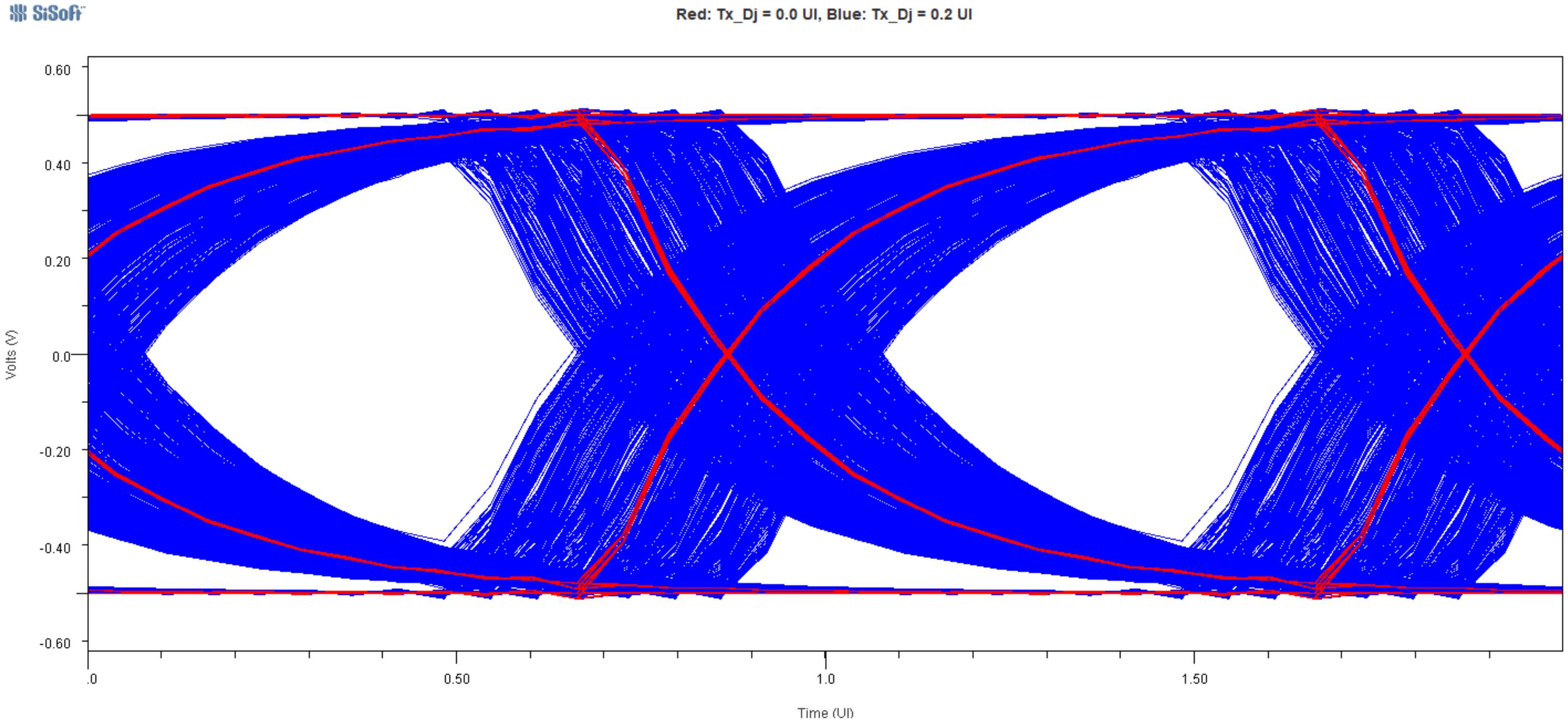
Rj = 0.2 UI is for 1 sigma

Tx_Dj Jitter Modulating the Tx Output

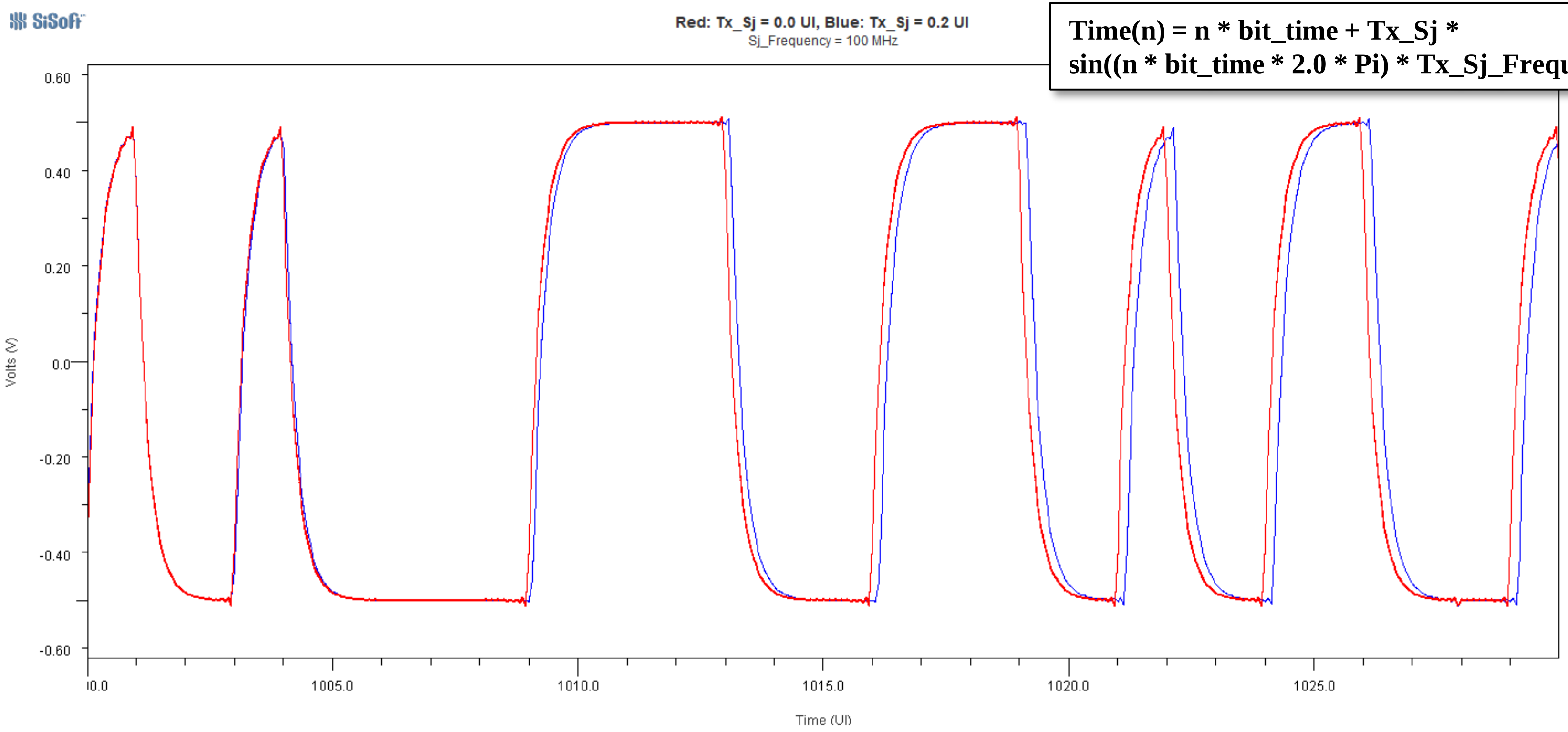
Time(n) = n * bit_time + 2.0 * Tx_Dj * rand()



Tx_Dj Jitter Modulating the Tx Output

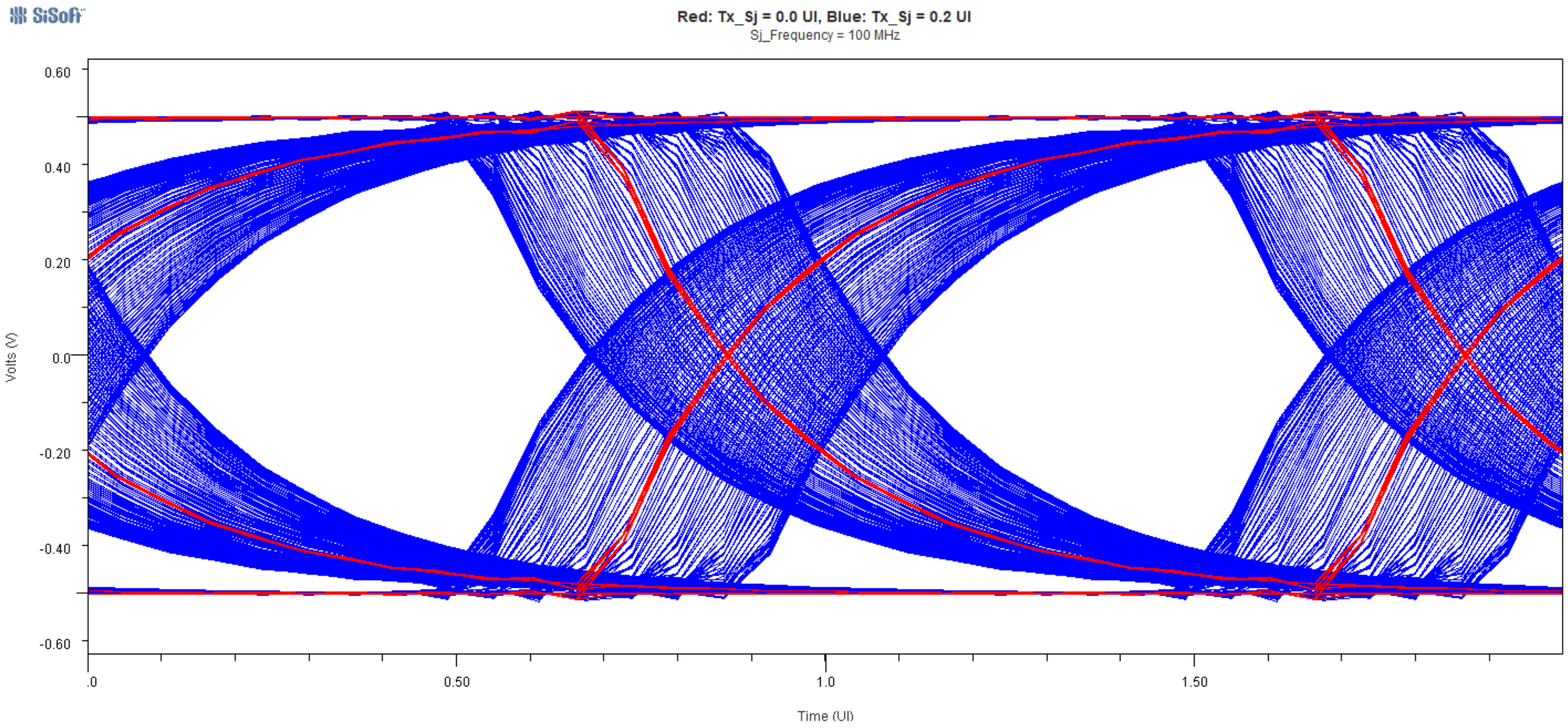


Tx_Sj Jitter Modulating the Tx Output



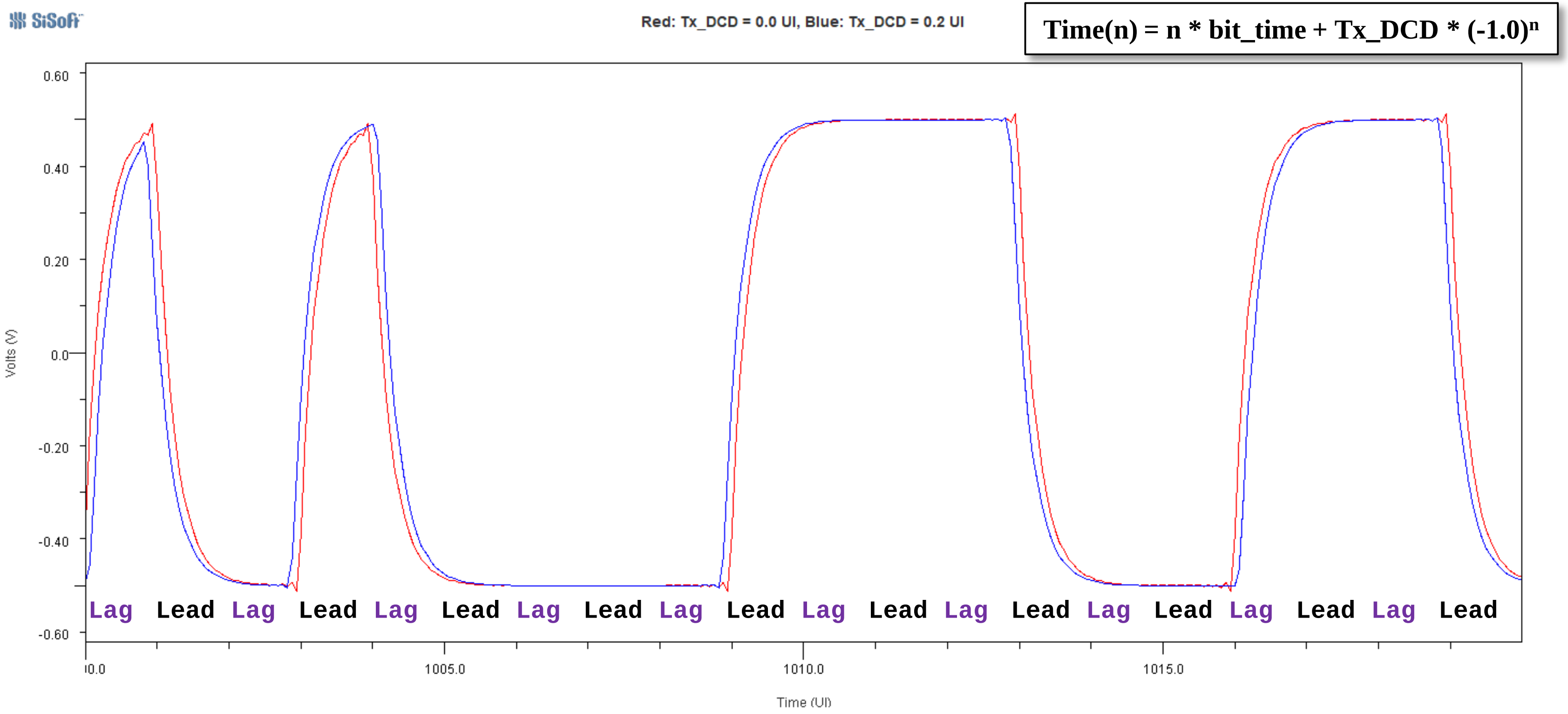
**Time(n) = n * bit_time + Tx_Sj *
sin((n * bit_time * 2.0 * Pi) * Tx_Sj_Frequency)**

Tx_Sj Jitter Modulating the Tx Output

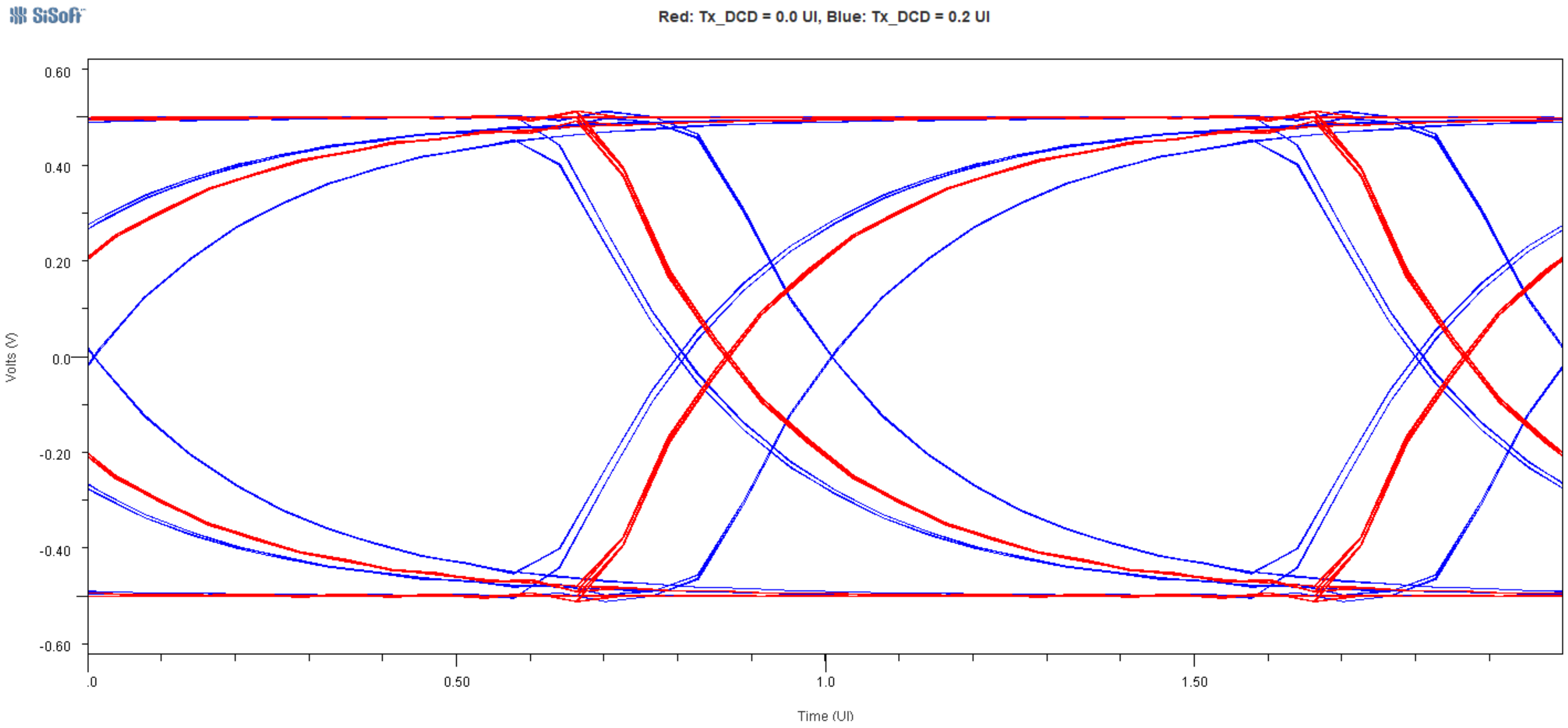


Tx_DCD Jitter Modulating the Tx Output

$$\text{Time}(n) = n * \text{bit\_time} + \text{Tx\_DCD} * (-1.0)^n$$

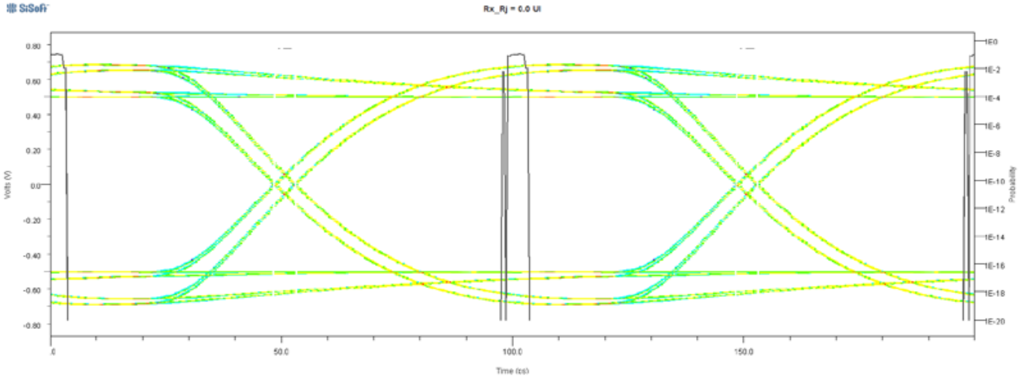


Tx_DCD Jitter Modulating the Tx Output

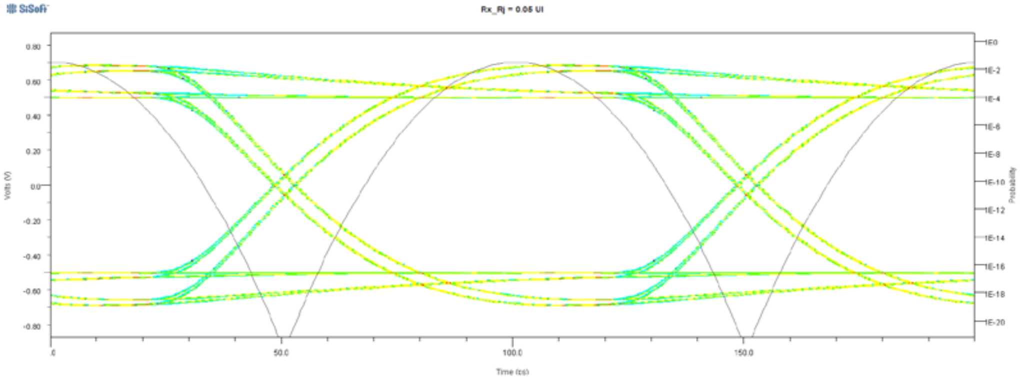


Rx_Rj Modulating the Sampling Clock

- Rx_Rj = 0.00 UI

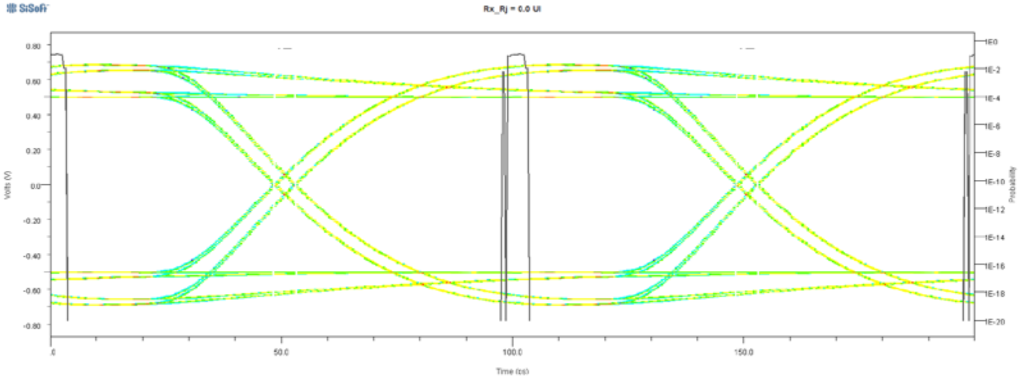


- Rx_Rj = 0.05 UI

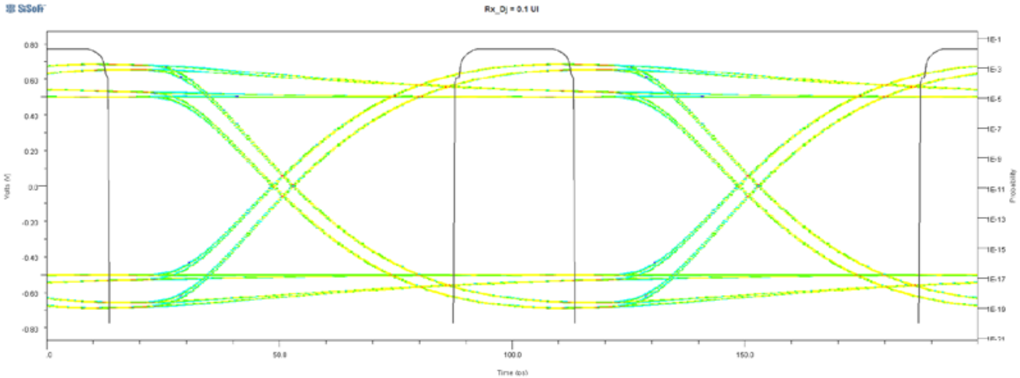


Rx_Dj Modulating the Sampling Clock

- **Rx_Dj = 0.00 UI**

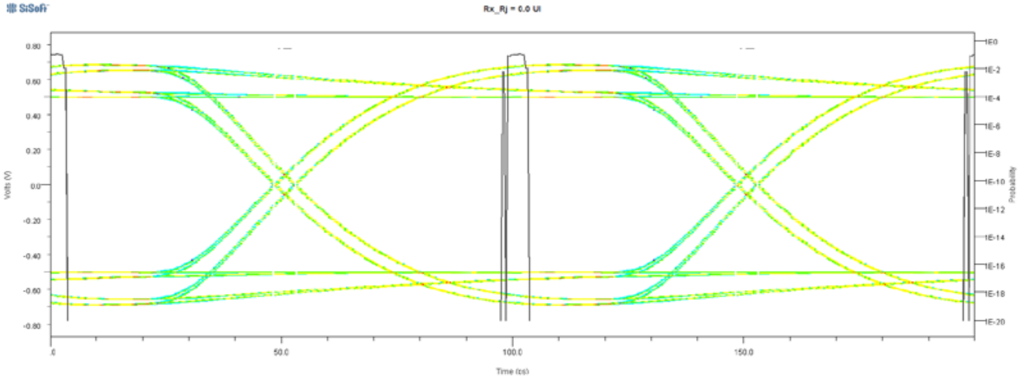


- **Rx_Dj = 0.10 UI**

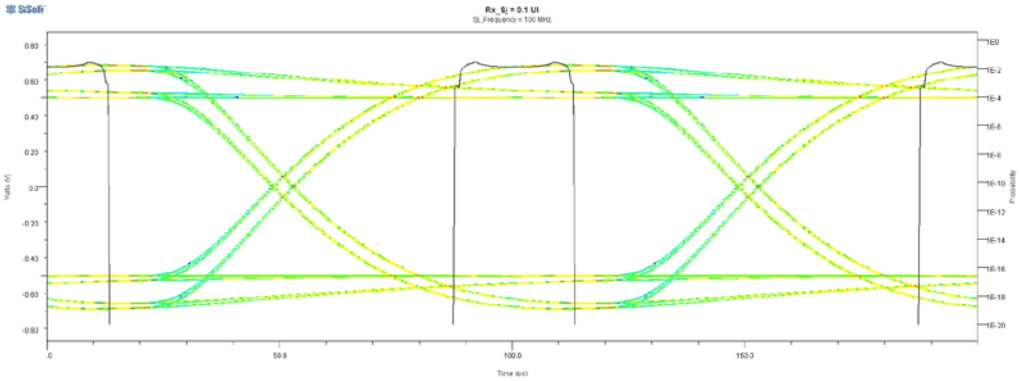


Rx_Sj Modulating the Sampling Clock

- **Rx_Sj = 0.00 UI**

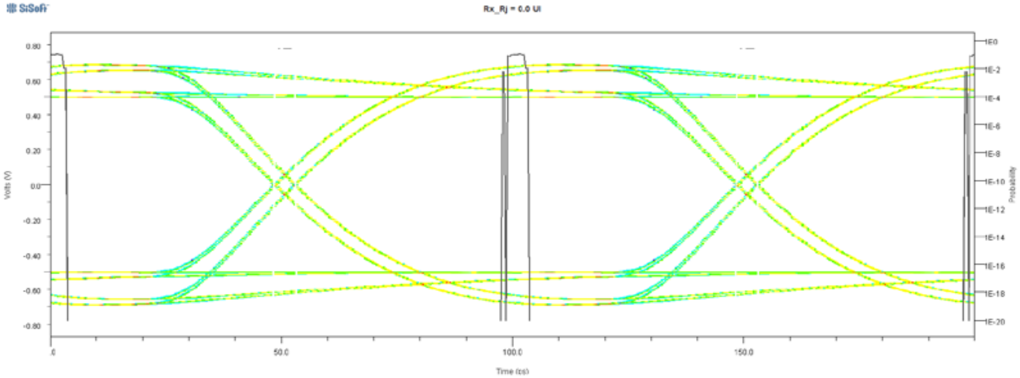


- **Rx_Sj = 0.10 UI**

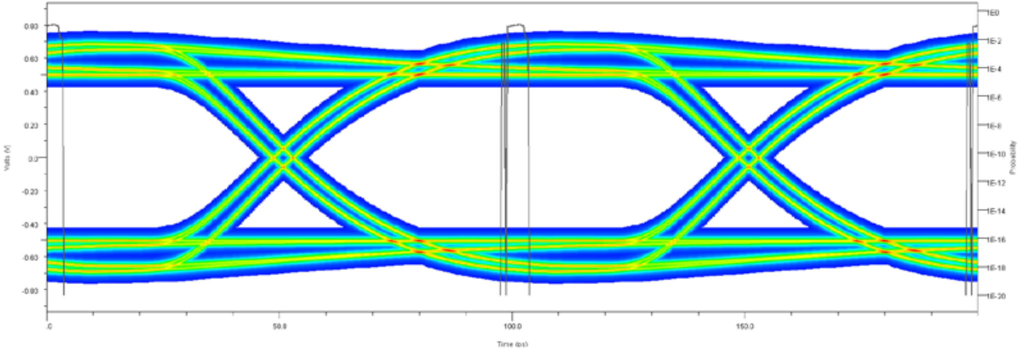


Rx_Noise Modulates the Sampling Latch Input

- **Rx_Noise = 0.000 V**

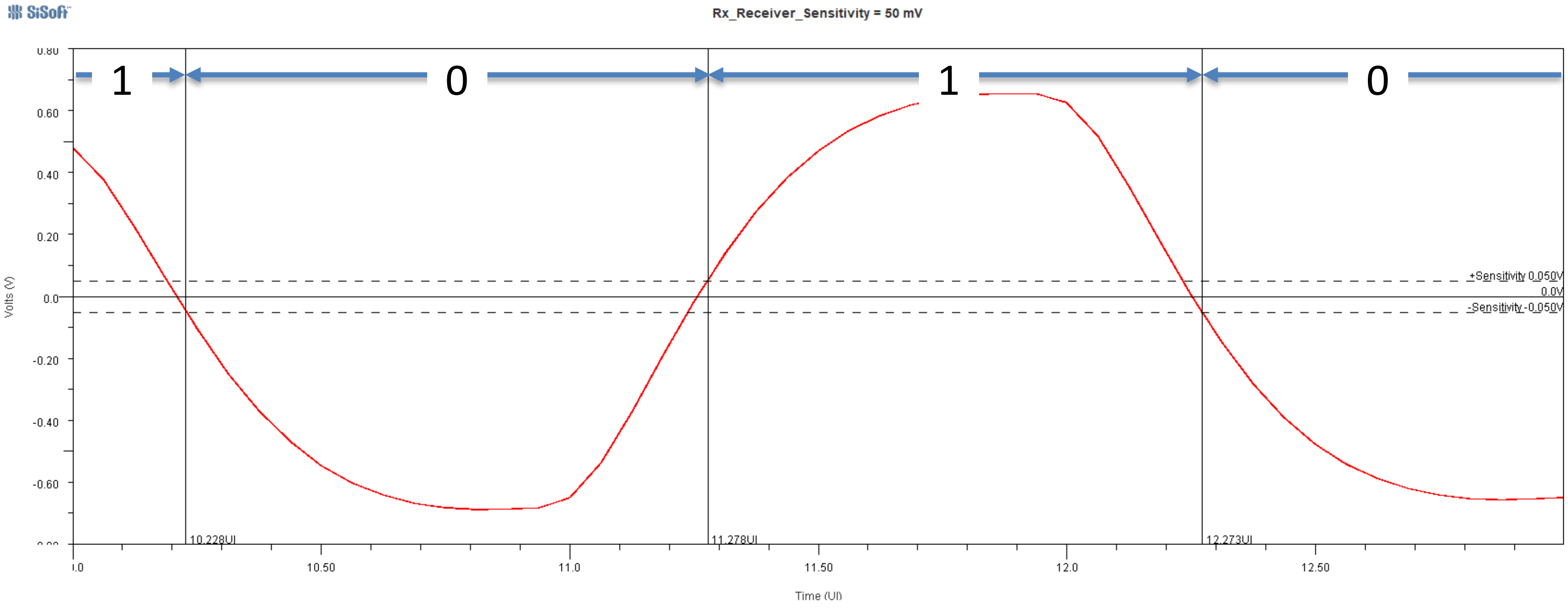


- **Rx_Noise = 0.005 V**



IBIS 7.0 probably will have Rx_Gaussian_Noise and Rx_Uniform_Noise (BIRD188.1)

Rx_Receiver_Sensitivity Applies Hysteresis



Jitter Can Be Handled Directly by Some Rx AMI Models

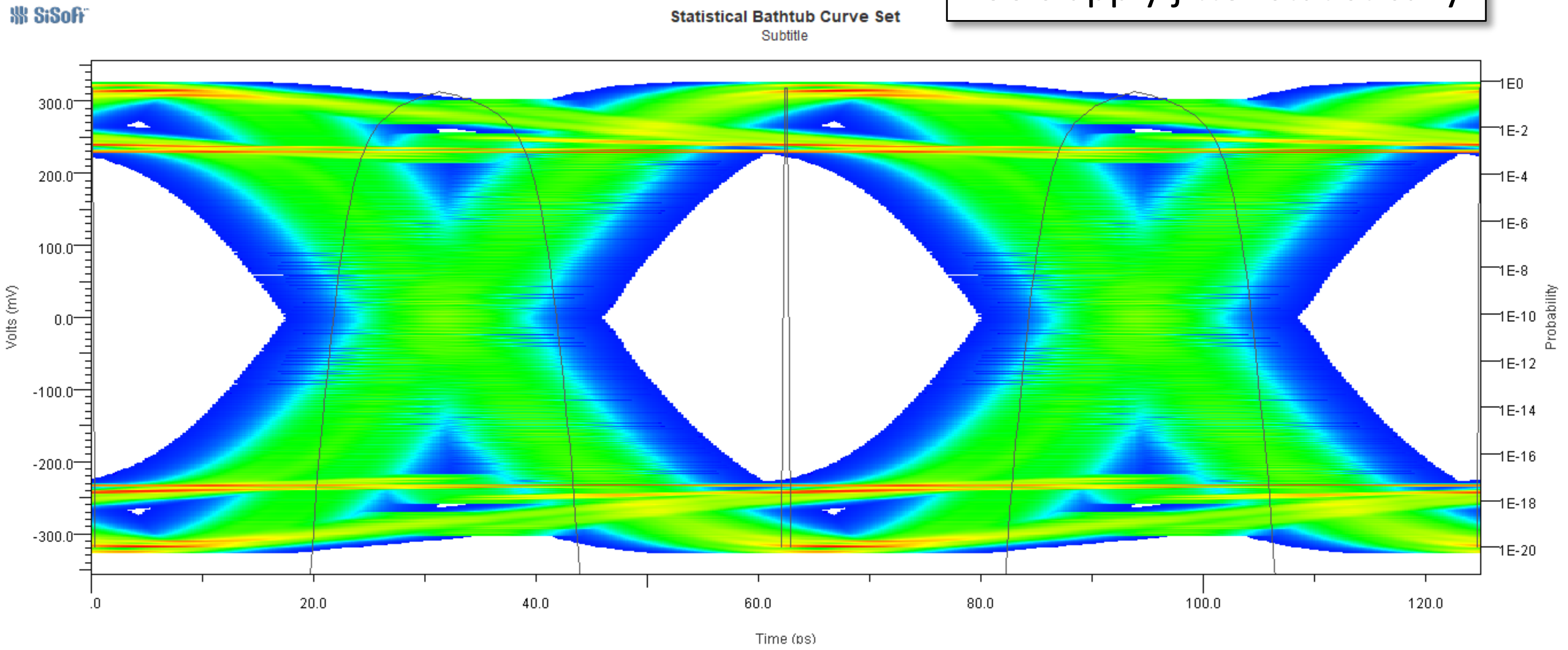
- All of the preceding slides show jitter handled by the EDA tool
- IBIS does not specify for all jitter types exactly how tools do that
- Some Rx IBIS-AMI models will jitter their clock output
- Jitter modeled internally by AMI_GetWave is reported to the tool:
    - Rx_Clock_PDF
    - Rx_Clock_Recovery_Rj
    - Rx_Clock_Recovery_Dj
    - Rx_Clock_Recovery_Sj
    - Rx_Clock_Recovery_DCD
- Tools must not add jitter if the model has already done so

But Clock Times Output is Not Required

- **IBIS does not require AMI_GetWave() to produce clock_times at all**
- **In this case tools are expected to supply clock recovery using the following AMI parameters:**
  - Rx_Clock_Recovery_Mean
  - Rx_Clock_Recovery_Rj
  - Rx_Clock_Recovery_Dj
  - Rx_Clock_Recovery_Sj
  - Rx_Clock_Recovery_DCD

No Time Domain Clock in Statistical Analysis

Tools apply jitter statistically



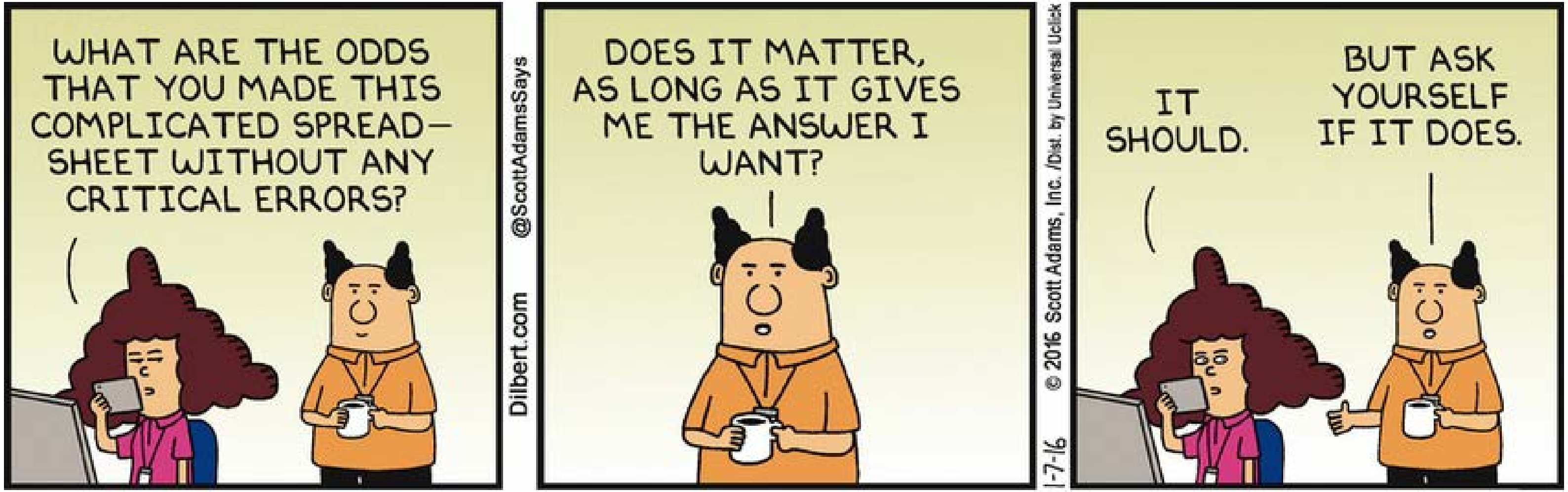
Summary - Clocks and Jitter

- **Eye height only really matters where the signal is sampled**
- **AMI Rx models return equalized waveforms & clock ticks (GetWave)**
- **Results post-processing and presentation is simulator-specific**
- **Simply reporting maximum eye height without considering the clock is wishful thinking**
- **Jitter is not automatic in an AMI model – someone put it there**
- **AMI jitter / noise facilities**
  - Tx jitter directly modulates the Tx output timing
  - Rx jitter modulates the sampling clock timing
  - Rx noise modulates the sampling latch input amplitude

Trusting Simulation Results

IBIS-AMI Simulation Craftsmanship

Quality in Today's Culture


WHAT ARE THE ODDS THAT YOU MADE THIS COMPLICATED SPREAD-SHEET WITHOUT ANY CRITICAL ERRORS?
Dilbert.com @ScottAdamsSays
DOES IT MATTER, AS LONG AS IT GIVES ME THE ANSWER I WANT?
1-7-16 © 2016 Scott Adams, Inc. /Dist. by Universal Uclick
IT SHOULD.
BUT ASK YOURSELF IF IT DOES.

www.dilbert.com January 7, 2016

Dr. Eric Bogatin’s Rule #9

Never perform a measurement or simulation without first anticipating what you expect to see

Be a Simulation Craftsperson!

- **Validate your data before use.**
  - If you don’t know how – ask, experiment, learn.

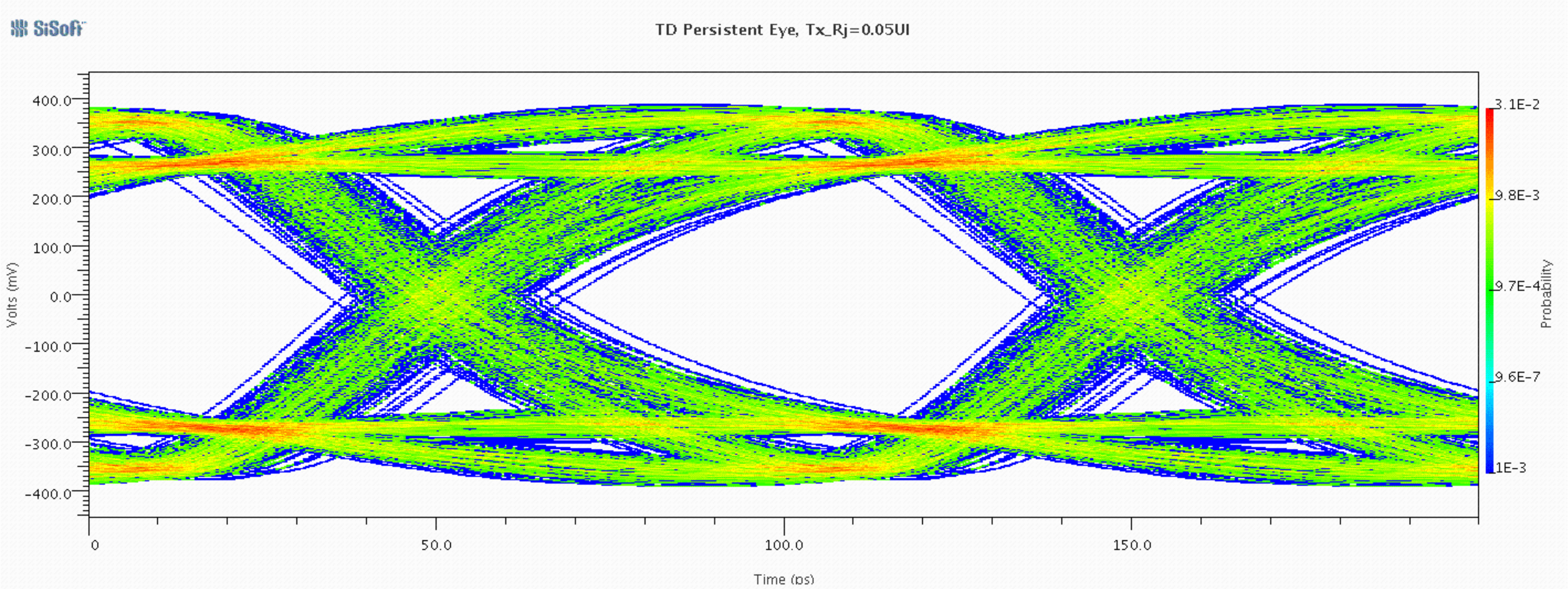
- **Take the time to understand your tools and processes.**
  - Know what results you expect. Question what doesn’t look right.

- **Collaborate, collaborate, collaborate.**
  - Complex, inter-related projects and blind assumptions are not compatible.

- **Quality is your responsibility!**

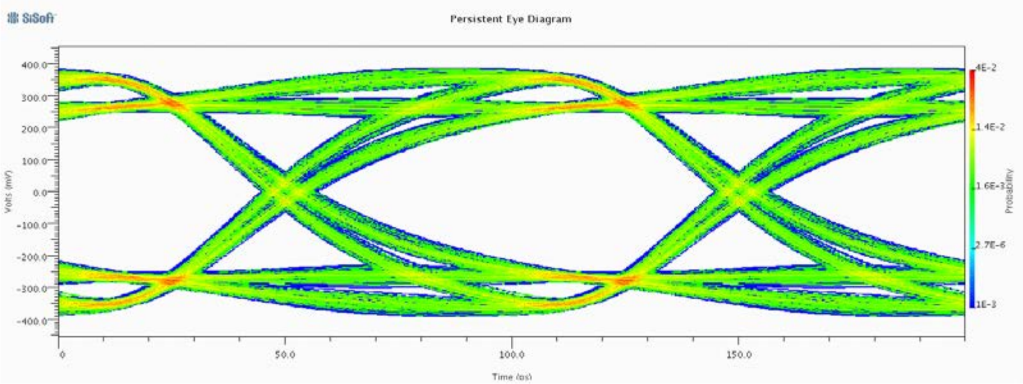
First Simulation: Is This Result Accurate?

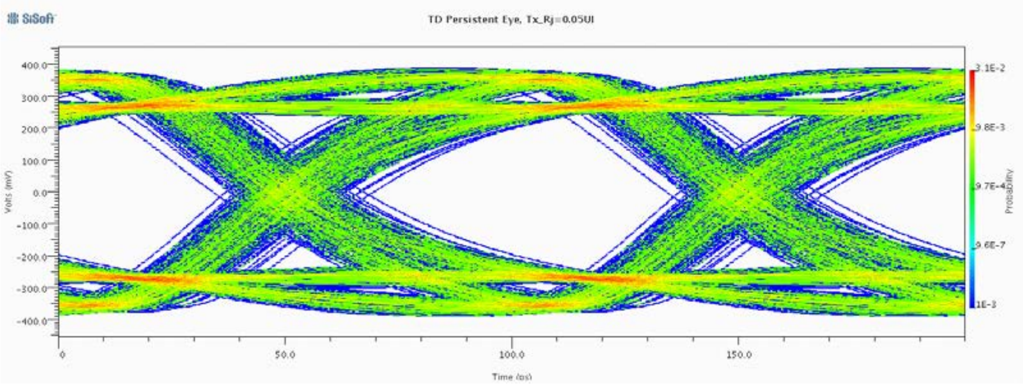
Hey, at least it runs! But was I expecting this much margin?



Is Jitter In the Model?

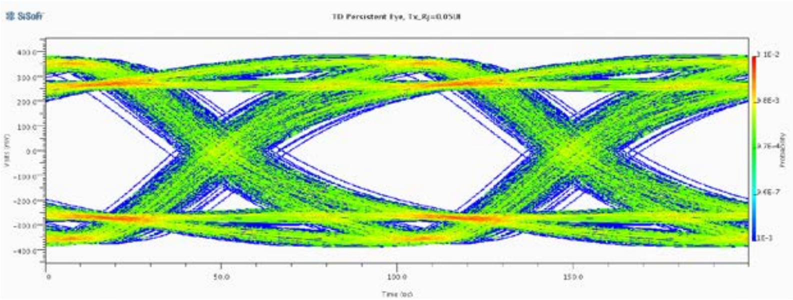
- **Turn jitter on and off to see if it makes the expected difference**



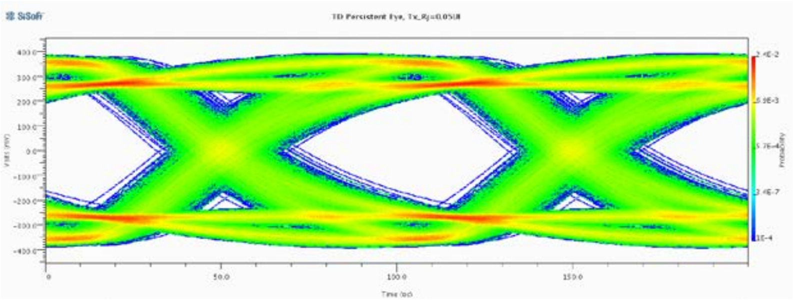


Did I Simulate Enough Bits?

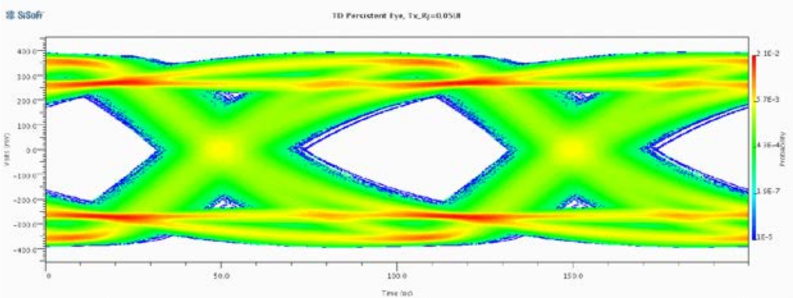
1,000 UI



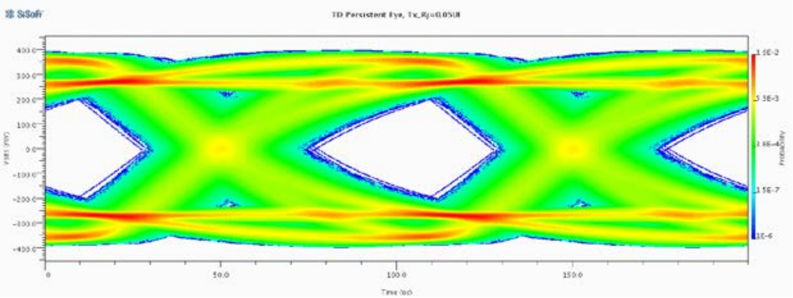
10,000 UI



100,000 UI



1,000,000 UI

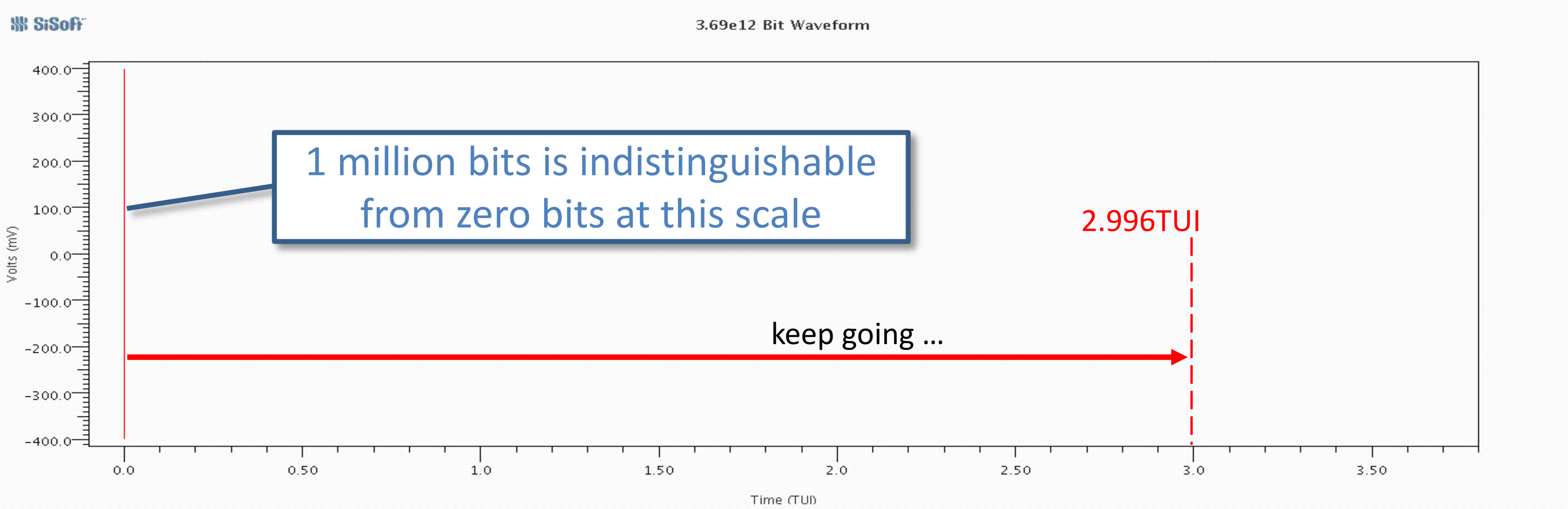


First, What Maximum BER Can I Tolerate?

▪ IEEE-802.3bj-KR4	FEC on	1e-5	
▪ IEEE-802.3bj-KR4	FEC off	1e-12	if low latency required
▪ OIF-CEI-56G FEC on		1e-4	
▪ OIF-CEI-56G FEC off		1e-20	
▪ PCIe-G3		1e-12	
▪ PCIe-G4		1e-12	
▪ DDR4		1e-12	eye mask rules
▪ DDR5		TBD	

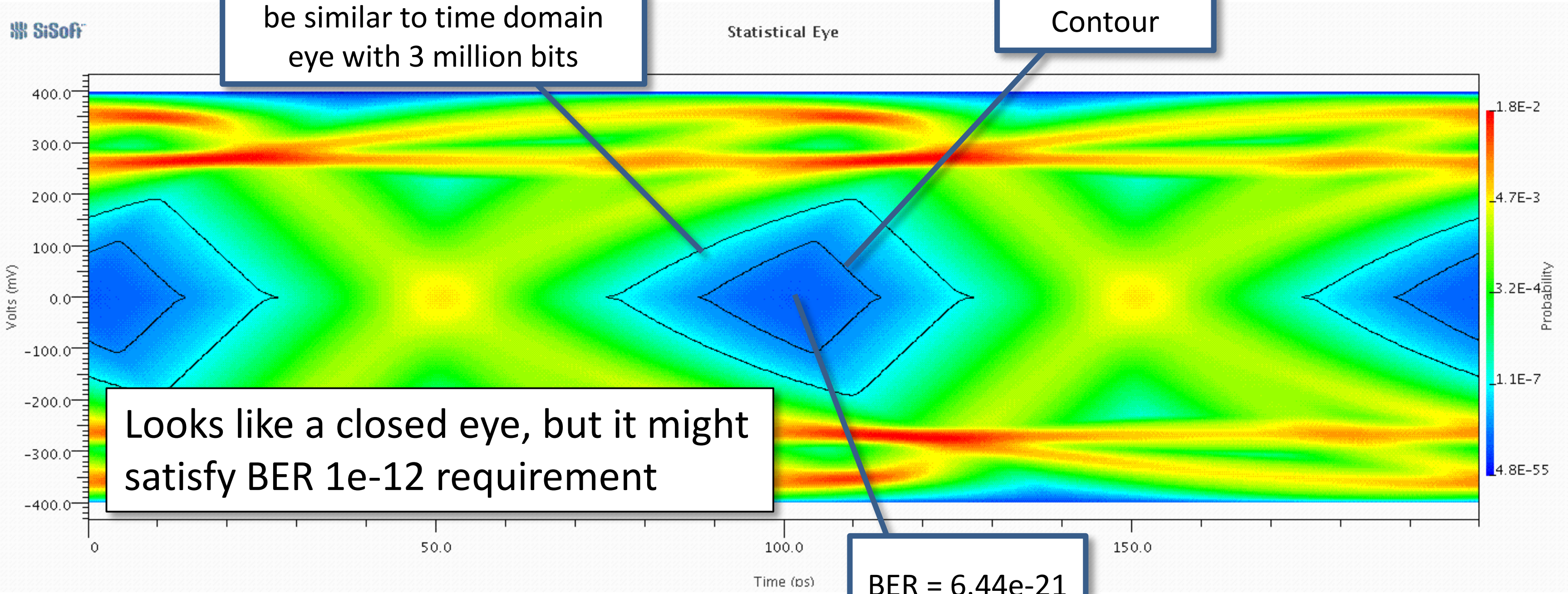
How Many Error-Free Bits for 1e-12 BER?

For 95% confidence of 1e-12 BER we need to run about 3e12 random bits with zero errors



1 million bits is indistinguishable from zero bits at this scale

Compare With the Statistical Eye



BER = 1e-6 Contour should be similar to time domain eye with 3 million bits

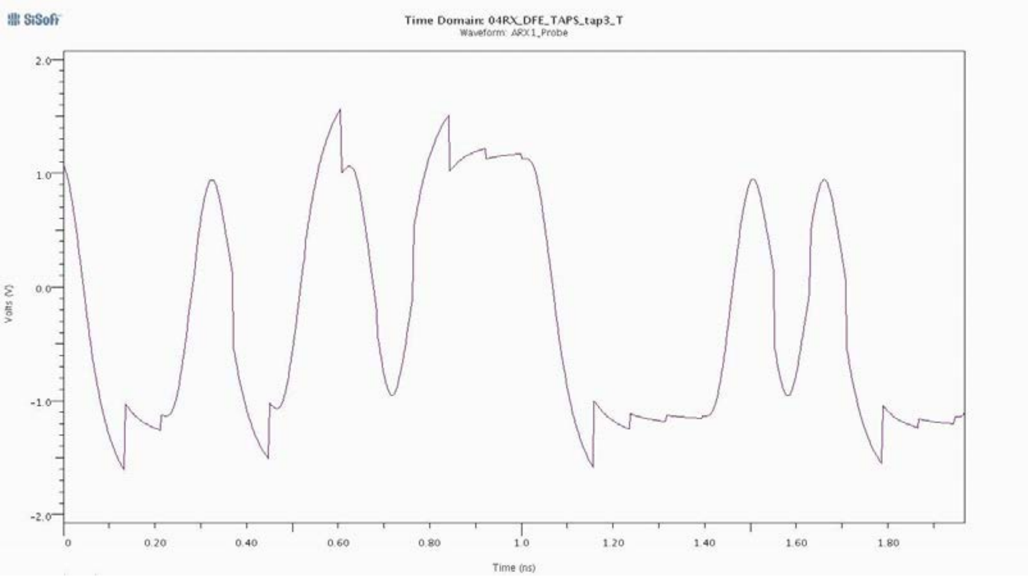
BER = 1e-12 Contour

Looks like a closed eye, but it might satisfy BER 1e-12 requirement

BER = 6.44e-21

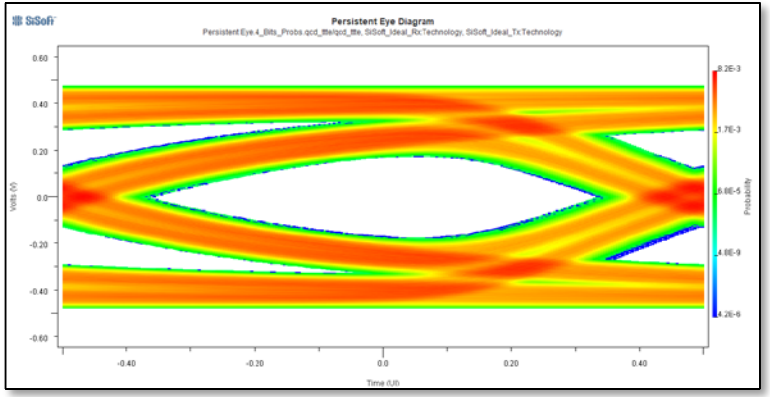
But Statistical Analysis Might Not Model the DFE

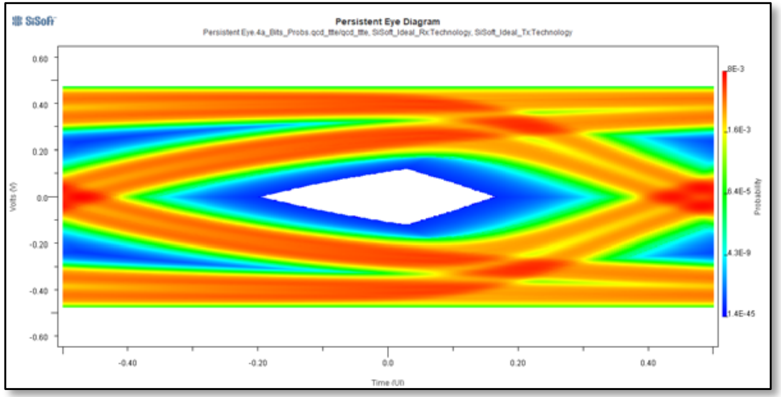
- The AMI_Init function is called once, so it can't respond to anything that adapts over time
- AMI_GetWave is called repeatedly, so it can
- AMI_Init can model DFE action, if it can somehow determine the settled DFE coefficients



Is My Simulator Extrapolating?

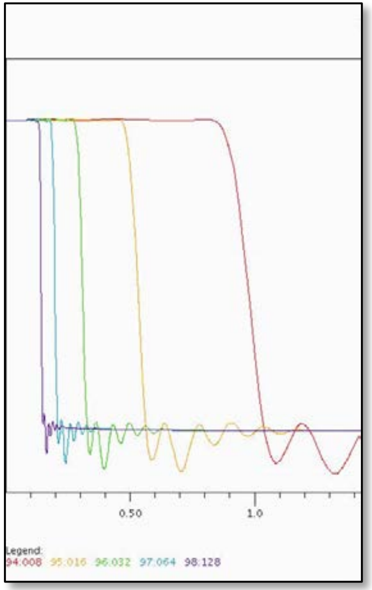
- Statistical extrapolation of time domain results combines the benefits of both domains
- Without it the eye opening may be optimistic
- Turn extrapolation on and off to see if statistical jitter seems right

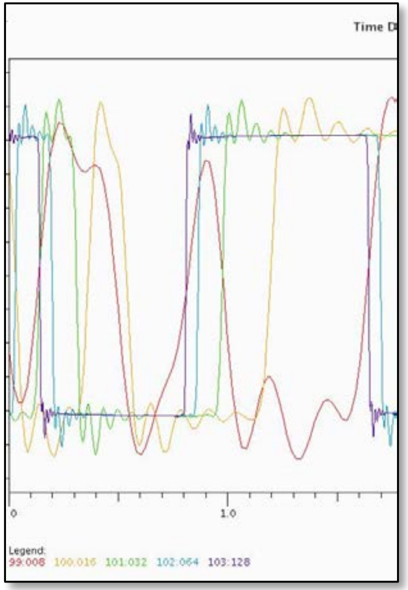


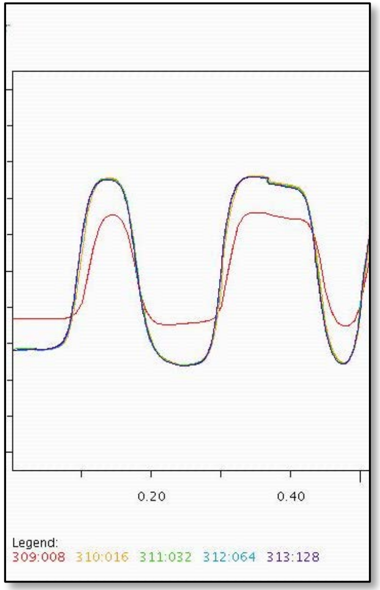


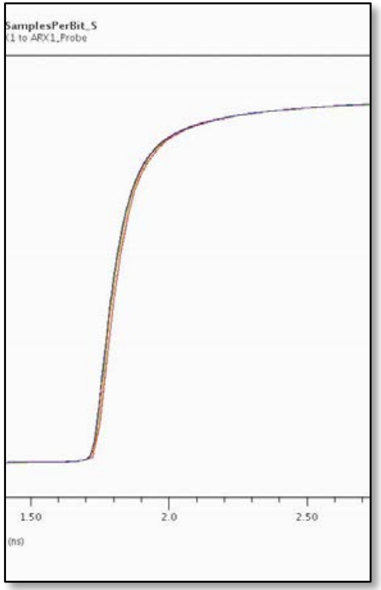
Are the IBIS-AMI Models Fully Compliant?

- **IBIS-AMI models are required to work at any samples/bit value**
- **If results vary, which result (if any) is correct?**



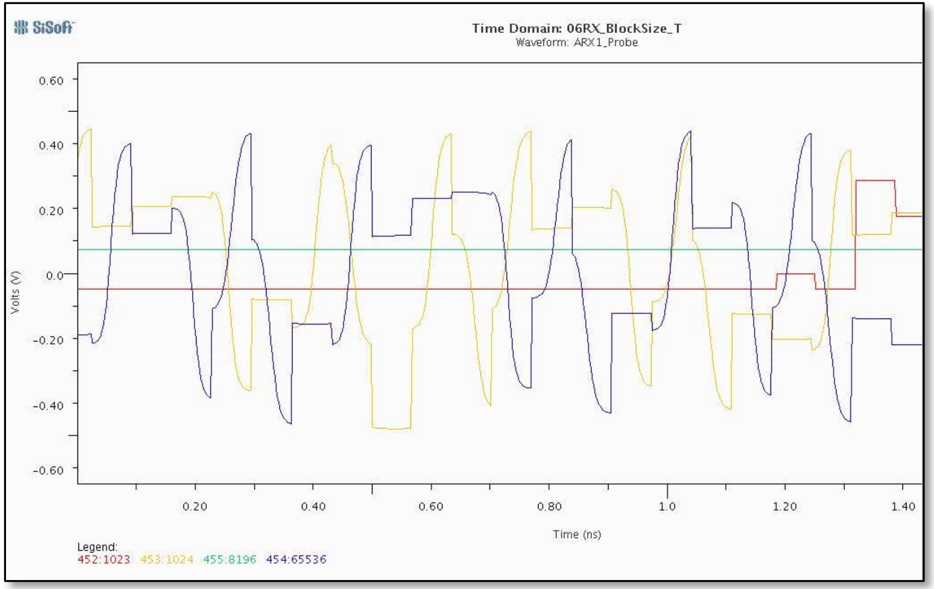


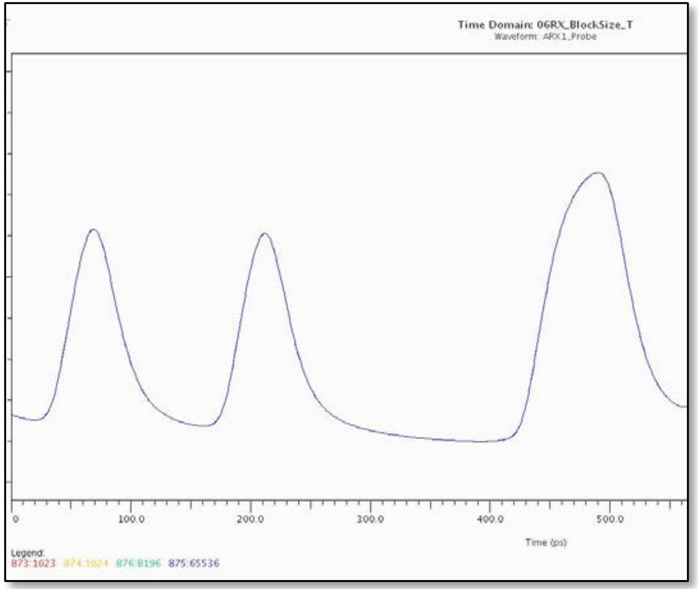




Check Block Size Too

- Block size should make no difference at all





Summary – Trusting Simulation Results

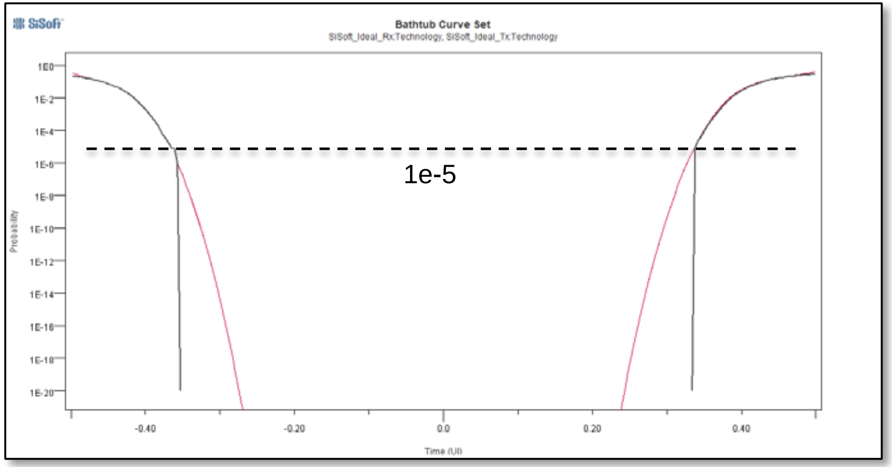
- Just because it runs, that doesn't mean it's right
- Just because you have an open eye, doesn't mean it's right
- Factors included in a complete channel simulation
- Eric Bogatin's Rule #9
- Starting simple
- Disabling / enabling simulation elements
- Debugging tips

Useful Tips and Tricks

Statistical vs. Time-Domain Simulation – There's No Perfect Answer!

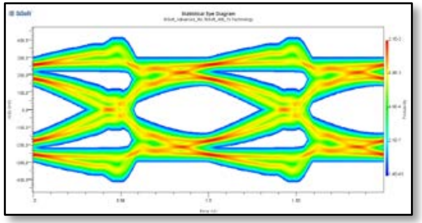
	Statistical	Time-Domain
Stimulus	Random, unlimited length	User-defined, # bits simulated
Statistical richness	>1e50	# bits simulated
Equalization	Static only	Static or dynamic
EQ adaptation	Final value only	Yes
Clock recovery	From simulator & modified	From model & modified
CDR tracking	No	Yes

Time-Domain Extrapolation

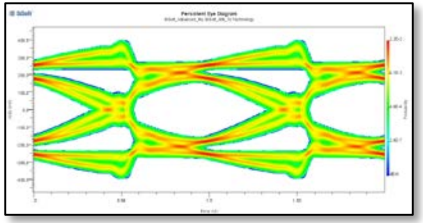


- … for those of you who said, "But I run 100,000 bits and plot the bathtub to 1e-12 all the time!"
- **Yes you do, BUT**
  - Results below 1e-5 are extrapolated by the simulator
  - Simulator extrapolation is tool-specific
    - What algorithm does it use?
    - Are Tx/Rx jitter factored in?
    - Does it include low-probability ISI? Crosstalk?

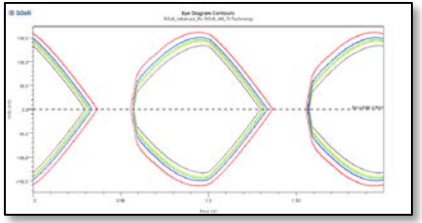
Comparing Statistical / Time-Domain Results

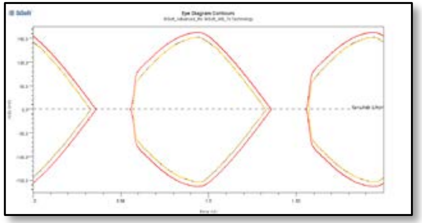


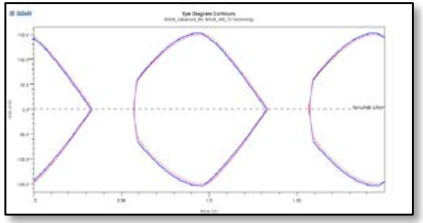
Statistical



Time-Domain





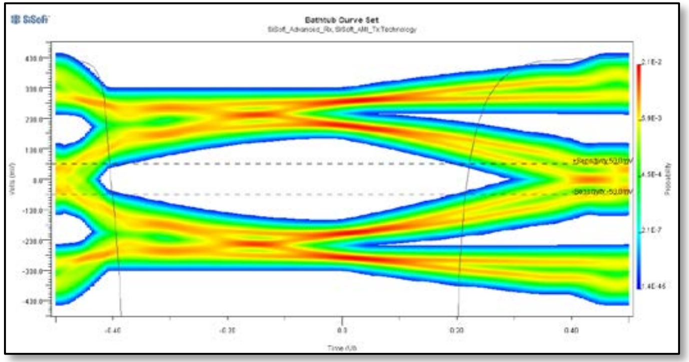


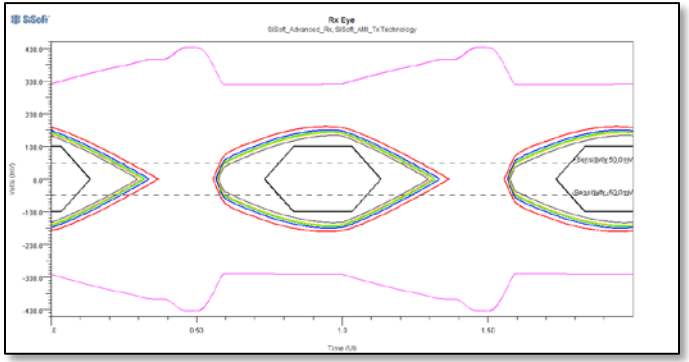
Statistical (red)
Time-Domain (blue)

750,000 bits simulated
500,000 bits ignored
250,000 bits of data

- **Eye diagrams ➔ overall trends**
- **Eye contours ➔ assess how differences affect BER**
- **Remember**
  - Eye contour shifts (right/left) between Statistical & Time-Domain don't matter
  - Jitter tracking can be modeled in Time-Domain simulation but not Statistical
  - Time-Domain eyes/contours will include "drift" behavior but Statistical will not
  - Available probabilities based on analysis type and bits simulated

Interpreting Results, Pass/Fail Analysis





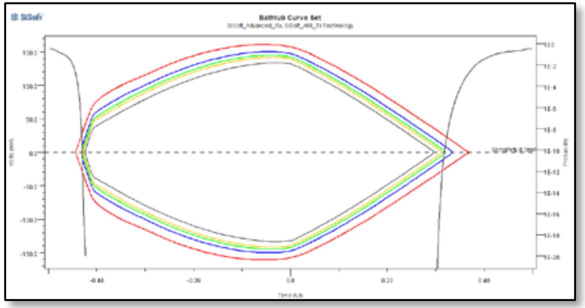
- **EDA tools produce bathtub plots and BER numbers, BUT:**
  - The AMI specification does NOT specify how simulators should process / plot results from model outputs
  - Post-processing / reporting is therefore tool-specific
- **Take time to understand how the Rx vendor expects the outputs to be interpreted:**
  - BER @ sampling threshold?
  - Eye mask @ probability level?

Rx_Receiver_Sensitivity and You

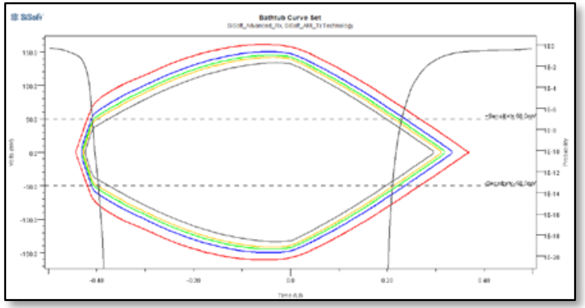
```
(Reserved_Parameters
     (Rx_Receiver_Sensitivity (Usage Info)
        (Type Float)(Range 0.0 0.0 0.1)
        (Description "Rx latch sensitivity.")
     )
```

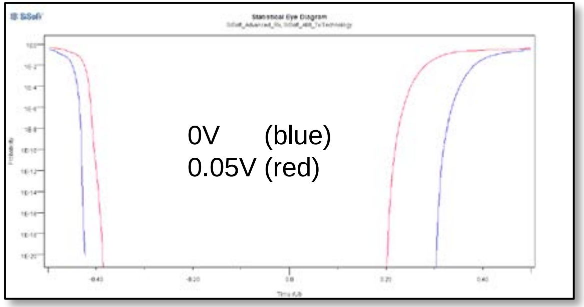
- AMI Reserved Parameter declares input sensitivity at the sampling latch
- Often overlooked (omitted / set to zero)
- Can have big impact on predicted BER

0.0 V



0.05 V

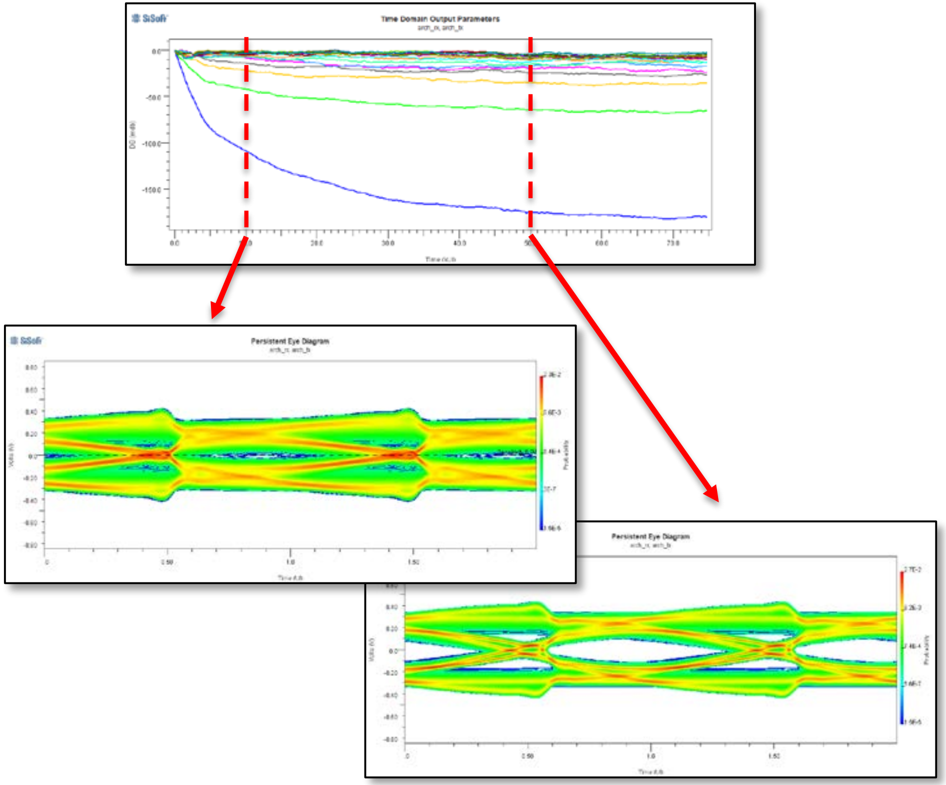




0V (blue)
0.05V (red)

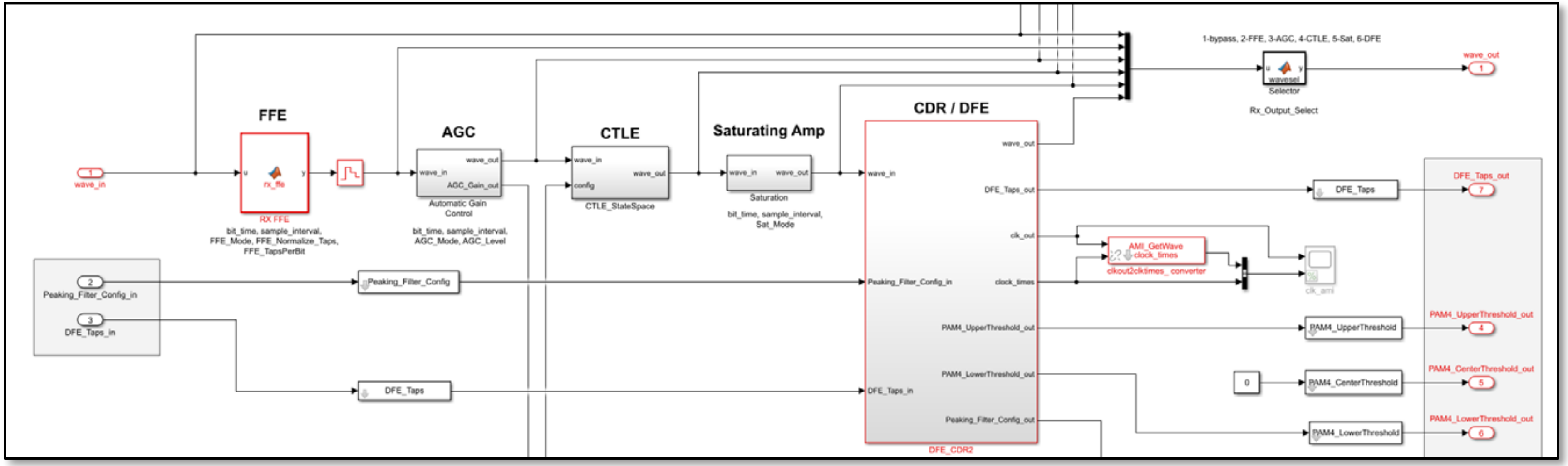
Bathtub curve comparison

Tracking Internal Model States



- **Determine which (if any) AMI parameters your model outputs using the .ami file**
- **Determine how to plot AMI output parameters in your particular simulator**
- **AMI parameter outputs let you**
  - Determine “Ignore_Bits” is set correctly
  - Gain insight into internal model operation
  - Diagnose model stability and performance issues

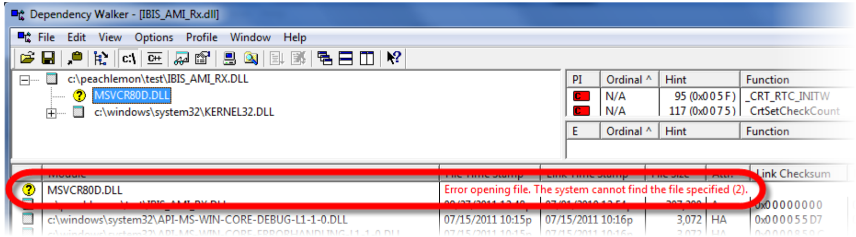
AMI Rx Debugging Techniques



- **Some AMI models can direct internal nodes to the model output**
  - This provides visibility inside the compiled model
  - If the Rx architecture is published and individual blocks can be put in “pass-thru” mode, simulation issues can be isolated/debugged faster

Simulation Crashes / Model Won’t Load

```
ERROR: Failed to load dynamically loadable module IBIS_AMI_Rx.dll
ERROR: Unable to load module. Aborting.
```



Dependency Walker - [IBIS_AMI_Rx.dll]
File Edit View Options Profile Window Help

```
c:\peachlemon\test\IBIS_AMI_RX.DLL
    MSVCR80D.DLL
    c:\windows\system32\KERNEL32.DLL
```

PI	Ordinal ^	Hint	Function
C	N/A	95 (0x005F)	_CRT_RTC_INITW
C	N/A	117 (0x0075)	CrtSetCheckCount

E	Ordinal ^	Hint	Function

Module	File Time Stamp	Link Time Stamp	File Size	Attr.	Link Checksum
MSVCR80D.DLL	Error opening file. The system cannot find the file specified (2).				
c:\windows\system32\API-MS-WIN-CORE-DEBUG-L1-1-0.DLL	07/15/2011 10:15p	07/15/2011 10:16p	3,072	HA	0x000055D7

Dependency Walker

- Algorithmic models are compiled code linked into the simulator at runtime
- Standard O/S runs apply: if required runtime libraries are missing, models won’t run
- AMI models <u>should be</u> self-contained; tools like Dependency Walker help identify issues

UI, Samples/Bit and Channel Model Bandwidth

10 GB/s Example

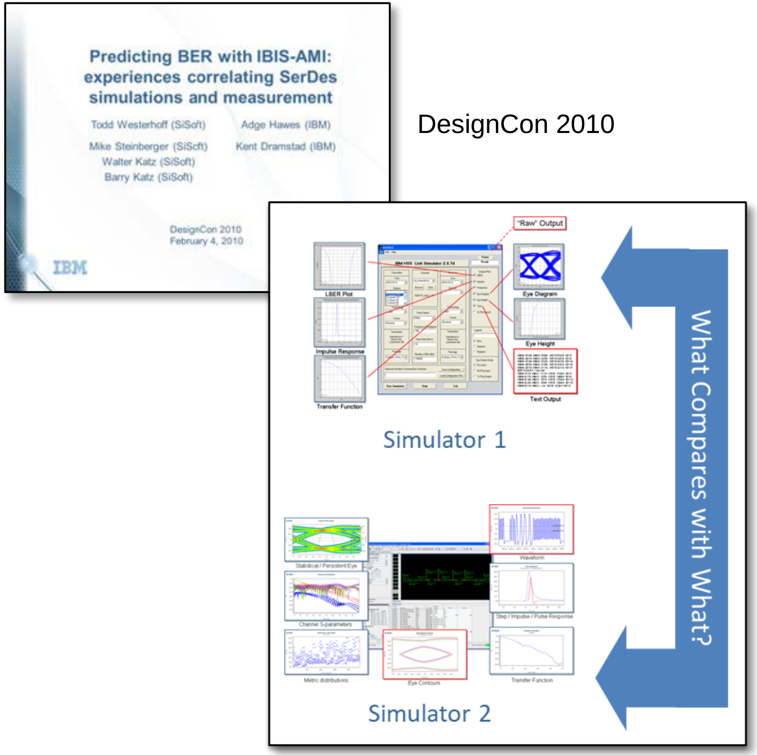
- **UI = 100ps, Samples/Bit = 16**
- **Sample_Interval = 6.25ps**
- **Bandwidth = 160 GHz**

25 GB/s Example

- **UI = 40 ps, Samples/Bit = 32**
- **Sample_Interval = 1.25ps**
- **Bandwidth = 800 GHz**

- **This is as important as it is annoying:**
  - Channel simulation with AMI models is a fixed time-step, DSP-type analysis
  - The channel impulse response and ALL model processing occurs at the same oversampling (samples_per_bit) ration
  - Increasing samples_per_bit to “improve simulation accuracy” increases the channel model bandwidth require to accurately calculate the impulse response

AMI Model Portability



Predicting BER with IBIS-AMI: experiences correlating SerDes simulations and measurement

Todd Westerhoff (SiSoft)
Mike Steinberger (SiSoft)
Walter Katz (SiSoft)
Barry Katz (SiSoft)

Adge Hawes (IBM)
Kent Dramstad (IBM)

DesignCon 2010
February 4, 2010

DesignCon 2010

- **Know your tools**
  - How simulators accumulate / plot data
  - Which plots compare and which don't
- **Build a simple reference example**
  - Start with an example so simple the result can be determined with pencil and paper
  - Add complexity in stages, correlating as you go along
- **"Vanilla" AMI models**
  - Avoid proprietary syntax
  - Test algorithmic models independently
  - Understand the analog model

Questions?

Todd Westerhoff
twesterh@sisoft.com

Mike LaBonte
mlabonte@sisoft.com

Walter Katz
wkatz@sisoft.com