Subject: SerDesDesign.com Channel Simulator

Author: John Baprawski; SerDesDesign.com (JB)

Date: May 2, 2023

For the past 10+ years, John Baprawski has provided cost-efficient high-quality IBIS-AMI models to 40+ high speed digital (HSD) integrated circuit (IC) companies using his IBIS-AMI Model Development Environment for use in any standards compliant SerDes system channel simulator. That work has relied on his free web-based tools including his SerDes Channel Simulator (**https://www.serdesdesign.com/home/**).

This paper highlights some of the features in his channel simulator with links to more detail.

Index

Overview

Be sure to read these two documents first:

READ_ME_FIRST - License_Agreement.pdf

READ_ME_SECOND - Instructions.pdf

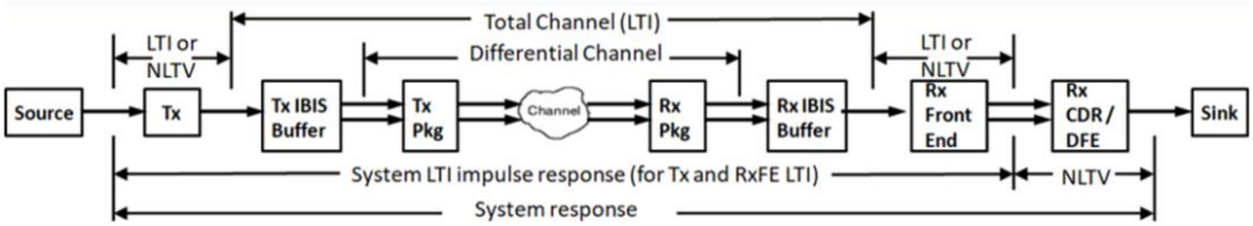
The SerDesign.com simulator is available for use either of these two ways:

- Using the free web-based simulator at: **https://www.serdesdesign.com/home/serdes-system-tool/**
    - Registration on the web site is required to use this free tool.
- Using the same simulator as an installable Windows tool.
    - See separate documentation on qualifying for use of this tool and for delivery and installation on a Windows PC. **SerDes_Design_PremiumAccount**.

Discussion in this paper uses the web-based free tool, but the discussion also applies to the Windows installed tool.

One reference to this tool is the Signal Integrity Journal Article: **Zero Cost SerDes System Channel Simulation | Signal Integrity Journal**

A typical representation of a SerDes system is shown in this block diagram:



This block diagram is generic for any channel simulator, but each tool has its specific way to represent this block diagram.

The constituent parts of this SerDes system include the following.

- TX IBIS-AMI:
    - This model typically contains a Tx equalizer often in the form of a feed-forward equalizer (FFE) with differential IBIS buffer output to the channel.
    - The IBIS buffer can optionally be a 4-port S-parameter file.
- Tx package (Tx Pkg), Channel, Rx package (Rx Pkg):
    - These pieces represent the differential channel.
    - Each part is optional may be represented using S-parameter files.
- RX IBIS-AMI:
    - This model typically contains an Rx continuous time equalizer (CTLE), a clock and data recovery unit (CDR) and decision feedback equalizer (DFE) with differential IBIS buffer input from the channel.
    - The IBIS buffer can optionally be a 4-port S-parameter file.

A key channel simulator property to keep in mind is that it considers that the entire analog content, the ‘Total Channel’ in Figure 1 between the TX AMI portion and the RX AMI portion), is linear and time invariant (LTI). As an LTI system, the entire differential analog section in the SerDes system is accurately represented by its single ended impulse response. This key concept enables the channel simulator to achieve its fast simulation speeds.

If an IBIS-AMI model is LTI, then its property is completely defined by converting its input impulse response into an output impulse response.

If an IBIS-AMI model is not LTI, then it is nonlinear and/or time variant (NLTV). As such its property is defined by converting its input time domain waveform into an output time domain waveform plus and additional signal (called clock_times) that give information to the channel simulator on where the output waveform sampling time instances are.

For the SerDesDesign.com tool, the Total Channel can be represented by either its impulse response or by the constituent parts for the Total Channel as shown in the above block diagram.

There are benefits in representing the Total Channel with its impulse response.

- It can be an impulse response that is exported as a data file from another channel simulator. In this case, the SerDesDesign,com channel simulation results will be the same as it would be in that other channel simulator since the AMI portion of the IBIS-AMI models have a response that if fully deterministic and the same in any channel simulator.
- It can be an impulse response that is exported as a data file from a SerDesDesign.com simulation for which the total channel is represented by its constituent parts.
- Simulating with the Total Channel impulse response would be much faster since the total channel constituent parts would not need to be converted to this impulse response.

Example SerDes Systems

An example SerDesDesign.com SerDes system with NRZ signaling is discussed in this document:

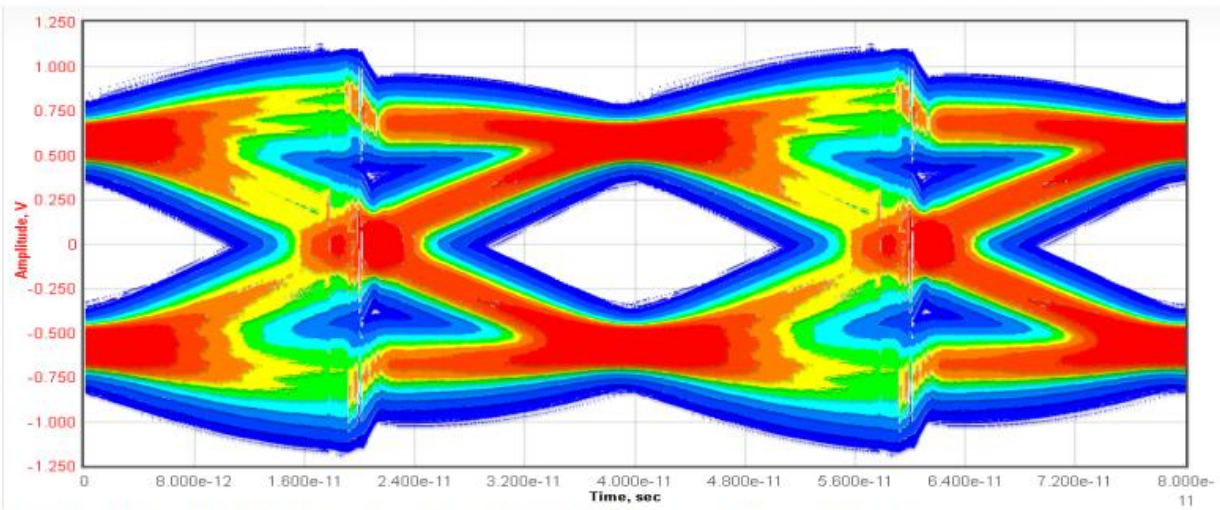
Example NRZ SerDes System Using IBIS-AMI Models (serdesdesign.com

The SerDes system discussed in this document uses these constituent parts:

- Bit rate = 25 Gbps; simulated with 32 samples per bit.
- Tx IBIS-AMI model
    - FFE with filter.
    - Download model here: **SerDes_TxFFE.zip**
- No TxPkg.
- Channel defined with S4P file with 30 dB loss at Nyquist for the differential transmission path.

- No RxPkg.
- Rx IBIS-AMI model
    - CTLE with CDR and DFE.
    - CTLE has 32 states defined with step response waveform data.
        - The CTLE state can be optimized for the given channel impulse.
        - The CTLE output can have auto gain control to specified level.
    - CDR is defined by its OJTS corner frequency.
    - DFE is defined by its nominal output eye peak level and number of taps.
        - The DFE taps can be automatically or manually initialized.
        - The DFE taps can continuously adapt during waveform processing.
    - Download model here: **SerDes_RxCTLE_CDRDFE.zip**

With Bit-by-Bit mode simulating 1,000,000 UIs, the resultant eye density plot (as shown in the above referenced document) is shown here:



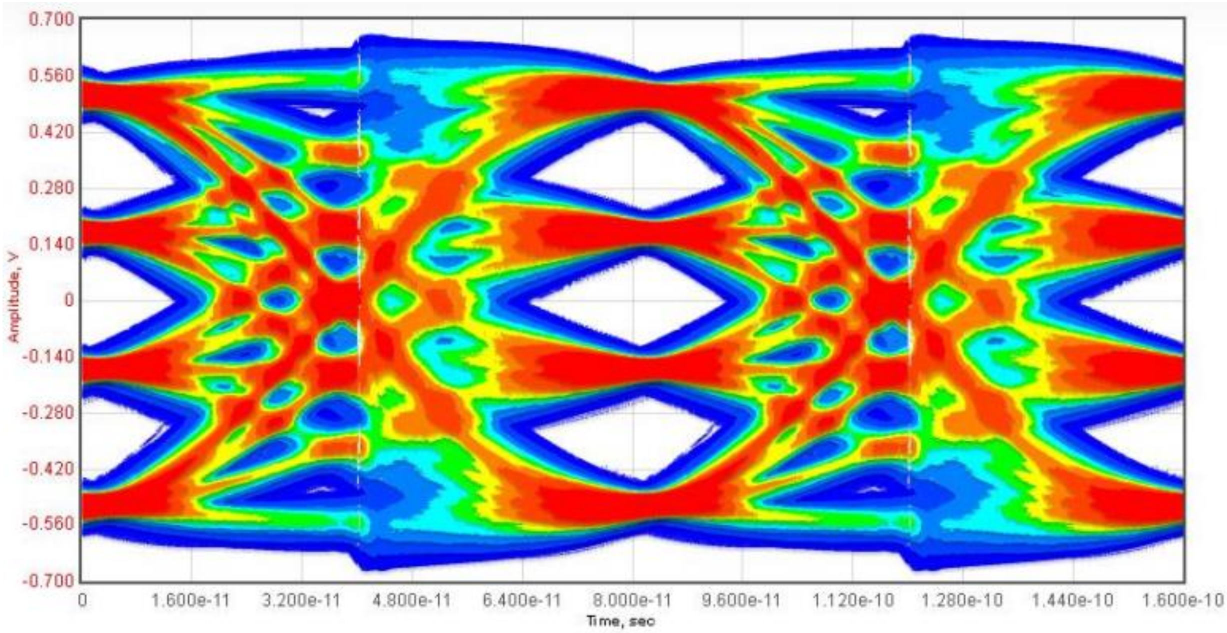
Redish colors are for higher densities. Bluish colors are for lower densities.

An example SerDesDesign.com SerDes system with PAM4 signaling is discussed in this document:

Example PAM4 SerDes System Using Behavioral Models

See this document for discussion of this SerDes system.

With Bit-by-Bit mode simulating 1,000,000 UIs, the resultant eye density plot (as shown in the above referenced document) is shown here:



Redish colors are for higher densities. Bluish colors are for lower densities.

The SerDesDesign.com Channel Simulator has many features that can be observed, investigated, and used on the SerDesDesign.com web site. This includes SerDes repeater systems and SerDes electrical-optical-electrical repeater systems.

The following topics are just a select few that highlight some of the development processes available.

How to accurately convert S-parameters to impulse responses

SerDes channels are typically defined with S-parameters which are in the frequency domain. For use in a SerDes system channel simulator, the S-parameters must be converted to their time domain equivalent impulse responses.

Many channel simulators in the industry make these conversions with a compromise in the S-parameter fidelity by introducing excessive filtering and/or including excessive high-frequency aliasing.

SerDesDesign.com have made S-parameter to time domain impulse conversions with no excessive filtering or any high frequency aliasing and with accurate reproduction of the S-parameters frequency characteristics up to the maximum frequency in the S-parameters.

See these references:

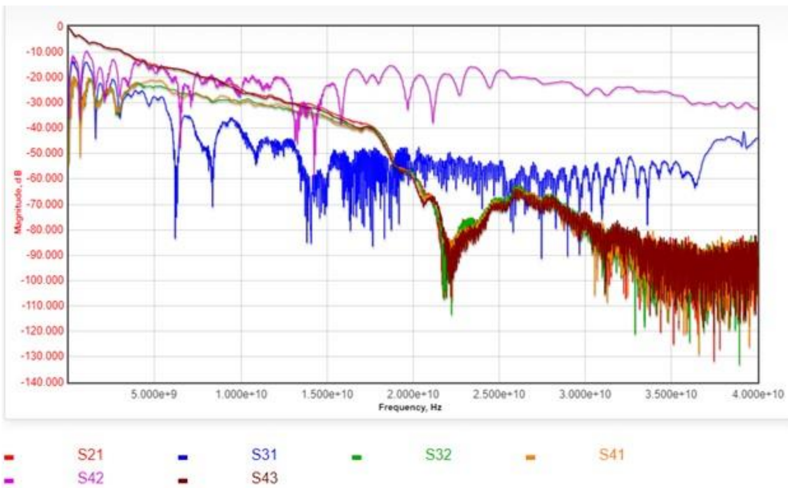
[Overcoming Signal Integrity Channel Modeling Issues]

A Better Anti-Aliasing Process

Consider a SerDes system with NRZ signaling using 25 Gbps at 32 samples per bit.

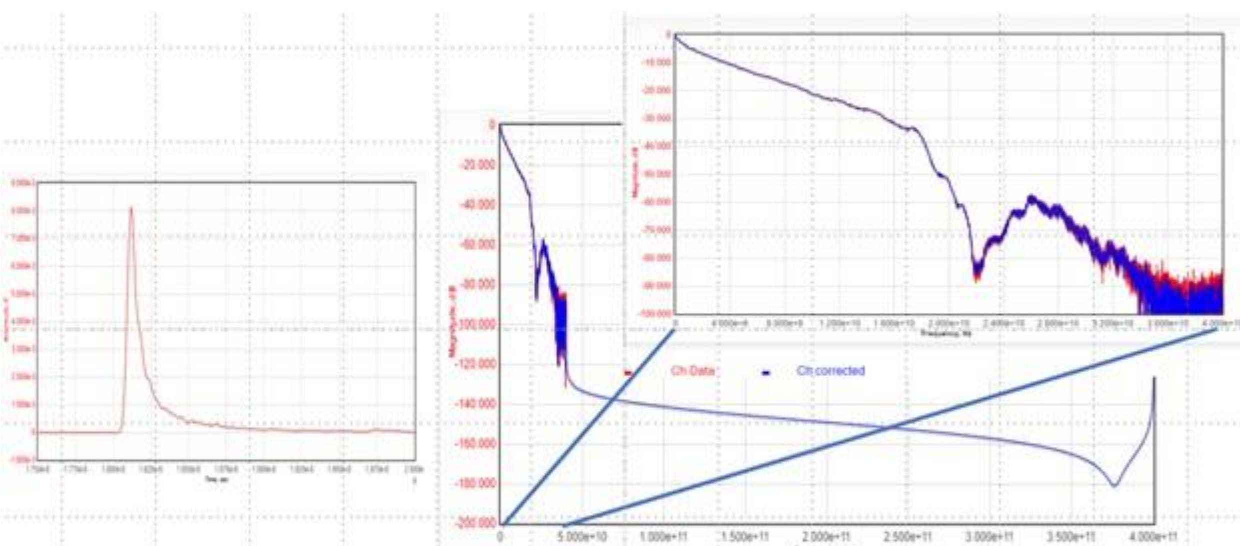
Also, consider the channel composed of one S4P file with differential inputs/outputs interfacing the Tx IBIS buffer differential outputs and Rx IBIS buffer differential inputs.

This figure shows the S4P data transmission magnitudes for Sij, i<j. The S4P data highest frequency is 40 GHz.



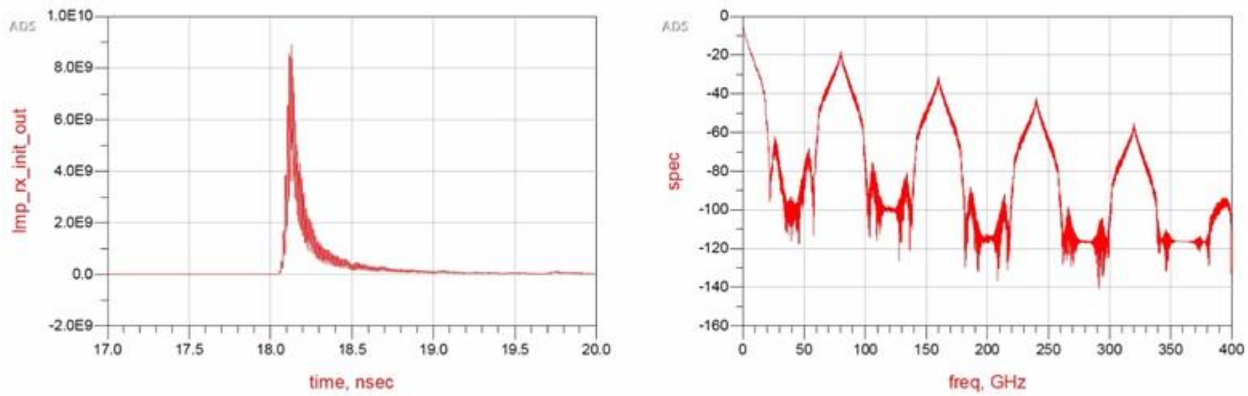
At Nyquist (12.5 GHz), the S21 and S43 characteristics have a loss of about 25 dB. Also observe the suck-out in the responses at about 22 GHz.

When this SerDes system with S4P channel is used in the SerDesDesign.com channel simulator, the resultant impulse response is shown in this figure (time domain, left; frequency domain, right).



Observe that this SerDesDesign.com channel time domain impulse is clean and smooth with no ringing. The frequency domain view shows smooth extrapolation beyond the S-parameter upper frequency of 40 GHz. In the frequency domain view, observe that the hdd21 response (blue) matches the sdd21 response (red) up to 40 GHz with reduction of the noise inherent in the S-parameters as the frequency approaches 40 GHz.

When this same system is used in an alternate channel simulator, the resultant impulse response is shown in this figure (time domain, left; frequency domain, right).



There is clear high-frequency aliasing in the frequency to time domain conversion process.

In the two articles referenced above, processes are described to:

- Convert S-parameters to what SerDesDesign.com calls Causal S-parameter which have no high-frequency aliasing.
- Convert any impulse response from another channel simulator to one that has no additional filtering and eliminates all high frequency aliasing.

Using processes from SerDesDesign.com (including in their channel simulators), any S-parameters can be accurately converted into accurate impulse responses with no high-frequency aliasing.

How to create IBIS buffer models from just the channel facing IBIS differential reflection coefficients

Tx and Rx IBIS buffers can be defined using 4-port S-parameters. Unfortunately, many SerDes circuit designs cannot provide an easy way to characterize these 4-port S-parameters. However, every circuit design can be evaluated for their output (Tx) or input (Rx) differential reflection coefficients versus frequency (S2P data).

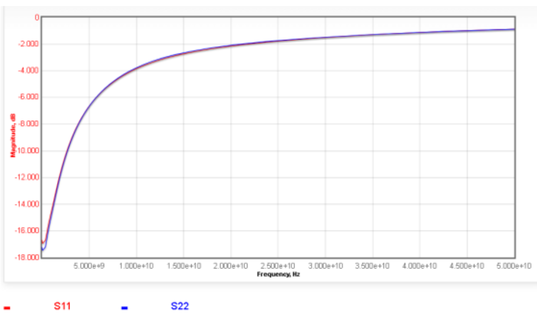
SerDesDesign.com provides a way to convert the S2P data to S4P data for use as the IBIS buffer definition.

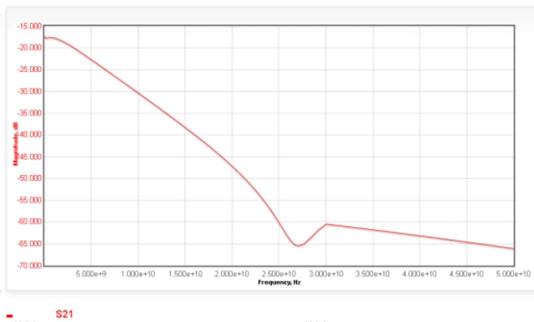
See detail in these SerDesDesign.com documents.

Tx_IBIS_based_on_S4P_data (serdesdesign.com); see section Defining the Tx IBIS S4P using S2P data.

Rx_IBIS_based_on_S4P_data (serdesdesign.com); see section Defining the Rx IBIS S4P using S2P data.

As an example, consider a SerDes NRZ system operating at 17.17 Gbps using a Tx IBIS buffer with differential output resistance represented with this S2P data out to 50 GHz in this file: TX_IBIS_Buffer_Example.s2p. The S11 and S22 characteristics are shown here on the left. The S21 and S12 characteristics are the same and shown on the right.





The SerDesDesign.com tool used was: **View SParameters Tool**

To convert this S2P data to Tx IBIS buffer S4P data, use it in the SerDesDesign.com channel simulator with ChannelType = 3:

Name	Description	Entry Value(s)	Status	Type	Limits	Comment	Action
ChannelType	Channel specification type	3		Integer	[0, 3]	0 = None 1 = Channel implulse data 2 = Channel S-parameter data 3 = Channel + Tx/Rx IBIS/pkg	Open...

With the Tx IBIS buffer defined as shown here:

Channel - 3. Tx Buffer/Pkg + Channel + Rx Pkg/Buffer

Name	Description	Entry Value(s)	Status	Type	Limits	Comment
EnableTxBuffer	Enable transmit IBIS buffer	3	✔	Integer	[0, 3]	0 = No 1 = Yes 2 = Use IBIS 3 = Use Alt IBIS
EnableTxTs4Corners	Enable corners for the Ts4file	0	✔	Integer	[0, 1]	0 = No 1 = Yes
Ts4File	Alternate IBIS model using S-parameter file. Alpha-numeric or underbar; start with alpha	TX_IBIS_Buffer_Example.s2p Choose File No file chosen	✔	File		Upload a file (Touchstone 1.0 format) or list previously uploaded file

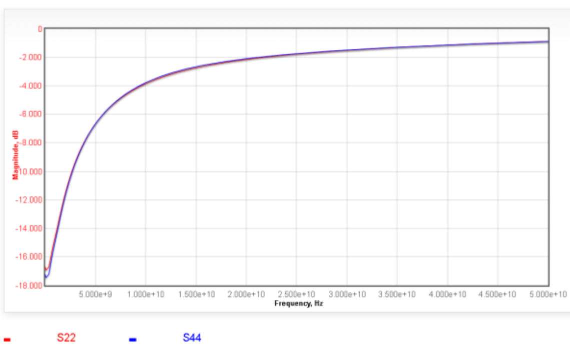
When the simulation is run, the S4P file will be generated as seen here in the simulator message panel and is named: TX_IBIS_Buffer_Example.s2p.s4p.

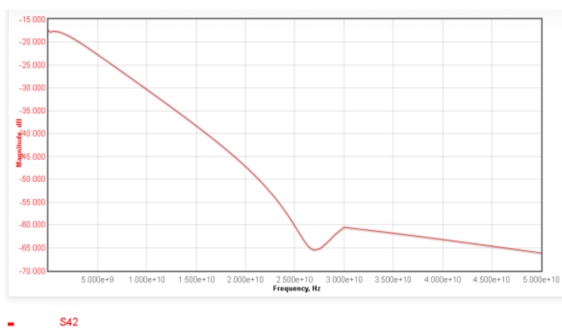
```
Tx IBIS buffer will be defined using the alternate AMI IBIS buffer definition with S-parameter file TX_IBIS_Buffer_Example.s2p.
Tx IBIS buffer Tx_VoltageRange = 1000
Reading S-Parameter file TX_IBIS_Buffer_Example.s2p.
S2P file converted to S4P file with filename = TX_IBIS_Buffer_Example.s2p.s4p.
Use of this S4P file as the TX IBIS buffer requires setting Tx_V = 1000 and Tx_R = 50.
Since S2P is used, the Tx_V is set to 1000 Volts and Tx_R is set to 50 Ohms.
For TxPkg S-parameter file, fMin = 1, fMax = 5e+10, fStep = 2e+06, numFreq = 25001
```

View these S4P parameters using this tool: **[View SParameters Tool]**

For the Tx IBIS buffer use, ports 2 and 4 are the differential output pair.

The S22 and S44 characteristics are shown here on the left. The S24 and S42 characteristics are the same and shown on the right.





As can be seen, the S4P S22, S44 and S42 characteristics are the same as the S2P S11, S22 and S21 characteristics.

This S4P file can be downloaded from the View SParameter Tool web page.

This discussion focused on using an S2P file to define the Tx IBIS buffer. A similar process can be followed to use an S2P file to define the Rx IBIS buffer.

How to model CTLEs using frequency domain data, time domain data or pole-zero representations

Continuous Time Linear Equalizers (CTLEs) are typically used in Rx circuits but may also be used in Tx circuits. As is done with SerDesDesign.com, CTLEs can often be modeled with a combination of linear and nonlinear characteristics. In this discussion, the focus is on the linear portion (called linear and time invariant, LTI).

As is done with SwerDesDesign.com, this LTI model can be modeled one of four ways:

- With response peaking factors; in the frequency or time domain.
- With poles and zeros; with one section.
- With time domain waveforms; with up to four cascaded sections.
- With frequency domain spectrums; with one section

See this SerDesDesign.com dialog box for a one section LTI Rx CTLE:

Receiver Front End - 1. CTLE/AGC - 1 section

Name	Description	Entry Value(s)	Status	Type	Limits	Comment
InputGain	Additional input scalar gain	1	✓	Real		
EnableModelType	Model type	0	✓	Integer	[0,4]	0=No equalization 1=List of peaking dBs 2=File with set of poles and zeros 3=File with set of step responses 4=File with set of spectrums
EnableAutoGainControl	Auto adapt output gain during initialization to achieve the AGC_Level	1	✓	Integer	[0,1]	0 = No 1 = Yes
AGC_Level	Target eye sampling instant peak to peak level	1.0	✓	Real	>0	
EnableRxJitter	Enable Rx jitter parameters	0	✓	Integer	[0,2]	0 = No 1 = Yes in sec 2 = Yes in UI units
ChAntiAliasingFc	Channel impulse anti-aliasing filter corner frequency in Hz	0	✓	Real	>=0	0 = Not used Used when >= SymbolRate

As can be seen, the type of CTLE is controlled by the EnableModelType parameter.

Detail discussion of this model is posted on the web at this link: **RxFE1 model**.

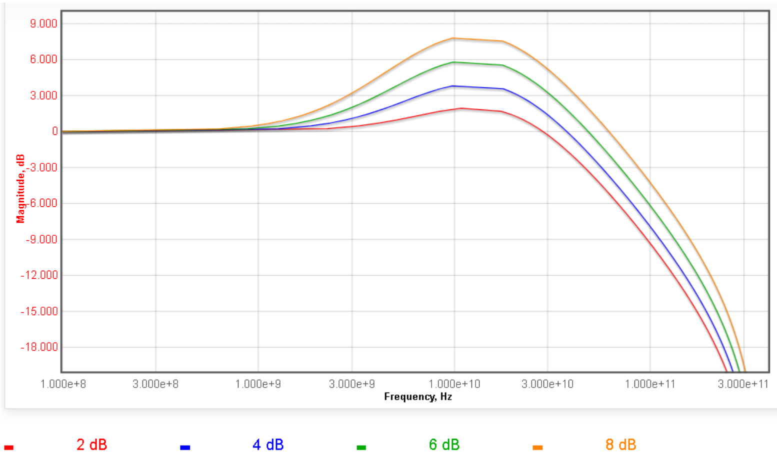
This discussion is on a summary overview of these four model types.

In all cases, the NRZ signaling bit rate is 25 Gbps and 32 samples per bit. When multiple specs are listed, the model can automatically select the one that gives the best eye opening for a given channel.

CTLE defined with Peaking Factors

For this model type, the peaking factor can be specified in the time or frequency domain.

For the frequency domain, consider peaking at 12.5 GHz (Nyquist for 25 Gbps) with peaking factors of 2, 4, 6 and 8 dB. Here is the resultant frequency domain characteristic:



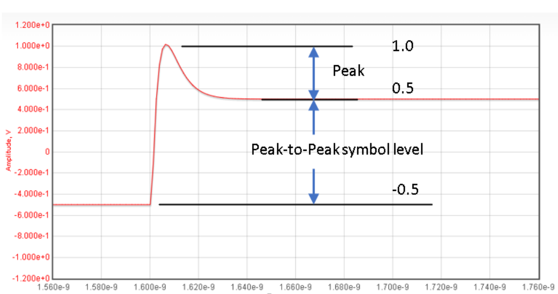
For the time domain, observe these peaking control parameters:

Data1Index	Data set index; used when EnableAdaptForChannel=0	4	✓	Integer	[1,RxFE1_NumData1]	Used when AdaptForChannel=0
NumData1	Number of data sets in the data	4	✓	Integer	>0	
EnabledBorV_Peak	Use peaking dB or Voltage	1	✓	Integer	[0,1]	0=FPeak 1=vPeak
vWidth	Peaking voltage width	0.3	✓	Real	[0,0.4]	
vWidthMode	Peaking voltage width mode	0	✓	Integer	[0,1]	0=Start at mid peak level 1=Start at symbol start
vPeakList	Amplitude peaking ratio relative to peak-to-peak symbol level	0.2, 0.3, 0.4, 0.5	✓	Real array	[0, 0.65]	

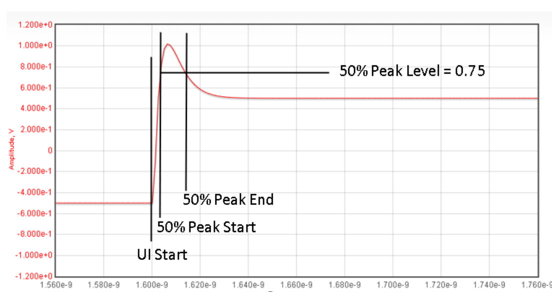
Observe that the EnabledBorV_Peak is set to 1 to enable time domain peaking.

Observe that the vPeakList has 4 entries and NumData1 is also set to 4. With automatic optimization disabled, the Data1Index=4 is used which selects the 0.5 factor in the vPeakList.

The meaning of vWidth, vWidthMode and vPeak is shown in these figures:



vPeakList entry = Peak / (Peak-to-Peak symbol level) = (1.0 – 0.5)/(0.5 – (-0.5)) = 0.5



For vWidthMode = 0; vWidth = (50% Peak End – 50% Peak Start) in UI
For vWidthMode = 1; vWidth = (50% Peak End – UI Start) in UI

In these figures, 1 UI = 40 psec. The x-axis has 10 divisions. 1 UI = 2 divisions.

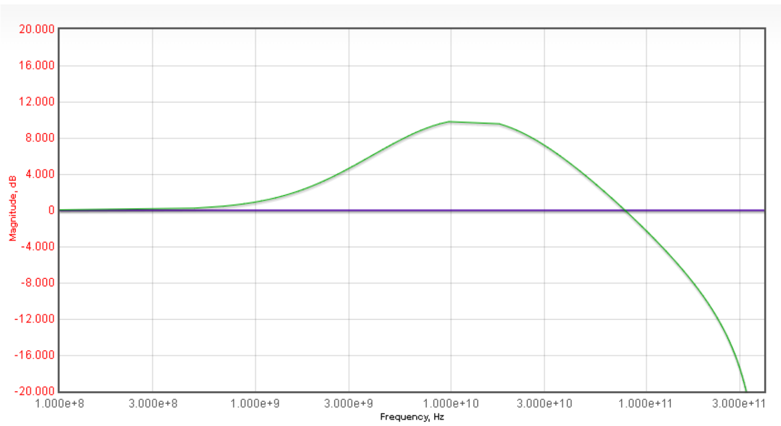
CTLE defined with Poles and Zeros

For this model type, the CTLE response is defined with a list of real and complex poles and zeros contained in a file.

The Pole-Zero file format is at this link: **Pole_Zero_File_Format**

Within the example listed in this file, with 10 dB peaking at 12.5 Ghz, the transfer function has a zero at 2.0 GHz, a pole at 10.31 GHz, another pole at 15.94 GHz and DC gain of 1.0.

This results in the frequency domain plot shown here:



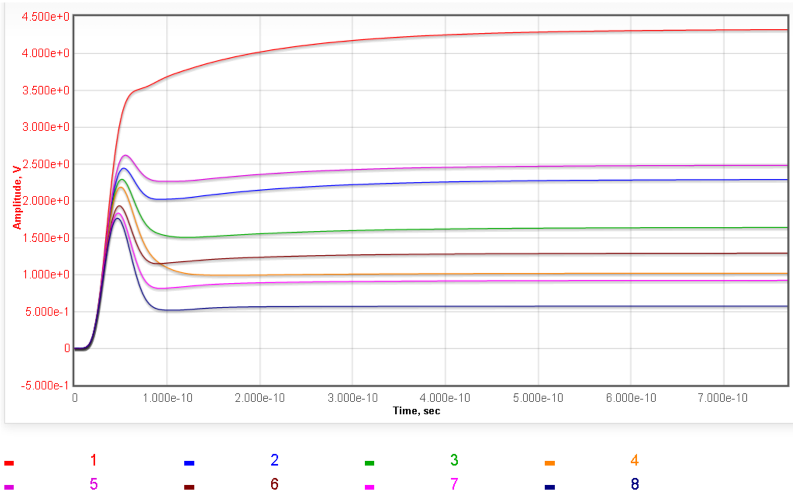
If the file has multiple pole-zero lines, then the CTLE model can automatically select the optimal one to give the best eye opening for the given channel.

CTLE defined with Time Domain Waveforms

For this model type, the CTLE response is defined with a list of time domain step responses contained in a file.

The Step Response file format is at this link: **Step_Response_File_Format**

An example file (RxCTLE_Data.csv) is used with 32 step responses for which the first 8 waveforms are shown here:



If the file has multiple step response sets, then the CTLE model can automatically select the optimal one to give the best eye opening for the given channel.

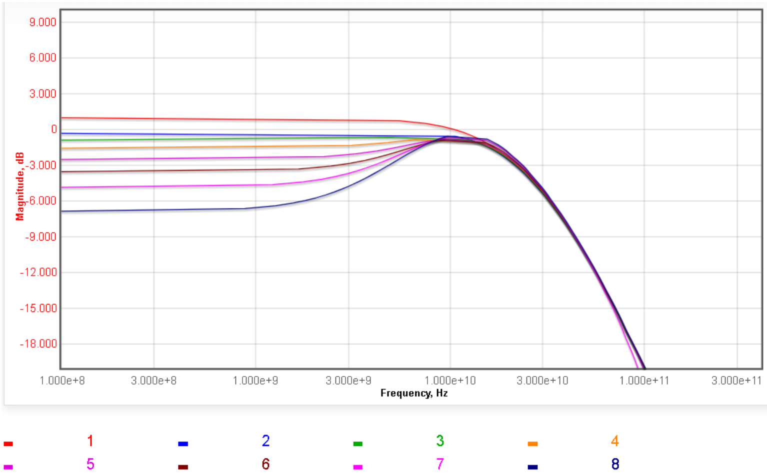
For this model type, one has the option to define a data file for each of three corner cases: typical, slow and fast.

<u>CTLE defined with Frequency Domain Spectrums</u>

For this model type, the CTLE response is defined with a list of frequency domain responses contained in a file.

For N responses, there are 1+2*N columns. The first column is for the common frequency for all sets. Each spectrum is defined with two columns, one for magnitude in dB, the second for phase in degrees.

An example file, RxCTLE_Data_Spectrum.csv, is used with 8 frequency domain responses for which the spectrum magnitudes in dB are shown here:



If the file has multiple spectrum sets, then the CTLE model can automatically select the optimal one to give the best eye opening for the given channel.

For this model type, one has the option to define a data file for each of three corner cases: typical, slow and fast.

How to model FFEs using waveform data

Feedforward Equalizers (FFEs) typically are used in Tx circuits but may also be used in Rx circuits. As is done with SerDesDesign.com, FFE can often be modeled with an explicit list of tap values, but can also be modeled using a set of time domain waveform data.

When modeled using time domain waveforms, SerDesDesign.com calls this the FFE Black Box Model. There is one pre-cursor, the main tap, one post-cursor and there may be multiple drive levels.

Detail discussion of this model is posted on the web at this link: **TxBB model**.

An actual use case by NXP is posted in this presentation made at the Keysight 2020 Tokyo Forum: **Keysight_2020_TokyoForum**; see section 4: Modeling a SerDes Tx IC with measured data.

This NXP model was used from 1.0 – 28.05 Gbps. The model had many states:

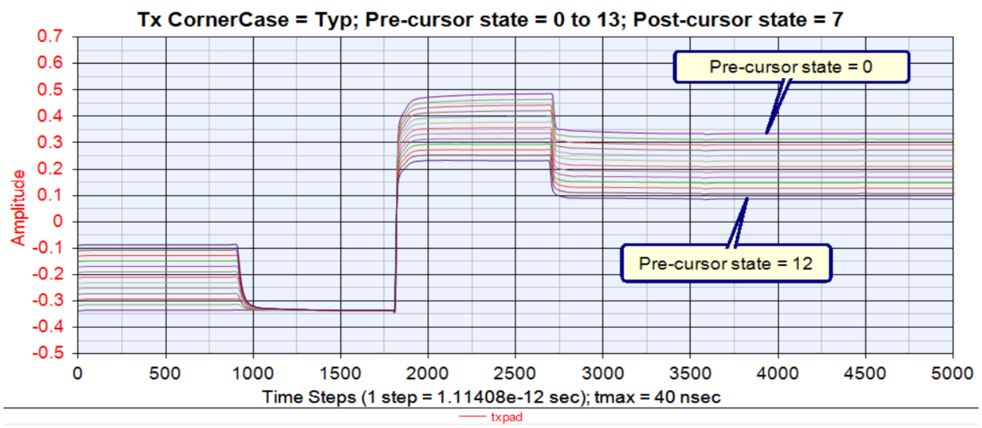
- Three corner cases: Typ, Slow, Fast
- Swing level with 14 states
- Pre-cursor with 25 states (12 negative, 0, 12 positive)
- Post-cursor with 33 states (16 negative, 0, 16 positive).
- In total, 14*25*33*3 = 34,650 states

The SerDesDesign approach uses a subset of these states for characterization, but still supports the full set of 34,650 states.

Waveforms were recorded while varying the Drive level control parameter over its full defined 14 states for each corner case. This was done with zero pre-cursor and zero post-cursor applied. For the 3 corner cases there are 14 * 3 = 42 waveform simulations (files).

Waveforms were recorded at the maximum Drive level while varying the pre-cursor over its 13 states (zero and negative) and the post-cursor over its 17 states (zero and negative) for each corner case. For the 3 corner cases, there are 13 * 17 * 3 = 663 waveform simulations (files).

Part of the characterization data was collected for one setting of post-cursor and a sweep of all pre-cursor states for the Typical corner and one Drive level.



For this model type, one has the option to define a data file for each of three corner cases: typical, slow and fast.

How to Generate IBIS-AMI Models from Behavioral Characteristics

SerDesDesign.com provides many different types of behavioral models for Tx and Rx SerDes circuits. The Tx or Rx circuit can be defined behaviorally for its IBIS buffer characteristic and behaviorally for it equalization (AMI) portion.

For example, the IBIS buffer can be defined with an IBIS circuit or with a S4P file per the IBIS specification.

See these SerDesDesign.com documents for Tx IBIS buffer modeling:

- **Simplified_Tx_IBIS_Buffer_used_in_SerDes_Simulations**
- **Alternate_Tx_IBIS_Buffer_used_in_SerDes_Simulations**

See these SerDesDesign.com documents for Rx IBIS buffer modeling:

- **Simplified_Rx_IBIS_Buffer_used_in_SerDes_Simulations**
- **Alternate_Rx_IBIS_Buffer_used_in_SerDes_Simulations**

The SerDesDesign.com Tx LTI AMI behavioral models can be either of these:

Transmitter (Tx) Feedforward Equalizer (FFE) based on tap values
Tx FFE model based on waveform data
Tx FFE model based on digital registers
Tx Continuous Time Linear Equalizer (CTLE)

The SerDesDesign.com Rx LTI AMI behavioral models can be either of these:

Receiver (Rx) CTLE – 1 stage
Rx CTLE – 2 stages
Rx CTLE – 3 stages
CTLE – 4 stages
Rx FFE based on tap values
Rx FFE Black Box based on waveform data

The SerDesDesign.com Rx NLTV AMI Nonlinear behavioral models can be either of these:

Rx Nonlinearity based on equations
Rx Nonlinearity based on files
Rx Nonlinearity based on files including compensation factors

The SerDesDesign.com Rx NLTV AMI CDR/DFE behavioral models can be either of these:

Rx Clock and Data Recovery model (CDR)
Rx CDR and Decision Feedback Equalizer (DFE)

The total Rx NLTV behavioral AMI model can be a combination of the above two types of NLTV behavioral models.

The total Rx behavioral AMI model is the combination of the total Rx LTI behavioral model and the total Rx NLTV behavioral model.

Having set up the Tx and/or Rx behavioral models as desired, the SerDes system channel can be run. After one has a successful channel simulation with results as desired, then one can enable the SerDesDesign.com feature to generate the Tx or Rx IBIS-AMI model files.

To do this enable the Setup Analysis parameter GenerateModels. For example, to generate the Rx IBIS-AMI model, one would set GenerateModels = 2 and give a name to the RxIBIS_AMI_ModelName. In this example the ModelName = Rx.

GenerateModels	Generate IBIS-AMI models Generate models after Run with satisfactory results	2		Integer	[0, 1, 2]	0 = No 1 = Yes: Tx only 2 = Yes: Rx only
TxIBIS_AMI_ModelName	Transmitter IBIS-AMI model name	Tx		ModelName		Used when GenerateModels=1
RxIBIS_AMI_ModelName	Receiver IBIS-AMI model name	Rx		ModelName		Used when GenerateModels=2

Now, when the simulation is run, the IBIS-AMI models are generated in a folder in the user's model data directory. See the typical message shown here:

> You have requested that a Rx IBIS-AMI model be generated.
>
> The data files associated with this successful analysis have been saved with reference number:
> user-<user_number>_<assigned_reference_number>

The generated files include the IBIS-AMI model *.ami, *.ibs, *.h, *.cpp files. The *.cpp files need to be converted to a Windows dll and/or Linux *.so file to complete the IBIS-AMI model deliverable.

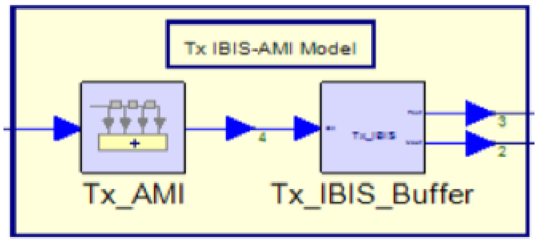
To achieve the Windows dll and/or Linux so files, one can use one of these two techniques.

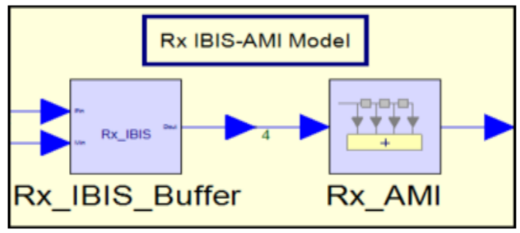
1. Use the SerDesDesign.com IBIS-AMI Model Generation product with the SerDesDesign.com IBIS-AMI Model Development Environment product.
   - See the Model Generation product documentation:
     - IBIS_AMI_Model_Generation.pdf
   - See the Model Development Environment product documentation:
     - IBIS_AMI_Model_Development_Environment.pdf
   - With this approach one can generate the *.dll and *.so files using these tools by following their instructions.
   - On Windows, the free Windows Visual Studio tool defined in IBIS_AMI_Model_Development_Environment.pdf needs to be obtained and installed.
   - On Linux, the free CMake tool defined in IBIS_AMI_Model_Development_Environment.pdf needs to be obtained and installed.
   - The generated *.dll and *.so files inherit the same time-based node-locked license as in the Model Generation product and are usable in any Channel Simulator on the same user Windows PC.
   - To remove the license, follow the instructions in IBIS_AMI_Model_Generation.pdf.

2. Send a zipped copy of the user-<user_number>_<assigned_reference_number> directory to SerDesDesign.com with ordering instructions defined at this link:
   - **Ordering_IBIS_AMI_Models**
   - This option does not require purchase of the tools in option 1 above.
   - If you had purchased SerDesDesign.com IBIS-AMI Model Generation and Model Development Environment products, your deliverable from SerDesDesign.com will include the buildable project with source code.
   - Otherwise, your deliverable from SerDesDesign.com will not include the buildable project or source code, but it will include the Windows dll and/or Linux so files as ordered.

How to improve CTLE/FFE models by de-embedding IBIS buffer characteristics from their waveform data

The Tx IBIS-AMI model (left, single ended input, differential output) and the Rx IBIS-AMI model (right, differential input, single ended output) are represented in these figures:





For discussion purpose, assume that the Tx/Rx AMI are LTI. When they are NLTV, then assume that the circuit has a measurement point between the LTI and NLTV portions and that the AMI model can thus be portioned into an LTI portion and an NLTV portion with the LTI portion interfacing to the IBIS portion.

Thus, for this discussion, assume the AMI is LTI (since the NLTV portion is characterized separately).

In many cases, the Tx/Rx circuit does not have a convenient measurement point between the AMI and IBIS portions of the circuit.

The combined AMI+IBIS are characterized together. To get the AMI portion, and IBIS model needs to be created and de-embedded from the AMI+IBIS data.

SerDesDesign.com supports this process.

The IBIS portion can be defined based on its Tx output/ Rx input differential reflection characteristic as discussed in the prior section: How to create IBIS buffer models from just the channel facing IBIS differential reflection coefficients.

The AMI+IBIS data can be collected as time domain waveform data as discussed in the prior sections: How to achieve model CTLEs using time domain data; How to achieve FFE models using waveform data.

For discussion purposes, let's work with an Rx circuit that has measured IBIS input reflection coefficient data (TX_IBIS_Buffer_Example.s2p) and Rx circuit IBIS+LTI CTLE waveform data (RxCTLE_Data.csv) using NRZ signaling at 25 Gbps with 32 samples per symbol.

For this example, in the SerDesDesign.com channel simulator, use this set up.

TransmitterType = 0; no transmitter used.

ChannelType = 3:

Set EnableTxBuffer = 1; with default ideal 50 ohms per differential pin.

EnableTxPkg = 0; this is by default

EnableChannel = 1; this is by default with the default S4P

EnableRxPkg = 0; this is by default

For the Rx IBIS buffer, use these setting:

EnableRxBuffer	Enable receive IBIS buffer	3	✔
EnableRxTs4Corners	Enable corners for the Ts4file	0	✔
Ts4File	Alternate IBIS model using S-parameter file. Alpha-numeric or underbar; start with alpha	TX_IBIS_Buffer_Example.s2p Choose File No file chosen	✔
R	Alternate IBIS model load resistance per pin	1.0e9	✔
Ts4filePinOption	Alternate IBIS S-parameter file pin pattern	0	✔
DeembedIBIS	Deembed Rx IBIS response from Rx AMI LTI model	1	✔
BufRl	Rx IBIS input load per pin for deembedding	50	✔

The IBIS S2P file will be converted to its S4P equivalent: Rx_IBIS_Buffer_Example.s2p.s4p.

DeembedIBIS = 1 so that the IBIS buffer characteristic is de-embedded from the Rx CTLE data.

BufRl is the Rx IBIS input load per pin for use during Rx CTLE de-embedding.

During simulation, the Rx_IBIS_Buffer_Example.s2p.s4p characteristics will be de-embedded from the Rx LTI AMI behavioral model (one that allows it).

ReceiverFEType = 1 with the following settings:

EnableModelType = 3; to allow CTLE using a set of time domain waveforms.

EnableAdaptForChannel = 1; to enable automatic selection of the optimal CTLE state.

NumData1 = 32; since the data file to be used will have 32 states

ColumnArrangement1 = 1; since the data file to be used will have format t, v1, v2, …

Data1File = RxCTLE_Data.csv; the data file to be used

ReceiverNLType = 1; Nonlinearity set to off by default; this just forces bit-by-bit simulation

ReceiverNLTVType = 0; No CDR or DFE

In the Setup Analysis section, make this change.

In the SetupBitByBitMode dialog box, set Analysis Symbols = 1000000 (1M UIs)

Then Run Analysis. The analysis message window displays the following:

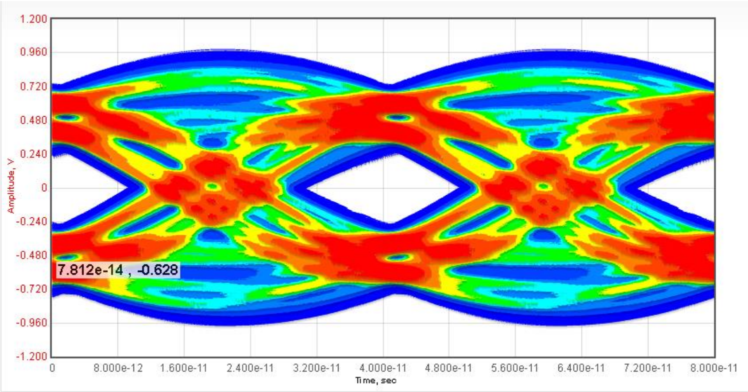
Rx IBIS buffer will be defined using the alternate AMI IBIS buffer definition with S-parameter file TX_IBIS_Buffer_Example.s2p.
Reading S-Parameter file TX_IBIS_Buffer_Example.s2p.
S2P file converted to S4P file with filename = TX_IBIS_Buffer_Example.s2p.s4p.
Use of this S4P file as the RX IBIS buffer requires setting Rx_R = 1.0e9.
Since S2P is used, the Rx_R is set to 1.0e9 Ohms.

The above message shows that the S2P IBIS buffer was converted to its equivalent S4P.

Running SerDesDesign_RxFE1
Deembedding IBIS buffer from Rx data will begin.
Completed deembedding IBIS buffer from Rx data.
Front end (FE) model based on step response data.
Optimization of EQ over count = 32 states resulting in max open eye ratio = 0.117335 with optimized states:
Data1Index = 17
To adapt channel+equalizer loss for peak to peak eye level (1), a gain of 11.5333 dB is added at the equalizer output.

The above shows that the data in the original Rx CTLE data file RxCTLE_Data.csv had the Rx IBIS buffer characteristic de-embedded resulting in optimization with Data1Index = 17.

The resultant eye density plot is shown here:



This all worked out fine, but the de-embedded data was not stored as a separate file.

If one would like to recover the de-embedded CTLE data as a separate file, then additional steps are required. The Rx IBIS-AMI model generation must be enabled as was discussed in the prior section.

Set GenerateModels = 2 and rerun the simulation.

The messages generated for the Rx CTLE are shown here:

Running SerDesDesign_RxFE1
Deembedding IBIS buffer from Rx data will begin.

Generate IBIS-AMI model started.

Create C:\SerDesDesign\user-37\files\model_data\user-37_1675485549_AMI_Solution directory.
Create source directory.
Create AMI-Rx directory.
Completed deembedding IBIS buffer from Rx data.
Front end (FE) model based on step response data.
Optimization of EQ over count = 32 states resulting in max open eye ratio = 0.117335 with optimized states:
Data1Index = 17
To adapt channel+equalizer loss for peak to peak eye level (1), a gain of 11.5333 dB is added at the equalizer output.
Create Configure-for-Linux-Makefiles.sh file.
Create Configure-for-win64-vs2015.bat file.
Create source\CMakeLists.txt file.
Create AMI-Rx\CMakeLists.txt file.
Create Rx.ibs file.
Create Rx.ami file.

Create Rx_AMI.h file.
Create Rx_AMI.cpp file.
Success in generating custom IBIS-AMI model files.

The above shows the success in de-embedding and in model generation.

For the web based SerDesDesign.com channel simulator, one must order the generated IBIS-AMI model with payment required. The de-embedded CTLE data file will be named RxFE_Data1.csv and the regenerated original CTLE data file will be named RxFE_Data1_Original.csv

For the personal Windows based SerDesDesign.com channel simulator, the resultant de-embedded CTLE data is located here:

C:\SerDesDesign\user-<user_number>\files\model_data\user-<user_number>_<reference_number>_AMI_Solution\source\AMI-Rx\RxFE_Data1.csv

The original unmodified CTLE data was regenerated and is located here:

C:\SerDesDesign\user-<user_number>\files\model_data\user-<user_number>_<reference_number>_AMI_Solution\source\AMI-Rx\RxFE_Data1_Original.csv

This contrast shows the benefit of becoming a Premium Account member.

Terms and Conditions

See terms and conditions for IBIS-AMI Modeling are at this link: **Terms & Conditions | Privacy Policy**