DesignCon 2016

Building IBIS-AMI Models from Datasheet Specifications

Eugene Lim, Intel Corporation

Donald Telian, SiGuys

Abstract

Some high-speed SerDes devices do not come with IBIS-AMI models. For situations when an AMI model is not available, this paper describes a process for building IBIS-AMI models using SerDes datasheet information and lab measurements. The process is illustrated using a case study of a PCIe Gen3 8 Gbps SerDes device by fine-tuning template models to capture important behaviors. Stress tests and eye scans are used to further tune and correlate model to actual hardware.

Author's Biographies

Eugene Lim is hardware design engineer in NVM System Engineering group at Intel Corporation. He is currently responsible for the group's product DDR3/4, ONFI4 design signal integrity and PCIe/SATA Signal Integrity Kits. Eugene received a BSEE and MSc degrees from McGill University.

Donald Telian is an independent Signal Integrity Consultant. Building on over 30 years of SI experience at Intel, Cadence, HP, and others, his recent focus has been on helping customers correctly implement today's Multi-Gigabit serial links. His numerous published works on this and other topics are available at his website www.siguys.com. Donald is widely known as the SI designer of the PCI bus and the originator of IBIS modeling and has taught SI techniques to thousands of engineers in more than 15 countries.

Introduction

With the awareness and usefulness of IBIS-AMI analysis increasing, the demand for IBIS-AMI models for any product with high speed SerDes designs follows. While many SerDes designs have IBIS-AMI model delivery as part of the SerDes design process and customer support, some SerDes designs do not. This occurs when SerDes vendors are not yet up-to-speed with building AMI models, or SerDes design teams have moved on and left behind only legacy SPICE models which are unsuitable for system-level signal integrity simulations (see Appendix A).

This paper describes a process for building IBIS-AMI models from SerDes design collateral such as SerDes datasheet, application notes and conference papers. The process is illustrated with a model built and subsequently adapted and validated using lab data extracted through a series of stress tests on actual hardware.

From a user perspective, AMI models enable powerful types of analyses not accessible with other types of models. From a modeling perspective, AMI models allow complete freedom of implementation when compared with typical IBIS model structures while still protecting proprietary information. Recognizing that SerDes implement typical equalization structures in both the Tx and Rx – such as FFE, DFE, and CTLE – many simulation vendors offer AMI model templates that allow users to build their own AMI models. This paper describes a process that makes use of such templates. Even though IBIS-AMI modeling freed the model maker from supposed "restrictions" related to the use of templates and the structures they assume, there remain advantages to the model maker, simulation tool, and end user when a template can be used. The process described is expected to work within most tool vendor's AMI model templates.

AMI Model Development

While the paper describes building IBIS-AMI model for an 8 Gbps SerDes for PCIe Gen3 application, this process can be applied to building IBIS-AMI model for other data rates and applications.

SerDes Datasheet, application notes, and SPICE model user guides are main sources of information as building blocks of the IBIS-AMI model. These building blocks were applied to an AMI model template with the assistance of tool vendor application notes to create an IBIS-AMI model that reflects the SerDes IP behavior.

Each block in the RX and TX models have certain parameters that define the characteristics of the model. To distinguish these parameters, the model parameters are denoted by the `COURIER NEW` font. While these parameters might be specific to the tool vendor model template used, similar parameters can be found in other vendor's template models.

RX Model

A typical PCIe SerDes Rx frontend consists of the blocks shown in Figure 1. The input is the pad of the SerDes, and the output drives the Rx latch after equalization.

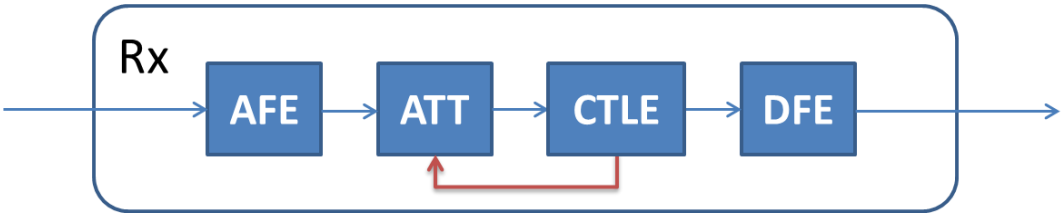

Figure 1: Typical Rx Device Frontend

These blocks are also common to the RX AMI template model from a tool vendor. The building blocks of one RX AMI template model [1] are shown in Figure 2.

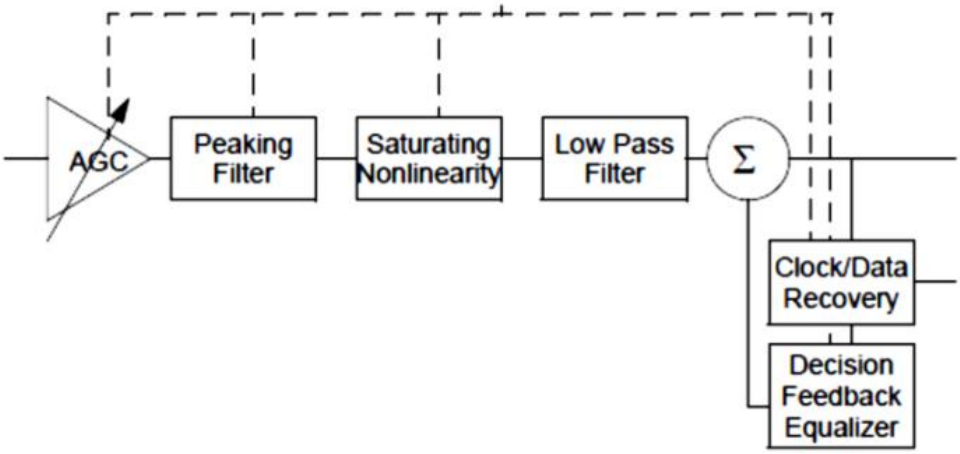

Figure 2: Rx AMI Model Template Features

While the block names can be different between the model template and the actual SerDes, the function of the blocks can help distinguish and link the blocks from SerDes to Model and are mapped in Table 1 below.

Function	SerDes Rx Frontend Block Name	AMI Model Rx Block name
Analog Front End	AFE	Analog Model
Attenuator/ Gain Control	ATT	AGC
Analog Boost/ CTLE	CTLE	Peaking Filter
Decision Feedback Equalization	DFE	DFE

Table 1: Mapping SerDes Block to Model Block

RX Analog Model Block

This section uses information from the SerDes datasheet to set the `Rt` (Receiver Differential Impedance) and the `Cc`(Receiver Capacitance) parameters in the model. The `Rt` parameter is the DC input differential termination in the datasheet and the `Cc` parameter, if unavailable, can be based on a typical value for the SerDes technology and fine-tuned using lab measurements.

RX AGC Block

For this block, model makers can typically use the SerDes' gain/attenuator curves to set the `AGC` parameter in the model. However, in this model the SerDes attenuator is used in conjunction with the CTLE to ensure a certain voltage amplitude at the input to the DFE. As such, the gain of the CTLE curves (`gains` parameters) is instead adapted to ensure a constant gain at 4GHz. This allows the `AGC` parameter to be used as a static control for overall amplitude provided to the DFE. This is described in more detail below.

RX Peak Filter Block

This block uses CLTE curve plots found in the SerDes app note to configure the CTLE of the model. For the CTLE in the template model, the frequency response for each CTLE is specified by the poles and zeros of a rational transfer function `peaking_filter.poles` and `peaking_filter.zeroes` and DC `gain`. These parameters specified in the continuous time domain (e.g. Hz), and the template model converts to sampled data coefficients at run time. A typical peaking filter characteristic family of curves is shown in Figure 3.

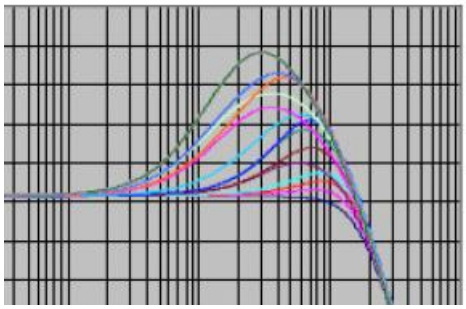

Figure 3: Typical Family of CTLE Curves

While it is possible to implement a peaking filter with curves as shown in Figure 3 (low-frequency referenced to 0dB and positive gain at ~4 GHz), this option was not chosen. Instead, the ~4GHz peak was attenuated to 0dB by manipulating the `gains` parameter. This gave the peaking filter the same characteristics as the combined effect of attenuation/CTLE in the device as well as the PCIe Specification [2]. The resulting curves are shown at left in Figure 4, with the PCIe Specification shown at right. By comparing these curves it can be seen that the model provides CTLE options beyond those suggested by the specification.

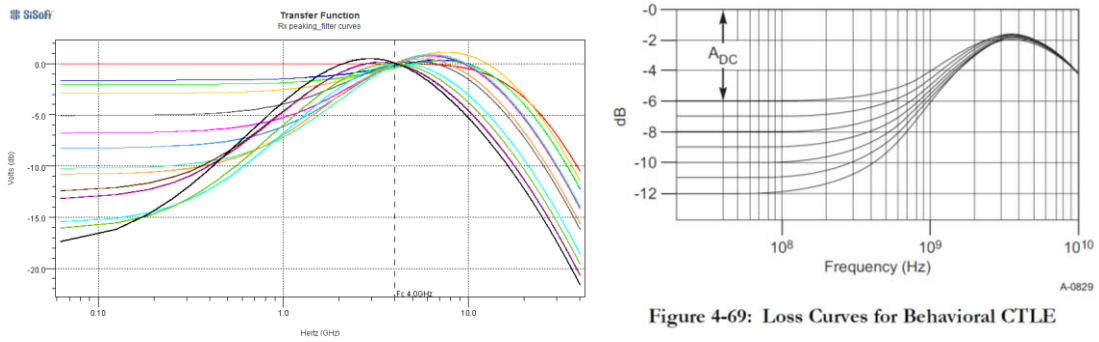

Figure 4-69: Loss Curves for Behavioral CTLE

Figure 4: CTLE Curves, Model and Specification

For the CTLE in the template model used for this study, the frequency response for each CTLE curve is specified by the poles and zeros of a rational transfer function and the DC gain ("gain"). These parameters are specified in the continuous time domain (e.g., Hz), and the template model converts to sampled data coefficients at run time. Various processes exist for deriving the correct poles and zeros, and it is also possible to manually manipulate the placeholder values in the model to produce the desired curves.

Rx Saturation/ Amplitude Control

Challenges arise when the SerDes RX Frontend architecture differs from the AMI Model template. One of the differences encountered in our case study was related to the red feedback arrow in Figure 1, which exists in the device but not necessarily in the model. The feedback exists to set the desired signal level at the input of the DFE within a 100 mV range to avoid saturation.

To model this behavior, the following sequence was used:

1. Use `gains` in Peaking Filter to adjust the peak of all CTLE curves to zero on the Y dB axis
2. Adjust `AGC` value until outer eye height stayed in the correct amplitude range across various system route lengths, with DFE off
3. Plot outer eye height vs length to confirm acceptability
4. If amplitude range is unacceptable, return to step 2. If acceptable, lock down `AGC` value.

This process worked well and provided an `AGC` parameter that can be used to additionally tune eye heights to match lab measurement.

One challenge of the process involved maintaining a consistent output amplitude range versus changes in system loss and equalization. To examine this, Figure 5 shows the amplitude (outer eye height) presented to the DFE after the combined AGC/CTLE model versus channel lengths from 4" to 32" (on the X axis). With the AGC off (red) the amplitude swings from 700mV to 250mV, while the amplitude is trained to stay in the desired range with AGC=0.16 (blue). This is an excellent result, given the 8x variation in channel length and (auto-selected) peaking filter variation (green, on the secondary Y axis). Further testing shows the output amplitudes (blue) can be adjusted up and down fairly linearly by adjusting the AGC value.

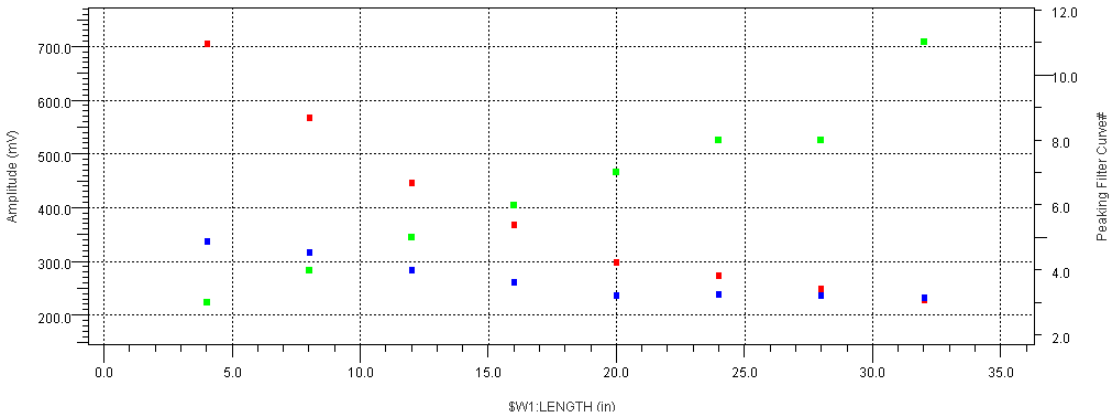

Figure 5: Amplitude at DFE vs AGC/CTLE/Channel Configuration

Rx DFE

An Rx DFE can be implemented in the model by selecting the proper number of taps and coding any saturation, non-linearity, etc. parameters the template might provide. While implementing a DFE in silicon can be complex, it is typically straightforward to implement in an algorithmic model.

Rx Jitter

Rx jitter can be added to the model using template appropriate syntiax and typical values shown below. These values should be consistent with published Tj (Total Jitter) values, which may or may not be broken down in the datasheet into lower-level jitter contributors

(e.g., Dj, Rj). However once the jitter parameters are coded into the model, they can be fine-tuned using lab measurements. Note that the _Clock_Recovery_ versions of the Rx jitter parameters are used, as these can simplify BER calibration. When specifying Rj, be sure to clarify if your template expects a peak-to-peak or rms value. For a BER of 10e-12, Rj_pp = 14.069*Rj_rms.

```
(Rx_Clock_Recovery_Dj     (Value 0.040)(Usage Info)(Type UI))
(Rx_Clock_Recovery_Rj     (Value 0.008)(Usage Info)(Type UI))
(Rx_Clock_Recovery_DCD    (Value 0.015)(Usage Info)(Type UI))
```

TX Model

The device Tx contains an analog portion (AFE) and a multi-tap FIR filter with numerous presets, as shown in Figure 6.

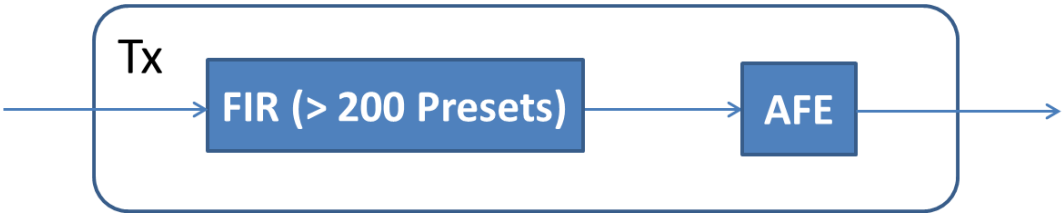

Figure 6: Typical Tx Device Blocks

Tx Analog Model Block

Similar to the RX analog model block, the information from the SerDes datasheet is used to configure the `Rs` (Transmitter Impedance), `Trf` (Rise/Fall Time), and `tx_swing` (Differential Voltage Swing) parameters in the AMI model. The `Cc` (Transmitter Capacitance) parameter in the AMI model can be based on a typical value for the SerDes technology and further using during lab measurement.

Tx FIR block

Tx equalization is straightforward to implement in an algorithmic model. PCIe Tx equalization includes one pre-cursor, one post-cursor, and a main-cursor that can reduce to 2/3 of its full value. The post-cursor can provide de-emphasis up to 1/3 of the main-cursor voltage, while the pre-cursor can provide up to ~24% of pre-shoot. The granularity of adjustment and the requirement that the absolute value of all taps sums to 1.0 results in more than 200 setting options. This range and relationship is similar to that shown in the PCIe Specification [2], shown in Figure 7.

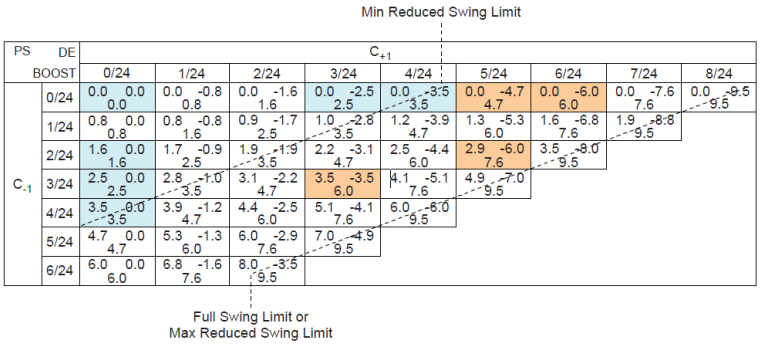

Figure 7: PCIe TxEQ Coefficient Space Matrix

Labels in the Tx .ami file can specify the amount of Preshoot and De-emphasis provided by each setting. Furthermore, certain discrete settings correspond to the PCIe-specified "Presets" P0-P9 [2, Table 4-16].

The presets were developed using a spreadsheet supplied by the SerDes vendor, by manipulating equations and text to create the correct syntax in the .ami file. Though this is a fairly large number of presets, syntax and constructs in the model made the process manageable.

Tx Jitter

Tx jitter can be added to the model using template appropriate syntax and typical values shown below. These values should be consistent with datasheet Tj (Total Jitter) values. Once the jitter parameters are added to the model they can be fine-tuned using common measurements, as demonstrated in the Lab Measurements section below.

```
(Tx_Dj          (Value 0.04)(Usage Info)(Type UI))
(Tx_Rj          (Value 0.008)(Usage Info)(Type UI))
(Tx_DCD         (Value 0.015)(Usage Info)(Type UI))
```

Lab Measurement and Correlation

Tx Correlation

In the realm of SerDes model correlation, much has been done and published regarding the Tx portion of the SerDes. As such, this paper will focus on the challenges of Rx model correlation while lightly touching on Tx correlation here.

For Tx correlation, the silicon models and measurement methods are fairly well-established and typically yield good correlation. To prove this point, the first correlation performed on the Tx model yielded the results in Table 2. As shown, prior to adapting the Tx model in any way, the model correlates to measurement to within 5% for all parameters – with the model typically on the conservative side, as desired. As additional measurements are made, the jitter can easily be adapted using the `Tx_Dj` and `Tx_Rj` parameters in the model.

Tx Parameter	Simulated	Measured	Unit	Delta
Eye Height	317	334	mV	5%
Eye Width	91	94	pS	3%
Dj	0.16	0.18	UI	-2%
Rj	0.024	0.016	UI	1%

Table 2: Simulated vs Measured Tx Parameters

Figure 9 shows the waveforms from which the measurements in Table 2 were derived, simulated (left) and measured (right).

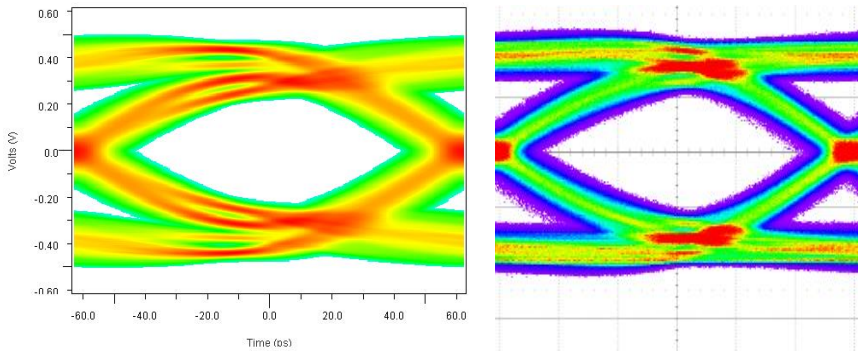

Figure 9: Tx Waveforms, Simulated and Measured

Rx Correlation

The Bit Error Rate Tester (BERT) is used commonly to test and characterize high speed serial interfaces. The BERT allows user to control composition of jitter (RJ, SJ, PJ, ISI.. etc), vertical noise interference and transmitter amplitude to create a variety of stress conditions to characterize a high speed serial device's Rx. To characterize the device under test (DUT), a serial bit pattern is sent to the DUT, the DUT SerDes will recover the data and loopback the pattern to the BERT. If the SerDes fails to recover the pattern correctly, the BERT will detect the pattern mismatch and log as bit error. This type of testing is essential as it does not require probing on the high-speed serial interface that is sensitive to any additional capacitive loading.

The BERT used has the knobs shown in Figure 10 to inject controlled impairment to perform a series of stress tests.

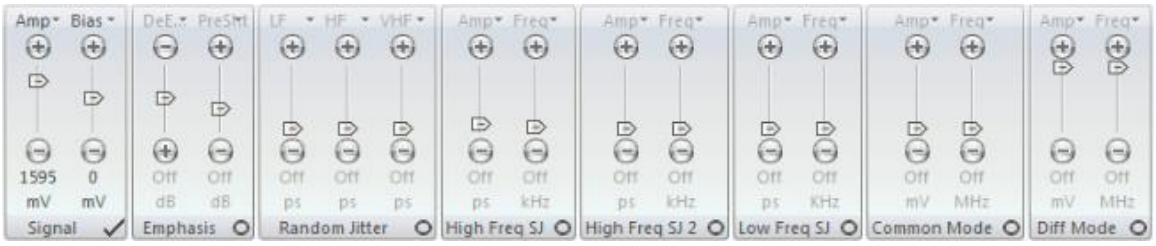

Figure 10: BERT-controlled Impairments

Table 3 shows a summary of tests by adjusting BERT knobs and behavior measured.

Stress Test	BERT Knobs	Behavior Measured
Jitter Tolerance	High Frequency SJ	Eye timing margin Clock Recovery Bandwidth
Interference Injection	Diff Mode	Eye Voltage Margin Data path gain vs Frequency Receiver Noise Figure
Receiver Sensitivity	Signal Amplitude	Minimum Latch Overdrive

Table 3: Stress Tests to Measure Rx Behaviors

The following series of stress tests have been identified to provide more accurate insight to SerDes performance and the results are used to fine-tune and confirm key parameters used in the Rx AMI model.

Rx Jitter Tolerance

Jitter Tolerance is the most used stress test method commonly found in different high speed serial interface protocol (PCIe, SATA, SAS, etc.), and produces results similar to Figure 11. A controlled sinusoidal jitter (Sj) is introduced into the transmitter clock and that sinusoidal jitter is increased in amplitude (green) until a specified bit error rate is reached (red). This procedure is repeated over a range of sinusoidal frequencies to produce an RX jitter tolerance plot.

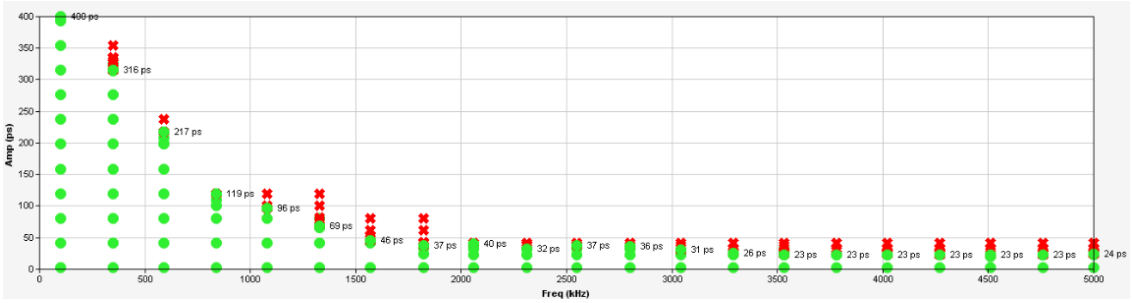

Figure 11: Jitter Tolerance Test Results

At lower jitter frequencies, the receiver's clock recovery loop tracks the jitter. However, as the frequency increases, the clock recovery loop tracking error increases. Beyond a certain frequency, the clock recovery loop has very little effect on jitter introduced into the data detection process. The timing margin for the measured bit error rate is the value where the curve flattens out. The knee of the curve is an indication of the clock recovery loop bandwidth.

Figure 12 stresses the model with Sj of increasing frequency and plots the deterioration of eye width. Note that the knee frequency is similar to that measured in Figure 10. In the model the knee frequency is varied using parameters `clock_recovery.step`, (recovered clock phase step size) and `clock_recovery.count` (early or late count to trigger a phase step) to adjust the loop bandwidth. A larger step size or smaller count value speeds up the clock recovery loop [1][5].

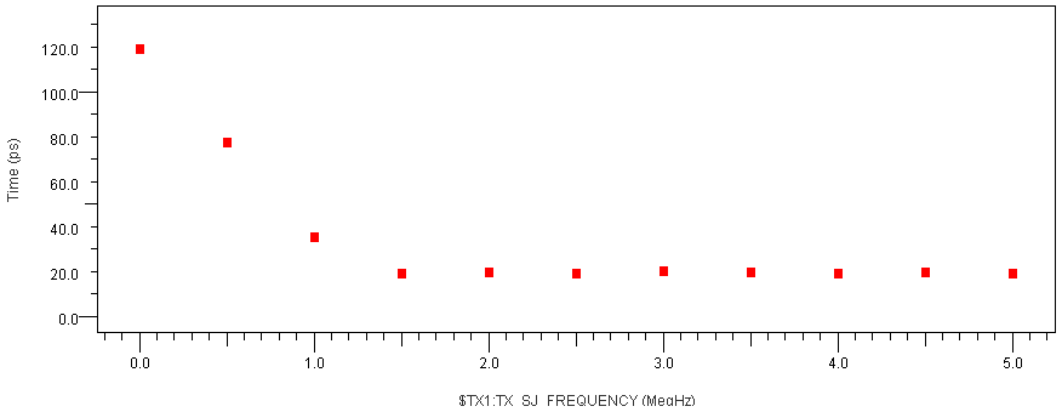

Figure 12: Model's Response to Sj Frequency

By manipulating the Rx sensitivity and clock recovery jitter parameters in the (Rj, Dj, Sj, and DCD) it is possible to map the eye widths in Figure 12 to measured BERs, enhancing the correlation of Figures 11 and 12. This mapping is partially described in [4] and is a subject of on-going research by the authors.

Interference Injection

The Interference Injection stress test couples a differential mode voltage noise onto the test pattern signal to characterize the vertical voltage margin and datapath gain response. The differential voltage noise is increased until a specified BER is reached and this is repeated over a range of frequencies. The process can be thought of as the voltage counterpart of the Jitter Tolerance stress test mentioned in the previous section.

From the lab measurement results in Figure 13, the datapath gain variation vs. frequency can observed. The datapath gain is inversely proportional to the differential noise amplitude. The injected differential noise frequency range is limited due to the range of the test equipment. The model's CTLE curves in Figure 4 is shown inset with injected noise frequency range highlighted in grey. The behavior model's CTLE curve can be seen to be the inverse of the differential noise response. Future work to further investigate the datapath and receiver decision frequency response is to use a RF signal generator with

broadband directional coupler by using a broadband coupler to increase differential noise frequency range and implement varying channel lengths to observe the effect on datapath gain response. The expected trend is as channel length increases, data path gain increased as shown in green in Figure 13.

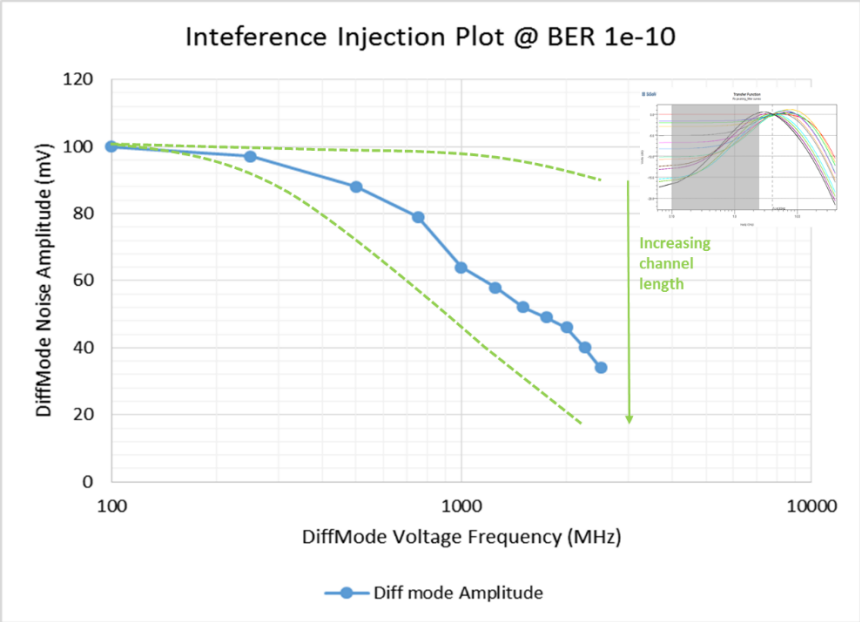

Figure 13: Interference Injection Plot

Receiver Sensitivity

The bit error rate is a very sensitive function of the transmit amplitude. A 1dB reduction in the transmitter amplitude can increase the Bit error rate many orders of magnitude. The BERT can provide a wide range of differential input to the receiver. This is useful in measuring the receiver sensitivity. If the rise/fall time of the transmitter is much smaller than the unit interval of the data and the channel has relatively low distortion, then the eye diagram at the receiver decision point will have a lot of timing margin, making the bit error rate insensitive to phase noise in recovered clock. However, when this is not true the bit error rate becomes very sensitive to the difference between the signal amplitude and the minimum latch overdrive. Therefore, the amplitude at which the signal changes abruptly will be an accurate measure of the receiver sensitivity. The AMI model's `Rx_Receiver_Sensitivity` (Receiver Sensitivity) parameter can be configured based on lab data measured in Table 4.

Tx Amplitude (mV)	BER Measured
900	0
800	0
…	0
200	0
150	1e-12
140	8e-12

130	2e-11
120	5e-2
110	2e-2
100	1e-2

Table 4: Rx Sensitivity Measurement

While the values in Table 4 demonstrate the anticipated abrupt change in BER versus amplitude applied, the voltage drop related to losses in the measurement path must be subtracted to derive the correct values for Rx Sensitivity in the model. As the AMI model is representative of die behavior, this de-embedding process should also include losses in the device's package parasitics. If the de-embedding cannot be performed directly within the measurement equipment, the process can fairly easily be done using simulation. After de-embedding, Rx Sensitivity values commonly range from 5mV to 40mV.

Summary

This paper has demonstrated a process for developing an AMI model of an advanced SerDes device using datasheet specifications and available templates. The resulting model is tuned to match silicon behavior using both simulation and measurement. The paper has shown a typical model development scenario, demonstrating common issues and appropriate solutions. As the IBIS Specification [6] does not currently specify standardized syntax for all common SerDes AMI model parameters, it is important to adapt the parameters shown to use syntax appropriate for unique template suppliers. Once done, an accurate AMI model can be created for SerDes' implementing common forms of equalization.

Acknowledgments

The authors wish to thank Cliff Jeske of Intel for his commitment to advanced SerDes simulation. Additional thanks goes to Todd Bermensolo and Aleksey Tyshchenko of Intel for reviewing this material, and Barry Katz and Michael Steinberger of SiSoft for their assistance in creating AMI models from templates and measurements.

References

[1] "SiSoft_Application_Note_Configuration_of_Advanced_Tx_and_Rx_Models", available at SiSoft's Support website. www.sisoft.com

[2] "PCI Express® Base Specification Revision 3.0" November 10, 2010, www.pcisig.org

[3] "Introducing Channel Analysis for PCB Systems" Telian, slide 28 http://www.siguys.com/resources/2004_Webinar_Introducing_Channel_Analysis.pdf

[4] "Moving Higher Data Rate Serial Links into Production – Issues & Solutions" Telian, Camerlo, Matta, Steinberger, Katz, Katz, DesignCon 2014 Best Paper http://www.siguys.com/resources/2014_DesignCon_MovingToHigherDataRates_paper.pdf

[5] "Studying Clock Recovery Performance using IBIS-AMI Models" Katz, Steinberger, DesignCon 2011 http://www.sisoft.com/elearning/secure/files/ClockRecovery_DesignCon2011.pdf

[6] IBIS Specifications at https://ibis.org/specs/

Appendix A: AMI versus SPICE Analysis

For the reader unfamiliar with the reasons why use and availability of AMI models is increasing, this section provides explanation. The following are a few reasons for using IBIS-AMI models instead of SPICE models for full channel system signal integrity analysis.

1) SPICE models simulate slowly. Transistor-level models are known to be ~10,000 times slower than AMI models for Time-Domain analysis [3] and many orders of magnitude slower for Statistical analysis. This difference is one of the reasons why AMI modeling was developed, because it enabled a more complete and robust analysis process due to its ability to process bit-streams in excess of one million bits in a reasonable amount of time.
2) SPICE models do not capture the complete details of the SerDes. Specifically, the Rx DFE is often missing. This is typical of silicon-level models, as too many transistors are required to implement the complete behavior of a DFE.
3) SPICE models are usually encrypted. This makes it difficult to adapt or configure the model correctly. Furthermore, global and model-level parameters cause conflicts that can cause simulation to crash or become unstable.
4) SPICE models are overly complex. This complexity can cause unnecessary support from both the model provider and user which results in time and resource overhead from both parties. As design cycle becomes shorter, this can easily become a bottle neck in the design.
5) SI tools prefer AMI models. Although system-level SI tools often provide ways to interface with SPICE models - for the reasons stated above - such interfaces are not the primary, optimal, and best-supported way to use the tools.