IBIS 5.0 AMI Basic Principles

Basis for existing models and existing flows

Walter Katz
IBIS AMI
October 20, 2009

SiSoft™
Signal Integrity Software, Inc.

High Speed SerDes Challenges and Simplifications

- **Simplifications**
  - **Tx final stage, and Rx input can be treated as Linear Time Invariant (LTI)**
    - **The Tx final stage, and Rx input can be included with the interconnect to in an LTI representation of the Analog-Channel**
    - **Simulation of this LTI representation of the Analog-Channel is very-very fast**
- **Complications**
  - **Tx requires signal processing of stimulus input prior to the Tx final stage.**
  - **Rx requires signal processing of the input to the Rx**
    - **Rx signal processing can be very complex**
- **IC Vendors consider signal processing algorithms proprietary IP**
- **Need to be able to simulate many millions of bits**

AMI Modeling Basic Principles

Two years ago the IBIS AMI specification (BIRD 104) was approved, and was later amended (BIRD 107). The principles that drove this specification reflected our combined knowledge of how algorithmic modeling was already being done, and our best guesstimates on near future algorithmic modeling requirements.

The current IBIS 5.0 AMI specification reflects these principles, and there are now a number of AMI that are written and operating successfully according to the current IBIS 5.0 AMI specification (modified to support digital input to Tx AMI_GetWave).

This document is a restatement of these principles, and what I have used as the basis for the proposed clarifications and improvements to the IBIS 5.0 AMI specification.

An overriding IBIS principle is that existing models that are legal for a release of IBIS remain legal for all future releases of IBIS, and that simply changing the version number of an IBIS model will maintain the legality of that model.

A SerDes Channel consist of Tx Signal Processing Analog-Channel Rx Signal Processing

The Analog-Channel is represented as an Impulse Response $h_{AC}(t)$

$h_{AC}(t)$ **is the differential mode impulse response of the interconnects between the Tx and the Rx and includes the reactive load (impedance) of the Tx final stage driver and the Rx receiver input.**

This impulse response can be created using many mathematical techniques, including, but not limited to simulation and TDR measurement.

Traditional IBIS does not model differential LTI buffers, this can be addressed with the introduction new AMI parameters.

A SerDes Channel consist of
Tx Signal Processing
Analog-Channel
Rx Signal Processing

Tx Signal Processing can be either LTI or non-LTI.

If LTI, it can be represented by an impulse response $h_{TEI}(t)$

If non-LTI, it is represented as a function $g_{TEG}()$ that takes a waveform in and outputs a waveform. Sometime there is useful LTI approximation when signal processing is non-LTI.

A SerDes Channel consist of Tx Signal Processing Analog-Channel Rx Signal Processing

Rx Signal Processing can be either LTI or non-LTI.

- **If LTI, it can be represented by an impulse response $h_{REI}(t)$**
- **If non-LTI, it is represented as a function $g_{REG}()$ that takes a waveform in and outputs a waveform. Sometimes there is useful LTI approximation when signal processing is non-LTI.**

AMI models are delivered as executable code and an ASCII .ami file

Executable code is in the form of a Shared Object (SO) or Dynamically Linked Library (DLL). For the purposes of this document DLL shall include the meaning of Shared Object.

- **All AMI DLL's have an AMI_Init entry and an AMI_Close entry. If a model is non-LTI, then is must also have an AMI_GetWave entry. If a model does have an AMI_GetWave entry, then it is telling the EDA tool that the model is non-LTI, and that using any LTI impulse response approximations of the models equalization may not accurately model the channel.**

The .ami ASCII file tells the EDA tool how to use the model.

- **Tells model if there is an AMI_GetWave entry.**
- **Information on how to configure the model.**
- **Information on how to use the results that the model generates.**

If a Tx and Rx models are LTI

LTI models should not have an AMI_GetWave entry.

The input to a Tx AMI_Init function is the impulse response of the channel $h_{AC}(t)$, and the model configuration. The output of the Tx AMI_Init function is the impulse response of the channel convolved with the impulse response of the Tx equalization (filter) $h_{AC}(t) \otimes h_{TEI}(t)$.

The input to an Rx AMI_Init function is the combined impulse response of the channel and Tx equalization $h_{AC}(t) \otimes h_{TEI}(t)$, and the model configuration. The output of the Rx AMI_Init function is the impulse response its input ($h_{AC}(t) \otimes h_{TEI}(t)$) convolved with the impulse response of the Rx equalization ($h_{AC}(t) \otimes h_{TEI}(t) \otimes h_{REI}(t)$).

If a Tx and Rx models are non-LTI

Both Tx and Rx must have an AMI_GetWave entry. AMI_Init must first be called to initialize both models for subsequent AMI_GetWave calls.

The input to a Tx AMI_Init function is the impulse response of the channel $h_{AC}(t)$, and the model configuration. The output of the Tx AMI_Init function is the impulse response of the channel ($h_{AC}(t)$) which *may* be convolved with an approximate impulse response of the Tx equalization ($h_{AC}(t) \otimes h_{TEI}(t)$). The Tx AMI_Init call also stores information about the conditions of the model (e.g. tap coefficients) that are used by the Tx AMI_GetWave call.

The input to an Rx AMI_Init function is the combined impulse response of the channel and Tx equalization $h_{AC}(t) \otimes h_{TEI}(t)$, and the model configuration. The output of the Rx AMI_Init function is the impulse response input ($h_{AC}(t) \otimes h_{TEI}(t)$) which *may* be convolved with an approximate the impulse response of the Rx equalization ($h_{AC}(t) \otimes h_{TEI}(t) \otimes h_{REI}(t)$). The Rx AMI_Init call also stores information about the conditions of the model (e.g. tap coefficients) that are used by the Rx AMI_GetWave call.

If a Tx and Rx models are non-LTI (What if impulse response output of AMI_Init is not modified)

If the Tx AMI_Init call does not return a modified impulse response the Rx AMI_Init call may not be able to optimally initialize the conditions for the subsequent Rx AMI_GetWave calls.

If either the Rx or Tx AMI_Init call do not return a modified impulse response then statistical analysis is disabled for this Tx and Rx model pair.

If a Tx and Rx models are non-LTI (Tx and Rx AMI_GetWave can be called multiple times)

A long stimulus pattern is partitioned into multiple smaller stimulus blocks. Each stimulus block is first processed through the Tx AMI_GetWave call. The Tx AMI_GetWave call applies the equalization prescribed by how the model was initialized by the Tx AMI_Init call. The output of this Tx AMI_GetWave is in turn passed on as input to the Rx AMI_GetWave call. The Rx AMI_GetWave call applies the equalization prescribed by how the model was initialized by the Rx AMI_Init call, and outputs a waveform at the decision point of the Rx model, and may also output clock ticks to indicate when the waveform is sampled by the model.

This Tx AMI_GetWave and Rx AMI_GetWave call sequence is repeated until all of the blocks in the long stimulus pattern are processed.

If a Tx and Rx models are non-LTI (Notes)

The model writer determines how the AMI_GetWave call uses the information passed to it from the AMI_Init call. The AMI_GetWave function may include LTI sections and a non-LTI sections. The LTI section may be passed to it from the AMI_Init call. The initialization of the non-LTI section may be passed to it from the AMI_Init call.

Equalization and filters will in general overlap the end of one stimulus block and the beginning of the next stimulus block. AMI_GetWave functions will normally need to implement some sort of Overlap and Save algorithm to accomplish this.

LTI models need to work with non-LTI models

The author of a Tx model has no information about which Rx model is going to be used with it. Similarly the author of an Rx model has no information about which Tx model is going to be used with it. So an author that writes an LTI Tx model has to anticipate that it will be used with a non-LTI Rx model ....

The EDA tool needs to be able to emulate the function of the AMI_GetWave call for LTI models when the other model in the channel has AMI_GetWave.

AMI models need to support a flow that allows analysis of waveforms generated by external sources.

External sources might include

- **SPICE simulation**
- **Sampled Data**
- **AMS simulation**
- **…**

In particular, this means that the EDA tool needs to be able to determine the equalization of the Rx LTI filter.

The Fundamental Principle of AMI Modeling

The fundamental principle of AMI modeling is that every EDA platform (both software and hardware) will give the same results when presented with the same Analog-Channel impulse response, the same AMI model conditions, and the same input stimulus pattern.

Each EDA platform may differ on how it chooses the inputs to the AMI model: Tx and Rx AMI model conditions, stimulus pattern, and the Analog-Channel impulse response. Each EDA platform may differ on how it processes the resulting outputs.