Keysight Technologies
Explore the SERDES design space
using the IBIS AMI channel simulation flow

White Paper

KEYSIGHT
TECHNOLOGIES

Introduction

Simulation of modern chip-to-chip links requires you abandon the SPICE-based approach and adopt a new approach based on an IBIS AMI channel simulation flow.

Why you can't use SPICE for today's SERDES

To avoid the skew inherent in parallel chip-to-chip links, modern ICs use multigigabit/s serial links with a serializer and deserializer (SERDES) input/output (I/O) circuitry at the transmitter (Tx) and receiver (Rx), respectively. Traditional SPICE-based simulation tools fail in the multigigabit regime for two reasons:

The I/O circuitry is too complex

In contrast to older sub gigabit SERDES (which contained just a handful of components to form the analog circuit), today's SERDES include a logic block that implements the signal processing circuitry required to mitigate the severe impairments that FR4 printed circuit board traces impose on multi-gigabit signals. Previously, the throughput of a SPICE-like simulator was thousands of bits per minute of simulation time. When the SPICE tool is presented with logic blocks containing 10,000-50,000 transistors, the throughput drops dramatically: only tens or a best hundreds of bits per minute of simulations time. It can take tens of hours or even several days to collect enough result bits to build a useful eye pattern diagram.

The design space is too large

When the only design parameters were the channel characteristics, plus one or two settings on the Tx and on the Rx – maybe gain and termination impedance – the signal integrity engineer needed to run only a handful of simulations to explore the design space for the optimum combination. In contrast, today's Tx and Rx come with dozens of field-settable registers that contain parameters that must be tweaked to open the eye. Examples include Tx de-emphasis filter tap settings, Rx equalizer tap settings, Rx clock data recovery (CDR) loop bandwidth and so on. To find the optimum combination the SI engineer must run thousands of simulations each of which must generate a figure of merit based on eye height and width at some specified BER contour value. SSD can be easily removed from the front panel and secured when not in use.

Dr. Colin Warwick, High Speed Digital Product Manager
Dr. Fangyi Rao, Master R&D Engineer
Keysight EEsof EDA

The IBIS Open Forum recognized the need for a new approach so when IBIS 5.0 was ratified in August 2008, it added a radical departure from the traditional SPICE-like IBIS flow. The new flow is specifically aimed at multigigabit SERDES and is built around a new class of simulator called a channel simulator.

The Forum set itself six goals in creating the new flow:

IP Protection

Models cannot be reverse-engineered because only machine code is shipped. IC vendors control which details are exposed to the user, without the need for the proprietary encryption keys that make models non-portable.

Portability

The same IC model runs in different IBIS-AMI simulators. IC vendors can 'write once, run anywhere.' They don't have to support one model for every EDA tool vendor. The signal integrity engineer can choose the best-in-class simulator, and not be forced into using the one that happens to support a particular IC model. Note however, that some models that claim to be AMI models actually use proprietary semantics and are therefore non-portable. Buyers beware!

Interoperability

Models from different semiconductor vendors work together e.g. a TI DSP connected to a Xilinx FPGA.

Performance

Ultralow BER contours in seconds not weeks.

Flexibility

The flow offers two modes: statistical and bit-by-bit (also known as "time domain") modes. Statistical mode is faster but less flexible. Specifically, the IC models must be linear and time invariant (LTI) in statistical mode, whereas they can be non-linear and time varying (NLTV) in bit-by-bit mode. Simple circuits like a de-emphasis filter are LTI, but for more complex circuits like adaptive equalizers and clock/data recoverers, you have to invest in an NLTV representation.

Optimization

Models expose control parameters (e.g. equalizer tap settings) that model the registers you set on the actual chip. You can set these manually in the simulator user interface (UI). Or you can allow a batch mode or optimization controller to set these for automatic and rapid design space exploration.

The last five goals are aimed at making the AMI approach technically sound, but the first one is aimed at overcoming a business issue related to the supply chain: it is in the interest of both IC vendors and IC consumers (the OEMs) for the IC vendor to supply the information needed to design the chip into a system. But there is a partly conflicting need on the part of the IC vendor to protect their IP. Encryption works, but this makes the models non-portable, forcing the IC vendors to create and support one model for each EDA vendor's encryption key. Thus, in the AMI approach, IC vendors provide only machine code executable representing behavior, not implementation. This gives adequate IP protection and reasonable portability. IC vendors must however provide one executable per computer platform (e.g. Win32, Win64, and Linux64) because executables are not portable across operating systems.

As mentioned above, to accomplish these ambitious six goals, the AMI flow is based on an entirely new type of simulator, called a Channel Simulator. Unlike SPICE, which uses computationally expensive algorithms like modified nodal analysis of the Kirchhoff current laws, Channel Simulator uses computationally inexpensive algorithms like superposition, convolution, and statistical analysis of the end-to-end impulse response. The eye pattern diagram is generated thousands or millions of times faster in a channel simulator compared to a SPICE-like simulator. To achieve the flexibility goal, the AMI flow allows five cases and this complicates any narrative describing the flow. For this reason, most of the rest of this article will explain each in some detail. If this is the first time you've read about AMI, it might take you a couple of readings to absorb all the implications.

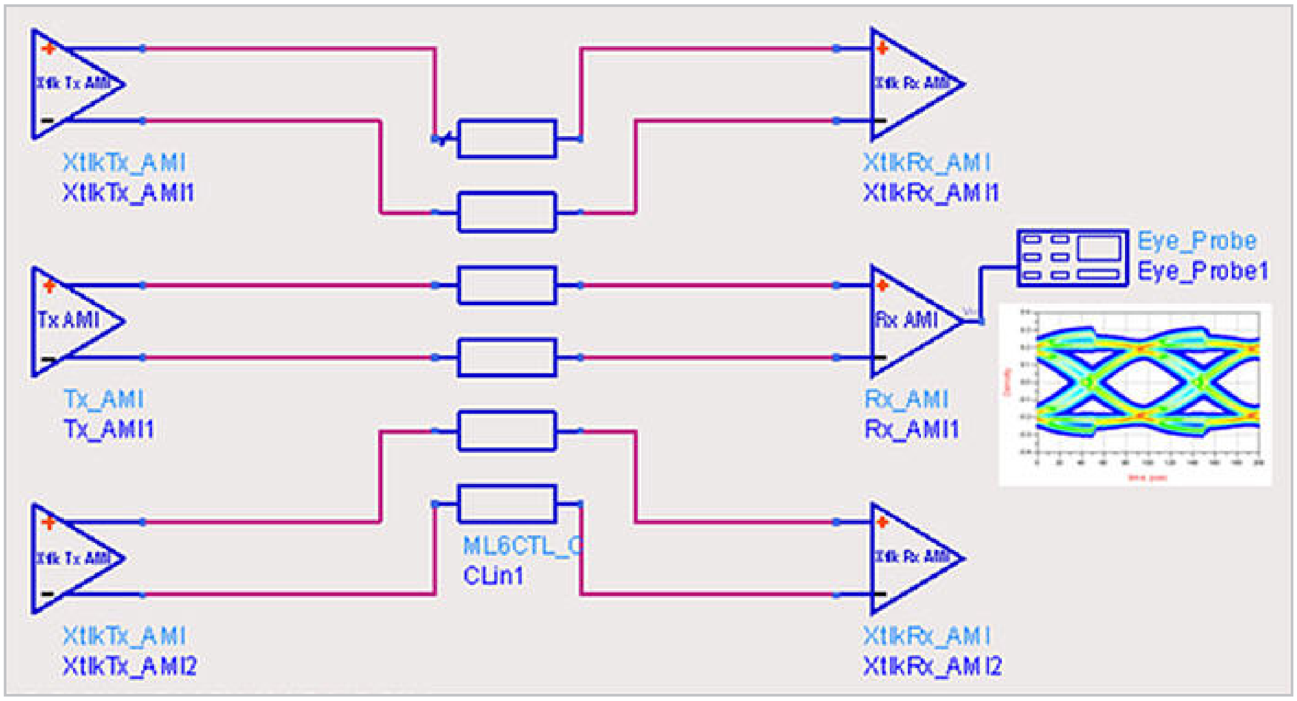

Figure 1. Representative topology for a Channel Simulator schematic. The transmitter IC (Tx) is middle left, the channel (representing the PCBs traces and so on) is in the middle center. The receiver IC (Rx) is middle right. Above and below the through (also known as the "victim") channel are the crosstalk (also known as the "aggressor") channels. Two aggressors are shown here, but the standard allows for any number from zero up.

In Advanced Design System (ADS) from Keysight Technologies, Inc. EEsof EDA, the Channel Simulator supports both AMI flow modes: bit-by-bit and statistical. IBIS 5.1 allows four cases in bit-by-bit mode but only one in statistical mode. Which of the four cases used in bit-by-bit mode depends on whether the Tx and Rx models have or do not have a function called the AMI_GetWave() function. This strangely named function is the mechanism IBIS AMI uses to support NLTV IC model behavior in bit-by-bit mode. In case you are wondering, the LTI impulse response behavior is captured in an equally strangely named function: AMI_Init(). As the standard coyly admits:

“While the primary purpose of the AMI_Init function is to perform the required initialization steps, it may also include LTI signal processing algorithms. Therefore, statistical simulations may be performed using the AMI_Init function alone.”

Model builders must provide either an AMI_GetWave() function or an impulse response inside AMI_Init(). Optionally, they may provide both, but (obviously) not an empty model:

Two Kinds of Tx/Rx Plus a “Dual” or “Hybrid” model		Model’s Init_Returns_Impulse flag is:	
		False (“Can’t be modeled as LTI”)	True (“LTI model via impulse response”)
Model’s GetWave_ Exists flag is	False (“Model is pure LTI”)	Empty model: not allowed	Typical case for Tx and simple Rx’s (fixed Eq. and no CDR)
	True (“NLTV model via waveform modification”)	Typical case for Rx (Adaptive Eq., CDR)	Buyer beware: LTI approximation of NLTV device if used in stat mode

Table 1. Two kinds of AMI model, plus a “hybrid”

If a model has both, the Channel Simulator decides which one to use according to the details of the case, given below. The following tables show the allowed and disallowed cases. For the purposes of this article, let’s give them names --Cases 1-5, but bear in mind that IBIS 5.1 doesn’t use these names.

		Tx model’s Init_Returns_Impulse flag is	
		False (“Tx cannot be modeled nor approximated as LTI”)	True (“Tx can be modeled as LTI using AMI_Init()”)
Rx model’s Init_ Returns_ Impulse flag is	False (“Rx cannot be modeled nor approximated as LTI”)	Not recommended. ChanSim issues warning	Not recommended. ChanSim issues warning
	True (“Rx can be modeled as LTI using AMI_Init()”)	Not recommended. ChanSim issues warning.	“Case 1”

Table 2a. Channel Simulator set to statistical mode

		Tx model’s GetWave_Exists flag is	
		False (“Tx does not have AMI_GetWave()”)	True (“Tx has AMI_GetWave())
Rx model’s GetWave_ Exists flag is	False (“Rx does not have AMI_ GetWave()”)	“Case 2”	“Case 5” (Practically never used)
	True (“Rx has AMI_ GetWave()”)	“Case 3” (Most common case)	“Case 4”

Table 2b. Channel simulator set to bit-by-bit mode. (Note: In IBIS 5.0, there was a third flag (“Use_ Init_Output”) besides the present Init_Returns_Impulse and GetWave_Exists flags. However, this flag caused much confusion and so Use_Init_Output is deprecated in IBIS 5.1.)

Here are the details of all five allowed cases. For simplicity here, we’ll focus on the through channel modeling:

Pre-work for all five cases

1. Calculate the impulse response of the analog circuitry and the channel:

$$h_{ch}$$

The analog circuitry and channel must be LTI in all cases. Normally you don’t need to see these but in case you do, ADS writes this into a *.txt file in the .../data folder of your project. The file name is `.../data/imp_tx_int.txt`.

2. Using AMI_Init() function of the Tx, convolve the Tx response with the channel response:

$$h_{ch.Tx} = h_{ch} \otimes h_{Tx}$$

...where $\otimes$ signifies convolution. Note that the AMI flow allows the Tx to be “smart” in step 2. Specifically, it may generate an impulse response h_{Tx} that mitigates the impairment represented by the input argument h_{ch}. Ideally, the Tx should set h_{Tx} such that it is the inverse of h_{ch}. ADS writes the resulting composite impulse response to both `.../data/imp_tx_out.txt` and `.../data/imp_rx_in.txt`

3. Using the AMI_Init() function of the Rx model, convolve the Rx response with the composite channel/Tx response:

$$h_{ch.Tx.Rx} = h_{ch.Tx} \otimes h_{Rx}$$

AMI flow also allows the Rx to be “smart” in step 3. It too may generate an impulse response h_{Rx} that mitigates the impairment represented by the input argument $h_{ch.Tx}$. Ideally, the Rx should set h_{Rx} such that it is the inverse of $h_{ch.Tx}$. ADS writes it to `.../data/imp_rx_out.txt`.

Case 1

Statistical mode

1. $h_{ch.Tx.Rx}$ is used to calculate eye pattern diagram and associated metrics using statistical methods.

In this case, no lengthy time domain waveform is needed so ultralow BER contours (e.g. 10^{-18} or even lower) of the eye pattern diagram can be obtained in seconds. Contrast this to SPICE which would take days or weeks. A restriction in Case 1 is that the channel, Tx, and Rx must all be LTI. Another restriction is that the user cannot set the bit pattern. In fact there is no bit pattern per se. The statistical calculations are based on the stochastic properties of a conceptually infinite bit sequence.

One advantage of ADS over competing channel simulators is that ADS properly captures jitter amplification through the channel to accurately predict eye opening. Some other EDA tools ignore this effect and thus overestimate the eye opening. For more details on jitter amplification, please see the IEEE EPEPS 2012 conference paper "Frequency Domain Analysis of Jitter Amplification in Clock Channels" by Rao and Hindi.

Note that if an IC models has an AMI_GetWave() function, it is ignored in Case 1. This poses a troubling question. If a model builder thought it necessary to place an AMI_GetWave() function in the model, we can infer that the real IC is NLTV. We make this inference because if the IC was exactly LTI, the model builder would have deemed the impulse response sufficient. By also furnishing the impulse response and thus allowing statistical mode operation, the model builder is implying that the provided impulse response is a sufficiently accurate LTI approximation of an inherently NLTV function. You should ask your model builder what the limits of this approximation are. If you are not satisfied with the answer, then we recommend you forego statistical mode and use bit-by-bit mode. Although it is slower, it always uses the more accurate method (i.e. AMI_GetWave() versus the LTI approximation) if both exist in the same model.

Case 2

Bit-by-bit mode when neither IC models have an AMI_GetWave() function

1. Generate the desired bit pattern:

$$b_{Tx.in}(t)$$

2. Convolve the composite channel/Tx/Rx response with the bit pattern to obtain an output waveform:

$$w_{Rx.out}(t) = h_{ch.Tx.Rx} \otimes b_{Tx.in}(t)$$

3. Create the eye pattern diagram from the Rx output waveform

This case is similar to statistical mode in that the channel, Tx, and Rx must all be LTI. It is slower than statistical mode especially for the long bit patterns needed for ultralow BER contours, but it does offer more flexibility in that the user can select any bit pattern they choose (worst case sequences, a particular line code, and so on).

Case 3

Bit-by-bit mode when the Tx model does not have an AMI_GetWave() function, but the Rx model does

1. Generate the desired bit pattern:

$$b_{Tx.in}(t)$$

2. Convolve the composite channel/Tx response with the bit pattern to obtain a Rx input waveform:

$$w_{Rx.in}(t) = h_{ch.Tx} \otimes b_{Tx.in}(t)$$

3. Pass the Rx input waveform to the Rx's AMI_GetWave function:

$$w_{Rx\,out}(t) = AMI\_GetWave_{Rx}\left(w_{Rx\,in}(t)\right)$$

4. Create the eye pattern diagram from the Rx output waveform

This is the most common case in practice. The Tx is often an LTI circuit such as a de-emphasis filter, the channel is LTI, and the Rx is a non-linear time-varying (NLTV) circuit like an adaptive equalizer and/or clock/data recoverer. The throughput is thousands of times faster than SPICE, on the order of a million bits per minute, and simulation length of 10^8 bits are easily achievable.

Case 4

Bit-by-bit mode when both IC models have an AMI_GetWave() function

1. Generate the desired bit pattern:

$$b_{Tx.in}(t)$$

2. Pass the bit pattern to the Tx's AMI_GetWave function:

$$w_{Tx\,out}(t) = AMI\_GetWave_{Tx}\left(b_{Tx\,in}(t)\right)$$

3. Convolve the channel response with the Tx output waveform to obtain a Rx input waveform:

$$w_{Rx.in}(t) = h_{ch} \otimes w_{Tx.out}(t)$$

4. Pass the Rx input waveform to the Rx's AMI_GetWave() function:

$$w_{Rx\,out}(t) = AMI\_GetWave_{Rx}\left(w_{Rx\,in}(t)\right) \quad)$$

5. Create the eye pattern diagram from the Rx output waveform

This allows even more flexibility than case 3 because both the Tx and Rx can be NLTV. The Rx can be "smart" in that it can use the foreknowledge from step 3 of the pre-work to optimize its AMI_GetWave() function for maximum mitigation of the impairment represented by $h_{ch.Tx}$.

Case 5

Bit-by-bit mode when the Tx model does have an AMI_ GetWave() function, but the Rx model doesn't

1. Back solve for the Tx response using deconvolution (deconvolution is signified here by $\%$):

$$h_{Tx} = h_{ch.Tx} \% h_{ch}$$

2. Back solve for the composite channel and Rx response.

$$h_{Rx.ch} = h_{ch.Tx.Rx} \% h_{Tx}$$

3. Generate the desired bit pattern:

$$b_{Tx.in}(t)$$

4. Pass the bit pattern to the Tx's AMI_GetWave function:

$$w_{Tx.out}(t) = AMI\_GetWave_{Tx}\left(b_{Tx.in}(t)\right)$$

5. Convolve the composite channel and Rx response with the Tx output waveform to obtain a Rx output waveform:

$$w_{Rx.out}(t) = h_{Rx.ch} \otimes w_{Tx.out}(t)$$

6. Create the eye pattern diagram from the Rx output waveform

Although the IBIS AMI standard allows this case for completeness, it is not used much in practice. It involves counterintuitive deconvolutions of the responses that were convolved in the pre-work. The reason these are needed is because, as mentioned above, the AMI_Init() function of the Tx and Rx can be "smart." They can optimize their responses based on the responses passed into them. For this reason it is necessary to "back out" the Tx response from $h_{ch.Tx.Rx}$ in order that the Tx response is not double counted. This case is the most complex one and you may need some time to digest all the nuances in the unlikely event that you plan to use it.

Jitter, Crosstalk, Repeaters, and Backchannel

The standard also allows one to model jitter and cross-talk. We anticipate a future version will codify advanced techniques that best-in-class Channel Simulators such as the one in ADS support as proprietary (and therefore non-portable) capabilities today. For example ADS 2012 supports a proposed change to a future version of the standard for advanced jitter called BIRD 123. It also supports on-die s-parameters in a way that is consistent with BIRDs 116-118. Other examples include backchannel signaling and NLTV channel elements such as repeaters. However, these all of these topics are beyond the scope of this article.

Summary

Channel simulation of SERDES with the IBIS AMI flow offers many advantages over other techniques including interoperability, portability, performance, flexibility, optimization, and IP protection. We expect it to become the 'gold standard' in chip-to-chip link simulation in the future. If you'd like to try this technique on your project, ADS Channel Simulator is available for evaluation at:

www.keysight.com/find/eesof-ads-evaluation

Americas	
Canada	(877) 894 4414
Brazil	55 11 33 51 7010
Mexico	001 800 254 2440
United States	(800) 829 4444
Asia Pacific	
Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	0120 (421) 345
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Other AP Countries	(65) 6375 8100
Europe & Middle East	
Austria	0800 001122
Belgium	0800 58580
Finland	0800 523252
France	0805 980333
Germany	0800 6270999
Ireland	1800 832700
Israel	1 809 343051
Italy	800 599100
Luxembourg	+32 800 58580
Netherlands	0800 0233200
Russia	8800 5009286
Spain	800 000154
Sweden	0200 882255
Switzerland	0800 805353
	Opt. 1 (DE)
	Opt. 2 (FR)
	Opt. 3 (IT)
United Kingdom	0800 0260637