DesignCon 2015

A New Methodology for Developing IBIS-AMI Models

Hongtao Zhang, Xilinx Inc.
hongtao@xilinx.com

John Baprawski
john.baprawski@gmail.com

Pegah Alavi, Keysight Technologies
pegah_alavi@keysight.com

Geoff Zhang, Xilinx Inc.
geoff.zhang@xilinx.com

Abstract

High speed serial link channel simulation is critical for system design and validation. The simulation must run fast while achieving adequate accuracy. Today, system vendors require Input/Output Buffer Information Specification Algorithmic Model Interface (IBIS-AMI) models long before silicon is available. To meet the demand, silicon vendors desire an efficient process for generating IBIS-AMI models efficiently and reliably.

This paper first discusses the challenges facing the IBIS-AMI model development. Then it proposes a new methodology for AMI model generation, along with engineering trade-offs needed. The methodology and procedure this paper presents have shortened the development cycle over the previously used method for a 20nm 32Gbps Serializer-Deserializer (SerDes). The developed IBIS-AMI model was tested across multiple Electronic Design Automation (EDA) platforms. Detailed modeling techniques for accuracy and speed improvement are discussed throughout the paper.

Authors Biography

Hongtao Zhang received his Ph.D. degree in Electrical and Computer Engineering from University of California, San Diego in 2006. He joined Xilinx in 2013 as a staff SerDes architect, developing SerDes architectures for both NRZ and PAM4 signaling. From 2010 to 2013, he was with SerDes design team at Oracle Corporation, where he worked on circuit design and architecture modeling. Prior to that, he worked on SerDes characterization at Texas Instruments, Dallas. His current interests are SerDes architecture development and simulation, analog and digital circuit implementation and optimization, and system level modeling.

John Baprawski is a consultant for Electronic System Level (ESL) Design. Since June 2010, he has focused on modeling high speed digital (HSD) integrated circuits (ICs) based on the industry Input/Output Buffer Information Specification (IBIS) Algorithmic Model Interface (AMI) standard. He created the Agilent Technologies AMI Training class based on their SystemVue and Advanced Design System (ADS) tools. He conducted AMI Training at over 30 high speed digital IC semiconductor companies and created many custom transmit (TX) and receive (RX) AMI models. Prior to this consulting work, John was the R&D manager at Agilent Technologies EEsof Division for 22 years with responsibility for initiating and evolving their system level design tools including OmniSys, ADS Ptolemy, Wireless Libraries and SystemVue products. He holds a Master's degree in Electrical Engineering from University of California, Los Angeles, and PhD studies at University of Michigan. His web site is www.johnbaprawski.com.

Pegah Alavi is a Senior Application Engineer with Keysight Technologies, where her main area of focus is high speed digital systems and designs. Prior to Keysight Technologies, Pegah worked in Texas Instruments and National Semiconductor. She has received her BSEE and MSEE from UCLA and Santa Clara University.

Geoff Zhang received his Ph.D. in 1997 in Microwave Engineering and Signal Processing from Iowa State University. He joined Xilinx SerDes Technology Group in 2013 to lead the SerDes architecture and modeling group. Since 1997 Geoff has worked at Xilinx, Huawei, LSI, Agere Systems, Lucent, and Texas Instruments. His current work involves SerDes architecture modeling and high speed system level analysis.

1. Introduction

In high speed serial communication, the increase of data rate lowers system margin. This means that either overdesigns are permitted, or at early stages of product definition, system engineers rely more heavily on link simulation to assess system performance. In turn, models must be more accurate and they are required at an earlier design stage, often before silicon is available. These requirements have made IBIS-AMI model development more challenging.

IBIS-AMI models have become ubiquitous among engineers working on high speed serial links. They are used for system level simulation to determine a product-worthy solution that delivers adequate system performance margin while maintaining competitive cost. With good IBIS-AMI modeling, the simulation should run fast so that system engineers can simulate tens of millions of bits to reveal system margin more accurately.

However, developing accurate IBIS-AMI models becomes a burden for SerDes vendors, as typically the SerDes model developed for the purpose of architecture analysis is not readily convertible to its IBIS-AMI counterpart. The often used architecture model today is based on C/C++ codes, Matlab scripts, Simulink models, and proprietary tools. In addition, the developed IBIS-AMI models have to run in both Windows and Linux systems.

There are many tradeoffs in generating IBIS-AMI models. Some of these will be addressed in this paper. The development process of an IBIS-AMI model for a 20nm 32Gbps SerDes is presented. The development steps for creating the IBIS-AMI model are illustrated. Some modeling trade-offs for accuracy and speed improvement are discussed.

The developed IBIS-AMI model was verified across multiple EDA platforms and correlates well with the design. The architecture of the SerDes for the AMI model developed is more complex compared to previous generations, yet the AMI model runs faster with better accuracy.

2. Brief Overview of IBIS-AMI Modeling

2.1 What IBIS-AMI models are and why they are needed

IBIS models were first generated by Intel as a means of providing models for external customers. Intel shared this work with EDA vendors in 1993 and other companies started adopting this. IBIS stands for Input/output Buffer Information Specification. IBIS files are behavioral models of individual analog buffers. As the chip designs became more and more complex, more DSP techniques were needed to maintain an open eye at the receiver data slicer. While the IBIS Open Forum worked with the industry to amend the IBIS standard to incorporate changes, analog IBIS buffer models could no longer keep up with the DSP design. Vendors were again providing their own encrypted models which were platform specific. Thus the IBIS community worked to extend the IBIS models and IBIS-AMI models were born.

AMI stands for Algorithmic Modeling Interface and as the name indicates, it is designed to handle modeling of the algorithmic functions of an I/O, namely the DSP. IBIS-AMI models provide the end user with the model portability that they need while ensuring the vendors that their IP is protected. Along with AMI a new class of simulators, the channel simulator, was introduced. Channel simulators make the assumption that the transmission channel is Linear and Time Invariant (LTI). This is an assumption that holds for the majority of high speed SerDes channels. Using this assumption, channel simulators are able to run very fast simulations with IBIS-AMI models. The combination of channel simulators and IBIS-AMI models can increase the simulation speed by multiple orders of magnitude in time-domain like analysis. Channel simulators also allow statistical simulation of IBIS-AMI models, where BER values of 10^{-18} or lower can be achieved in a matter of minutes.

Because of the ultra-fast simulation speeds and high level of simulation accuracy that can be achieved with diligently developed IBIS-AMI models, IBIS-AMI models have become an integral part of SerDes system design. SerDes vendors routinely provide IBIS-AMI models with their chips, in most cases before the final chips have been released. End users depend on high fidelity IBIS-AMI models and an accurate channel simulator to be able to predict their system behavior in their simulations.

IBIS-AMI models are highly configurable. Model developers can design their models so that the equalization settings, adaptation loop parameters and PVT corners can be chosen by the user. This allows the model to match the hardware performance in a real system.

2.2 How IBIS-AMI models are used in system simulations

IBIS-AMI models are developed for SerDes devices, namely Transmitters, Receivers, and mid channel devices such as Redrivers and Retimers.

Figure 2.1 shows a typical system to be simulated. Figure 2.2 shows the same system with block level descriptions. The main point to consider here is that what is shown as the channel contains many parts which may be individually modeled. This includes the connectors, Printed Circuit Board (PCB) traces, vias, and package models to name a few. The composite channel includes all of these with the addition of the analog IBIS buffer models. As mentioned before, channel simulators make the assumption that this composite channel is LTI. This enables the channel simulator to use fast simulation techniques which enable users to run millions of bits in a matter of minutes.

Since the composite channel is LTI, the channel simulator determines its impulse response. This impulse response is used for fast system simulation and analysis.

Once the composite channel has been characterized by the channel simulator, the transmitter's AMI model's effect on the channel is analyzed. If a bit-by-bit simulation (analogous to time domain simulation) is being performed, then every bit generated by the transmitter is convolved with the composite channel model's impulse response.

The last step in this system simulation is to consider the effects of the receiver. The receiver's AMI model is then simulated with the function resulting from transmitter's convolution with the composite channel. This is represented as the final system's response.

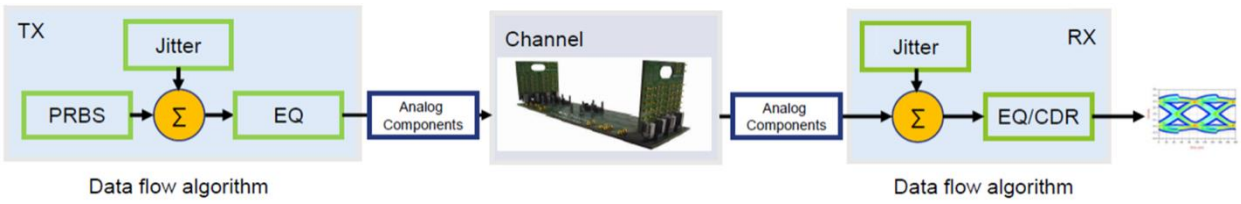

Figure 2.1: End to End Simulation with IBIS-AMI models

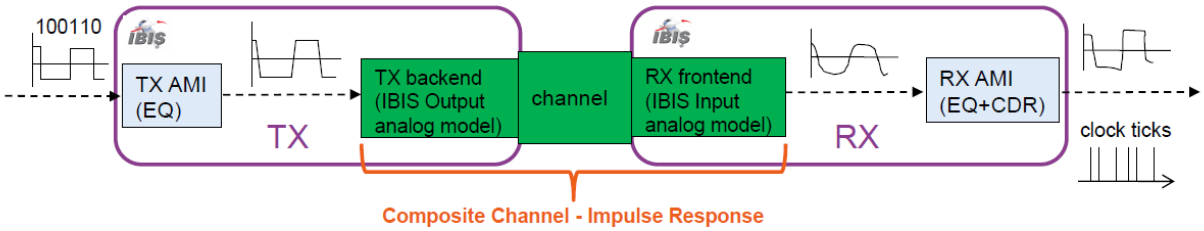

Figure 2.2: End to end simulation - block level

As data rates increase, there has been a growing need for mid-channel equalization devices. These can be classified as repeaters that just recondition the signal; and retimers that recondition and re-clock the signal. Mid channel devices break the linearity assumption of the channel as they are often non-linear devices. The IBIS Open Forum has a BIRD, Buffer Issue Resolution Document, which addresses this. Basically, if there are N mid-channel devices present, the channel is broken to N+1 LTI segments. Figure 2.3a shows an end to end system where N=1, Figures 2.3b-3c demonstrate how a mid-channel device is added.

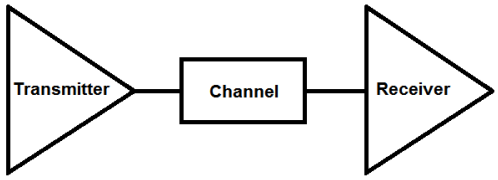

Figure 2.3a: End to End system

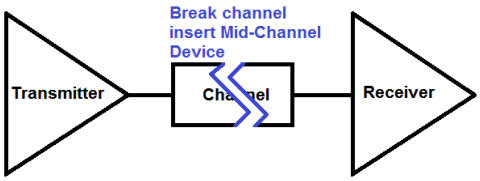

Figure 2.3b: End to End System - insert Mid Channel

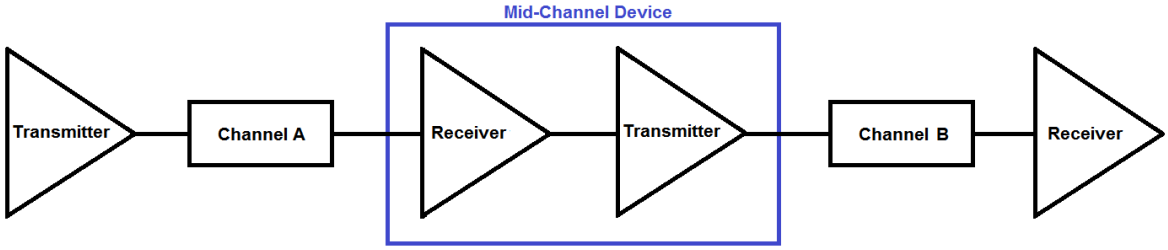

Figure 2.3c: End to End System with Mid-channel device

To insert a mid-channel device, the channel in Figure 2.3a is broken in two channels and one mid-channel device is inserted in the middle. The resultant is Figure 2.3c, where the channel is now broken into two channels: channel A and channel B.

The linearity assumption holds for channel A and channel B simulations, and each segment is simulated separately, with the result of the first segment, containing channel A, acting as the stimulus for the second segment.

With the fast simulation speeds that can be achieved with channel simulators, designers can explore a very large solution space comprising various AMI model parameters and other system variables. This enables them to have a much better understanding of how their system will perform under different conditions.

3. IBIS-AMI Model Development Challenges

3.1 Architecture development vs. AMI model generation

AMI models are typically generated after the architecture development phase, improved over the design cycles and finalized based on silicon results. AMI models are not simple architecture models and have to match the circuit implementation details in order to accurately reflect the behavior of the silicon across process, voltage and temperature variations (PVTs). For example, in a typical SerDes system shown in Figure 3.1, due to PVT variations, one could have multiple different equalization responses for a given circuit and setting. AMI model developers need to find a convenient approach to include these variations into their models even if the architecture model is more or less ideal.

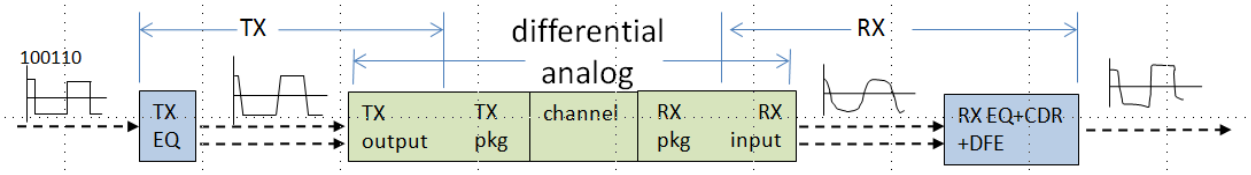

Figure 3.1: Typical SerDes system

3.2 Circuit vs. IBIS-AMI model representation

One of the challenges for IBIS-AMI modeling is to find the best approach to model the behaviors of various analog and digital circuits. For sophisticated SerDes systems, there are many circuits such as continuous time linear equalizer (CTLE), feed-forward equalizer (FFE), and finite impulse response filter (FIR) that can be treated as LTI and can be represented both in the time domain with their impulse or step response or in the frequency domain with a transfer function, S-parameters, or zero/pole model. Frequency domain representations must be converted into the time domain during simulation and may lose information if the model is overly simplistic or if any interpolation is needed, however, through block-by-block fast Fourier transformation (FFT) and inverse FFT (IFFT), simulation speed can be greatly improved.

For digital adaptation circuits, such as clock and data recovery (CDR) circuits, an algorithmic modeling approach should be taken. These circuits are time varying and can only be modeled in the time domain, same as nonlinear circuits.

3.3 AMI model abstraction

A key channel simulator property to keep in mind is that it considers that the entire analog content between the TX AMI portion and the RX AMI portion is LTI. Thus, referring to Figure 3.1, as an LTI system the entire differential analog section in the SerDes system is accurately represented by its impulse response. This key concept enables the channel simulator to achieve its fast simulation speeds.

Three challenging decisions need to be made when converting TX and RX circuits into their IBIS-AMI representations:

1) Partition the TX and RX designs into portions that can be modeled into IBIS and those that have to be modeled into AMI
2) Partition the TX and RX design into a signal flow and a control flow
3) Determine parameters for the control flow

3.3.1 Partition the TX and RX designs into IBIS portion and AMI portion

Each IBIS-AMI model has an IBIS portion and an AMI portion. Though high speed mixed signal circuit designers may have IBIS analog representations for their TX output and RX input buffers, IBIS-AMI modeling requires rethinking the IBIS representation in context with the channel simulator and in context with what is to be modeled within the AMI portion of the IBIS-AMI model.

For example, a TX IBIS analog output buffer model may have content defining C_comp values, Pulldown/Pullup Voltage-Current (VI) tables defining nonlinear resistance, Rising Waveform/Falling Waveform Voltage-Time (VT) tables, Ramp rising/falling rate of voltage change versus time (dV/dt) values, as well other important analog functions. An RX IBIS analog input buffer model may have content defining C_comp values, GND_clamp/POWER_clamp Voltage-Current (VI) tables defining nonlinear resistance, as well other important analog functions.

Should all or some of this IBIS analog content be used in the IBIS-AMI model?

Since IBIS VI tables are 'absorbed' by the channel simulator into its impulse model as part of the entire analog content, this means that the channel simulator inherently linearizes any impedance nonlinearity in the VI tables. The IBIS standard does not define how the channel simulator is to treat VI table impedance nonlinearity. One can expect each channel simulator to be somewhat different. This is part of the cost of the channel simulator simplifying assumptions that gives it its greater simulation speed. Any IBIS analog VT table inherently contains a filtering functionality that must be carefully used so as to not duplicate any filtering already included in the AMI portion.

Oftentimes, one can start off the development of the IBIS-AMI model by simplifying the VI tables to represent a linear 50 ohm impedance and eliminating any VT table. Then one can include detail in the VI tables and VT tables as needed.

3.3.2 Partition the TX and RX designs into a signal flow and a control flow

Partitioning the TX and RX designs into a signal flow can be a challenging task. For the purposes of this discussion, consider that the TX design contains an LTI FFE and that the RX design contains a CTLE, CDR and DFE. The content in Figure 3.1 above can be restructured in context with the TX and RX IBIS-AMI models as shown in Figure 3.2.

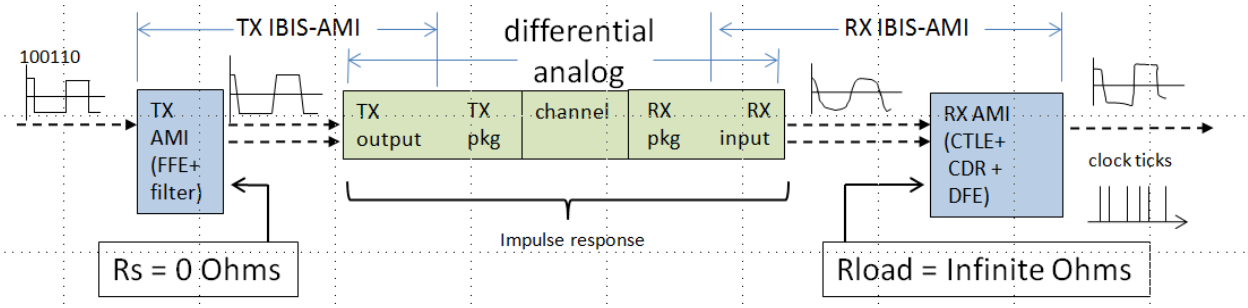

Figure 3.2: Typical SerDes system with IBIS-AMI models

Recognizing the signal flow vs. the control flow helps with the modeling procedure. For example, the CTLE itself is on the signal flow but its settings can be on a control flow through CTLE adaptation. Separating the two and modeling them in two different modules makes the model more modular and easier to maintain and debug as the two are typically running at different speeds and have a different modeling approach.

3.3.3 Determine parameters for control flows

Providing the IBIS-AMI models with parameters that represent the TX and RX circuit control flow requires careful consideration. The circuit controls define a set of operating conditions for the circuit.

For example, for a TX AMI model, operating conditions might be defined for setting gain values, pole values, FFE tap value, PVTs, etc. For an RX AMI model, another set of operating conditions might be defined. Aligning the IBIS-AMI model parameters and their values with the circuit control nomenclature keeps the model usage consistent with the actual circuit hardware.

3.4 AMI model development

High speed mixed signal IC designers and architects are not programmers. They are better equipped at dealing with various circuit design or algorithm developments. However, IBIS-AMI model development requires much different skills:

- skill in C/C++ coding
- skill in creating code that is compatible with both Windows and Linux
- skill in compiling/linking on Windows and Linux
- skill with the IBIS-AMI standard
- skill on parsing IBIS files

Oftentimes, the circuit designers or architects become AMI model developers and are tasked to pull together disjoint pieces of their IC designs from incomplete IC specifications, from Spice simulations and from cryptic MATLAB/C++ code. These disjoint pieces may lead to incomplete AMI models. Oftentimes, the challenge to convert and use algorithm design code leads to AMI models that are not IBIS spec compliant.

3.5 AMI model compilation

As defined in the IBIS standard, the AMI models are based on C/C++ code and are provided for use in a channel simulator as an executable Dynamic Link Library (DLL) file for 32 bit and 64 bit Windows-based PCs or a Shared Object (SO) file for 64 bit Linux systems. The channel simulator interfaces with the AMI model by calling the AMI model AMI_Init(), AMI_GetWave() and AMI_Close() interface functions as defined in the IBIS standard.

For IBIS-AMI developers who only want to focus on the coding processes, conforming to the IBIS standard and generating various models for different platforms and operating systems without any kind of automation can be tedious and challenging.

3.6 AMI model debug

Debugging compiled AMI models is usually very challenging and requires advanced debugging skills, especially when there are not many debug hooks available in the simulation platform to

allow users to step through the simulation or to observe intermediate results. It is always desired to be able to debug new algorithmic code within a model development platform, independently from use in any channel simulator.

4. Proposed IBIS AMI Development Flow

4.1 Model hierarchy partition

Following the discussion on the challenges involved in IBIS-AMI modeling, a proposed IBIS-AMI development flow is presented in this section. Before the detailed modeling work is started, the first thing an IBIS-AMI model developer needs to do is to have a good understanding of the system to be modeled and wisely partition the design into a hierarchical structure. One needs to determine the level of abstraction and how many individual components are to be modeled. In addition, debugging hooks need to be embedded throughout the model. Furthermore, since there are a lot of reiterative processes expected during the process of model building, compilation and debugging, one needs to maximize the usage of automation scripts.

Based on a good understanding of the system to be modeled, one then needs to decide which circuits can be modeled in the IBIS portion and which in the AMI portion. For linear analog circuits whose behavior does not change over time, one can choose to model in the .ibs file using VI tables, C_Comp and other IBIS linear analog functions. For nonlinear analog circuits, even though their behavior can also be described through different VI tables in the .ibs file, channel simulation tools from different EDA vendors might have different interpretations when combining the nonlinear behavior with the linear channel impulse response. As a result, one could see slightly different simulation results using different simulation tools with the same IBIS-AMI model.

To achieve consistent results from different EDA platforms, one approach is to include all nonlinear circuits in the AMI portion of the model. For any linear circuit that has a nonlinear circuit between itself and the pad, it is best to be included in the AMI model. In addition, for linear analog circuits that are controlled by adaptive digital signals, it is convenient to treat them as time varying system and put them in the AMI model. Please note that in this section, the definition of ‘analog’ vs. ‘digital’ is from the circuit design perspective so that circuits modeled in the AMI portion are not necessarily considered as digital circuits.

As in circuit schematics, IBIS-AMI model should have a hierarchical structure. The exact hierarchy does not need to 100% match the circuits, but a closely matched hierarchy is easier to follow and debug. It is also convenient to reuse sub-blocks as do circuit designers. A completely flattened model is not recommended for complicated mixed-signal systems. Typically, 2 to 3 levels of hierarchy strikes a good balance.

A mixed signal system includes an analog domain and a digital domain. Understanding the boundary is as important as understanding the function of each circuit block. For digital domain blocks, hierarchy can be more flat and can be divided up simply based on functionality. For analog domain blocks, it is more convenient to have multi-level hierarchies. In addition, one

needs to clearly understand the difference between the actual signal flow path versus the control path. The two flow paths are typically running at different speed and should be modeled separately.

4.2 Modularization and component reuse

Once the domain and hierarchies are determined, the model for each individual component can be independently created. For functionally similar components, one choice is to create a generic model that can cover all the functions and thus different components can just be different instantiations of the same model with different modes. The advantage of doing so is to assure lower coding maintenance cost, although programming efficiency could be sacrificed. For example, in the case of a common function that needs to be updated; this method reduces the amount of coding change and the corresponding verification workload which leads to less chance of making errors.

For analog circuits, each component model needs to be verified independently and correlated with the corresponding circuit behavior before it is used in the system model. This is a typical bottom up approach. Alternatively, a simple system model can be built and verified first with additional models added incrementally or with existing models enriched. This is a top-down approach. Sometimes a mix of the two approaches can be used for modeling a complicated system with multiple hierarchies. For example, a SerDes receiver can take a top-down approach that includes CTLE, summer, and DFE. The internal CTLE can take a bottom-up approach.

4.3 Debugging hooks

Debugging an IBIS-AMI model is not a trivial task. It is important to include debugging hooks in the model from the very beginning so that signals can be easily probed and debugged. Typical hooks include direct signal output, bypass switches, forcing a control signal at fixed values, and data dumping into a file. It is often convenient to be able to verify the model before compilation or before the model is loaded into the EDA tools. Some AMI development platforms enable users to simulate the model before it is being compiled. Having debug hooks specific for these platforms are usually very helpful. It is also possible to use the compilation tool to debug the compiled DLL/SO files by attaching the source code to the EDA tool or the AMI development tool that executes the DLL/SO, but the debugging procedure can be more involved.

4.4 Scripting and automation

There are many places along the AMI model generation flow that can utilize scripting automation. Once the core code is developed, a wrapper should be available to automatically generate the AMI_Init(), AMI_GetWave(), and AMI_Close() procedures before code compilation. The script should also be able to generate the .ami file automatically from the parameter list that is defined in the model. This is very useful as the core code requires frequent updating during the AMI model development phase; it becomes tedious and error-prone to update the corresponding procedures or .ami files every time a compilation is needed or AMI

parameters are changed. Having the automation scripts available not only guarantees that the compilation and build process is robust, but also that the generated code is IBIS compliant and readily interfaced with channel simulation tools. This allows the AMI developer to focus on the core C++ code development.

5. Engineering Trade-offs

5.1 Accuracy vs. speed

Inherently, it is acknowledged that the use of IBIS-AMI models in a channel simulator results in an engineering trade-off with decreased accuracy for increased simulation speed. The highest software modeling accuracy is with the representation of hardware as a circuit design used in a transient simulator. However, for modeling and simulating SerDes systems, it has been determined in the market that a transient simulation is simply too slow to give practical and usable results within a reasonable amount of simulation time.

When performing IBIS-AMI models accuracy vs. speed trade-offs, it is important to keep the customer in mind and limit the development investment cost to achieve just enough accuracy that would be acceptable to the customer.

5.2 Custom models vs. library models

There are existing tools on the market that can help AMI model developers to partially automate the process of IBIS-AMI model generation. Oftentimes, these tools provide a rich library of DSP blocks that may be used in an AMI model and provide C++ model coding templates that keeps one focused on scientific coding (a skill all engineers have) with minimal skill in detail C++ knowledge.

Use of the library models can obviously save development time. These library models are oftentimes vigorously verified and have less chances of having coding bugs. One can quickly build an initial model with the library models for initial proof of concepts.

As higher level of accuracy and more programming flexibility are needed, custom models are always preferred or required. But with more custom models in the IBIS-AMI model, the verification work increases significantly as not only each individual model needs to be fully tested, the interactions between them sometimes also require thorough testing.

5.3 Hierarchy levels

A key concept in implementing a new AMI model is to think hierarchically. In any schematic based simulation tool it is common to define a top level schematic and within it place instances of other sub-network designs. A schematic based design inherently is hierarchical with sub-networks nested within sub-networks.

Similar to schematic hierarchy, one needs to decide IBIS-AMI model hierarchy, so that lower level modules can have multiple instances within higher levels with each instance having its own parameters. This adds a great coding flexibility and easy maintenance of the model. In addition, one can easily explore engineering trade-offs as one can conveniently switch between low accuracy models and high accuracy models for lower level modules.

6. An IBIS-AMI Model Development Example

In this section an IBIS-AMI modeling example of a Xilinx designed 32 Gbps SerDes receiver at the 20nm technology node is presented. The detailed model generation flow is shown below.

6.1 How it is partitioned

In this example, all of analog circuits are modeled in the AMI and only an ideal termination is modeled in the IBIS. The benefit of doing that is that one can accurately model the nonlinear behavior of the analog circuits and maximize the consistency across the simulation platforms. The level of hierarchy is set to be 3, which shows an engineering trade-off between code reuse and hierarchy maintenance. The more hierarchical levels there are, the easier it is to reuse the code, but the more difficult it is to manage and verify the entire model. At the top level, there are 6 modules, shown in Figure 6.1:

- One module for the receiver analog front end (AFE) block to model the RX pad and termination related impairments;
- One module for the CTLE;
- One module for the summers;
- One module for DFE;
- One module for clock generation and clock recovery;
- One module for digital adaptation blocks.

The last module in the digital domain is modeled directly from register transfer level (RTL) or architecture model. Its hierarchy is partitioned according to its functionality. The rest of the modules are in the analog domain and are modeled after the circuit implementation. Among them, AFE and clock generation modules have just one hierarchy level down. CTLE, DFE, and summer modules contain many instances of the same circuits and there is one more level of hierarchy for these modules.

6.2 What the hierarchy is

To start with, a top-down approach is used. A simple low pass filter is initially used for the AFE. Extracted zero-pole models are used for CTLE. An ideal clock source is used to generate the sampling clocks needed. Dummy adaptation blocks are placed where fixed values instead of the

adapted values are used in the initial model. Debugging hooks are inside the model so that each stage of the CTLE can be independently bypassed and probed.

On the other hand, inside the summer module, a bottom-up approach is adopted. First, each lower level circuit is modeled based on its function. For example, the summing node, clocked slicers and latches are modeled and verified individually before they are put together to form the summer module. For a half-clocked architecture (even/odd paths), one only need to model a half branch and then instantiate it twice with two different clock inputs. For the DFE module, a similar approach and modeling procedure is used.

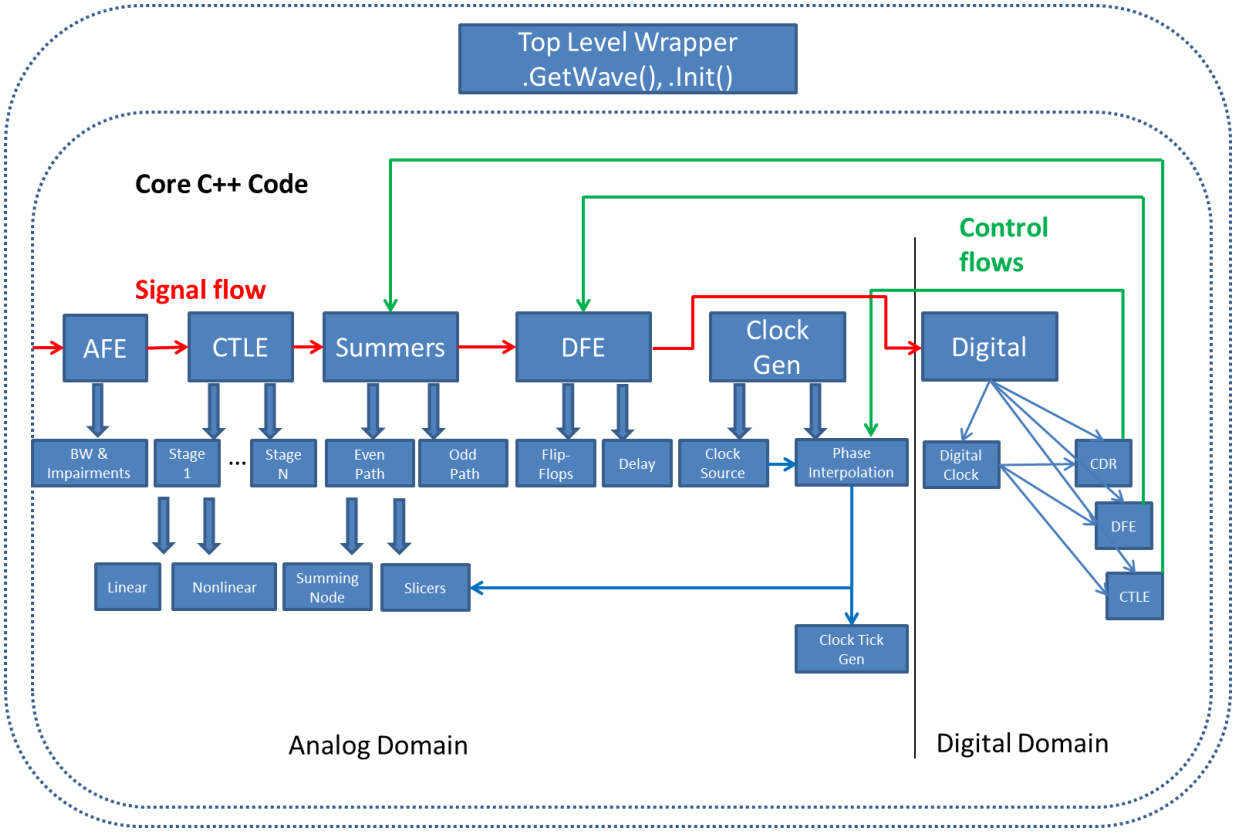

Figure 6.1: Example IBIS-AMI hierarchy and partition of a 20 nm, 28Gbps SerDes

After this initial model is tested and verified, it is gradually enhanced. A more complicated termination model is used to accurately capture the non-ideal termination and limited front end bandwidth. For the CTLE, unique and accurate frequency responses are modeled up to 40GHz for each available setting and application mode across silicon PVT corners. Analog nonlinear behavior is also captured and modeled for each setting and application mode across PVT. To speed up simulation, a block-by-block computation is adopted within this model. Time domain data are converted into frequency domain through FFT before they are reshaped by the CTLE transfer function. The processed data are then converted back into time domain. A conventional bit-by-bit convolution approach is also available for this model if an accurate CTLE adaptation process needs to be monitored. To increase the efficiency of the memory usage, dynamic

memory allocation is used throughout the code. (Some of the EDA tools do not allow AMI models to occupy a lot of stack memory. EDA simulation may crash without warning if too much stack memory is used).

DFE path delay is modeled for each critical timing path across PVT. Tap quantization effect is also included. Each tap can be individually bypassed for the debugging purpose. Mismatch impairments are modeled and added in the clock generation block to match circuit simulation. Clock tick generation block is also inserted for channel simulators to align output waveforms.

Inside the digital module, one common digital clock model is shared among various adaptation blocks for adaptation synchronization. All adaptation blocks are functionally independent and modeled separately. For example, there is one model for clock data recovery (CDR), one model for DFE adaptation, one model for CTLE adaptation, and so on. All adaptations can be overridden manually or initialized to a predefined value. The adaptation process can be optionally saved into files for debugging or for behavior study. All adaptations are verified and correlated with the architecture model initially. At later stages, they are also correlated with silicon.

Detailed domain and hierarchy partition is illustrated in Figure 6.1.

6.3 Debug hooks and development tools

In this example, Keysight's SystemVue is used as the platform for AMI model development and generation. One can use its graphical interface to partition the system into different hierarchies and to modularize the model. Further, one can use its built-in library models for some of the analog circuit components such as delay buffer to save code development cycles. One can setup jitter as desired and insert s-parameter files in a test bench. As a good debugging feature, a free emulated digital communication analyzer (DCA) from Keysight can be used for waveform probing at simulation in SystemVue before the model is compiled. This DCA tool is the same as used in Keysight oscilloscopes.

Microsoft's Visual Studio is used for C++ code development and debug. This code debugging environment allows the user to debug code while running in the development environment simulator. It supports setting break points within the code, stepping through the code as it is running, inspecting the values and states for various variables, resetting the values and states and more. Since the algorithmic model code is based on C++, the C++ compiler and linker inherently catches many coding errors such as wrong array handling, wrong memory management, wrong data type usage and more compared to standard C coding. With this rich AMI model debugging environment from SystemVue and Visual Studio, one can reliably debug and fix most algorithmic and coding problems encountered and confidently use the exported AMI model in any channel simulator with no debugging in the channel simulator required.

All modules discussed and modeled above are first compiled into one DLL file and then imported into SystemVue as a custom library. Then these custom models are properly connected

hierarchically to form the system level model. After testing the system level model in various test benches, one can use SystemVue to automate the top level C++ model generation, in which process AMI_GetWave() and AMI_Init() are created and essentially wrapped around the core C++ code. At the same time, an .ami file is generated to reflect the model parameter list. A dummy .ibs file is also readily available from automation. One can later modified this .ibs file to include additional behavior information that is not modeled inside the C++ model. Finally, one can compile the top level C++ model using Visual Studio to generate the .dll file that is defined in the .ibs file. A complete Windows IBIS-AMI receiver model that includes one .dll file for 64-bit Windows, one .dll file for 32-bit Windows, one .ami file and one .ibs file is then finished. The detailed modeling flow is shown in Figure 6.2.

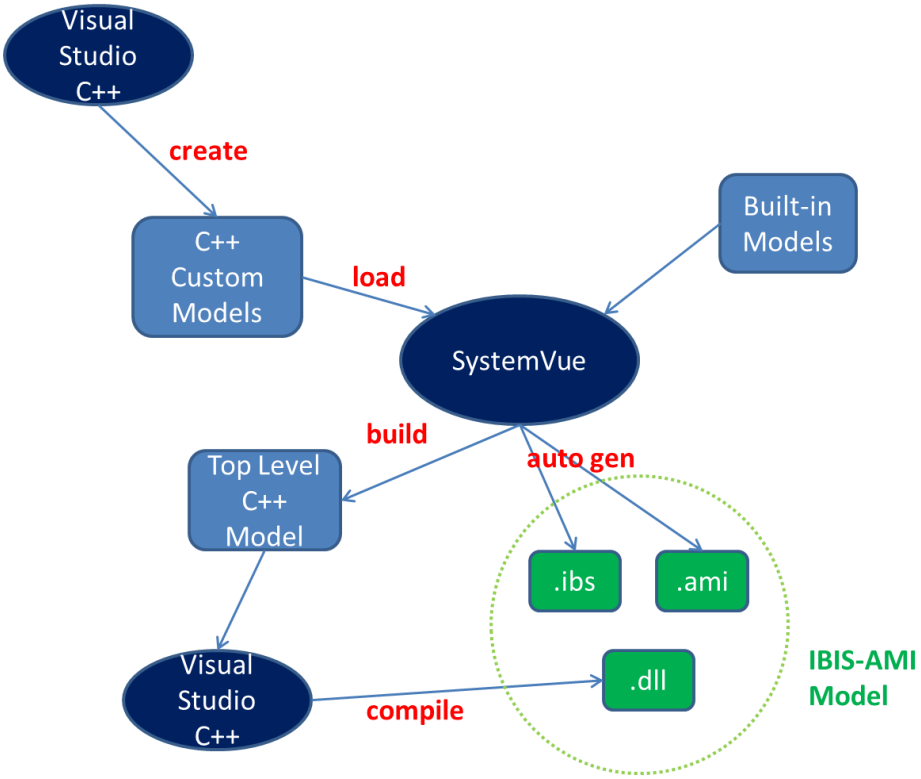

Figure 6.2: Modeling flow example

In addition, a configuration shell script is automatically created during the top level C++ code generation process. This script can be run on Linux to generate the make file that one can use for Linux .so file generation. The .ami file and .ibs file for Linux are the same as for Windows. With the steps described above, a Linux version IBIS-AMI receiver model can be created based on the existing C++ code with only a little amount of extra effort.

The same development flow is used for the TX IBIS-AMI model generation, although the modeling procedure is much less complicated and only one level of hierarchy is needed. The control flow for TX FIR coefficients is not adaptive and is directly set through the AMI model parameters. The TX analog driver is modeled similarly as the RX CTLE and has both linear and nonlinear models.

7. Results Verification and Correlation

7.1 Results

Using the proposed IBIS-AMI generation flow, the complete model was created within 2 months – well ahead of the planned schedule, and its simulation time is approximately 6 minutes per million bits, improved 5 times compared with models generated from legacy flows.

Figure 7.1 shows the test bench in the Keysight Advanced Design System (ADS) Channel Simulation tool used for IBIS-AMI model verification.

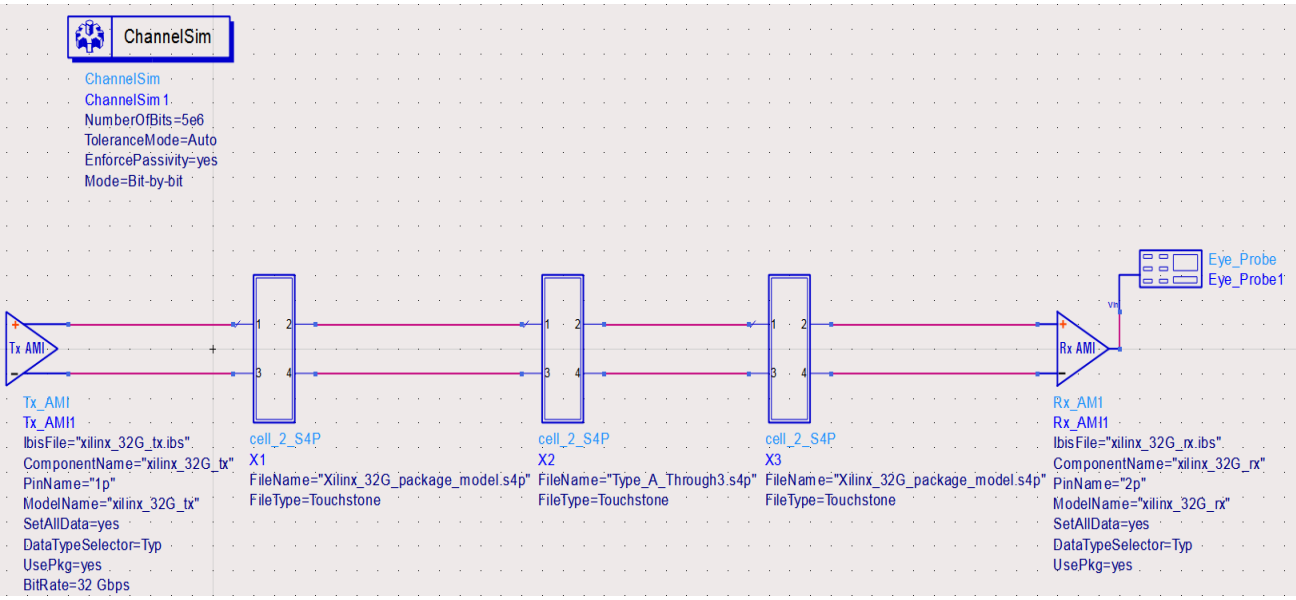

Figure 7.1: ADS test bench for IBIS-AMI model verification

Two simulation results are presented here as examples. Figure 7.2 shows a ~22dB loss channel (PCB trace + package) at 16GHz. Figure 7.3 shows a ~30dB loss channel at 14 GHz (PCB trace + package). 5 million bits are simulated in each case at 32 Gbps and 28 Gbps, respectively.

Figure 7.4 shows the eye plot and BER contours for the 22dB loss channel at 32 Gbps with an eye opening of more than +/-75 mV at BER=1E-18. Figure 7.5 shows the eye plot and BER contours for the 30dB loss channel at 28 Gbps with an eye opening of more than +/-40mV at BER=1E-18.

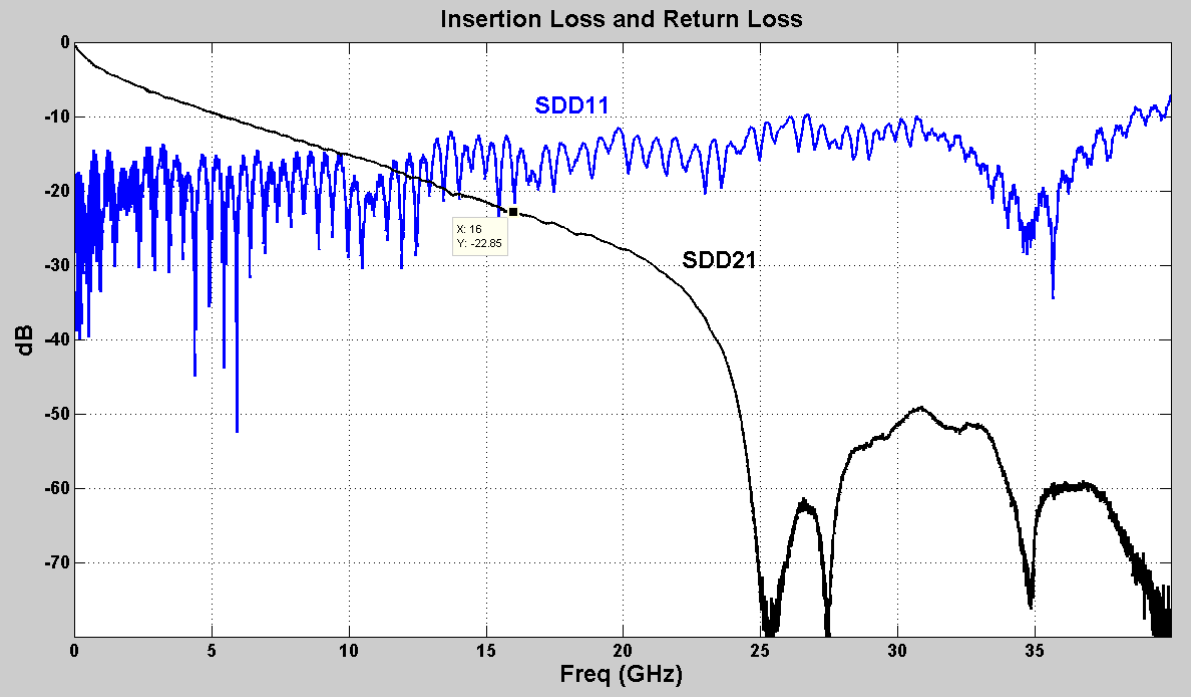

Figure 7.2: S-parameters for a ~22dB loss channel (including package loss) at 16GHz

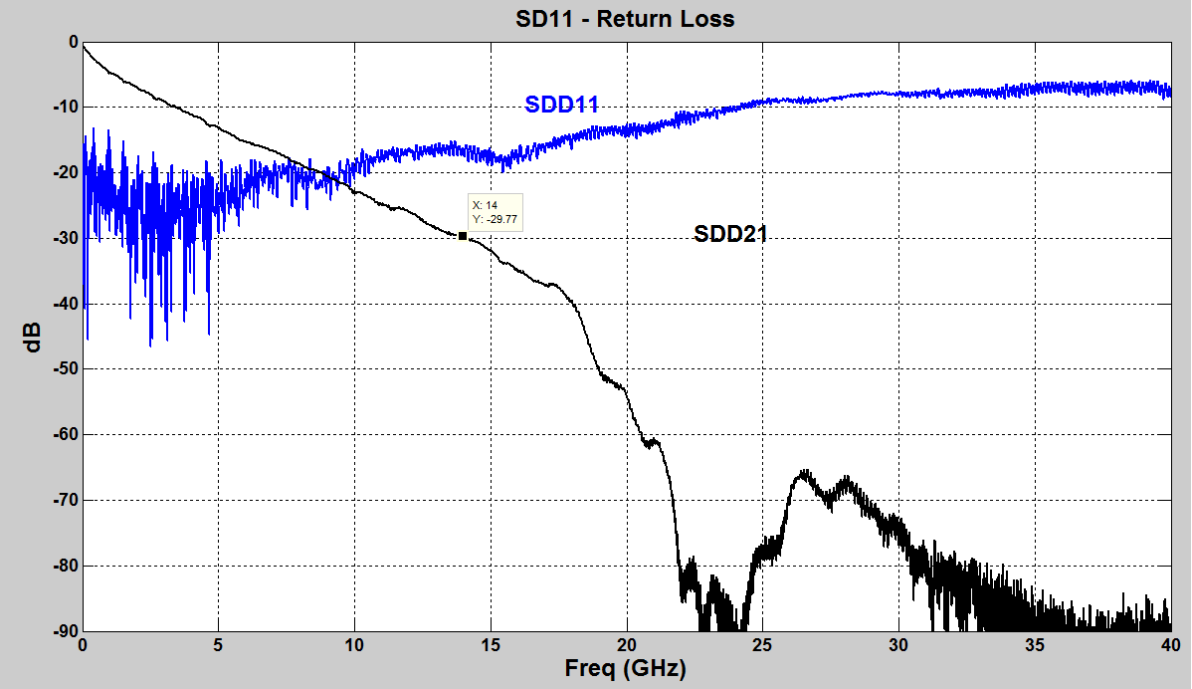

Figure 7.3: S-parameters for a ~30dB loss channel (including package loss) at 14 GHz

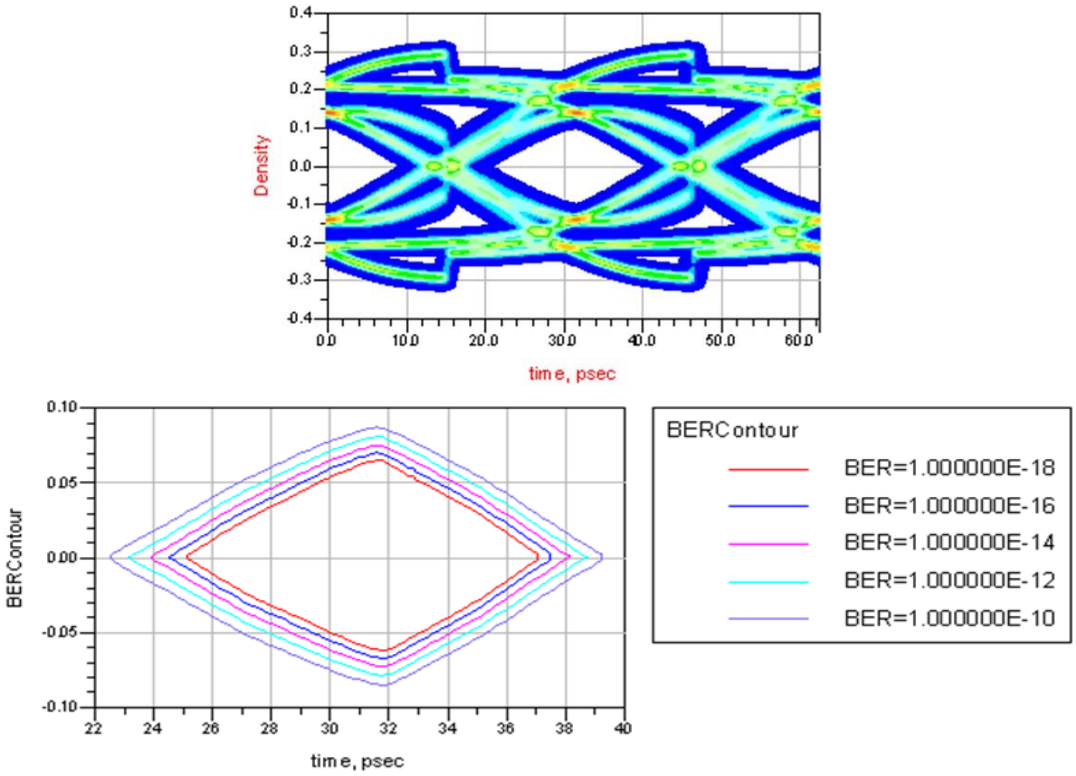

Figure 7.4: Eye plot and BER contour, at 32 Gbps for the 22 dB loss channel in Figure 7.2

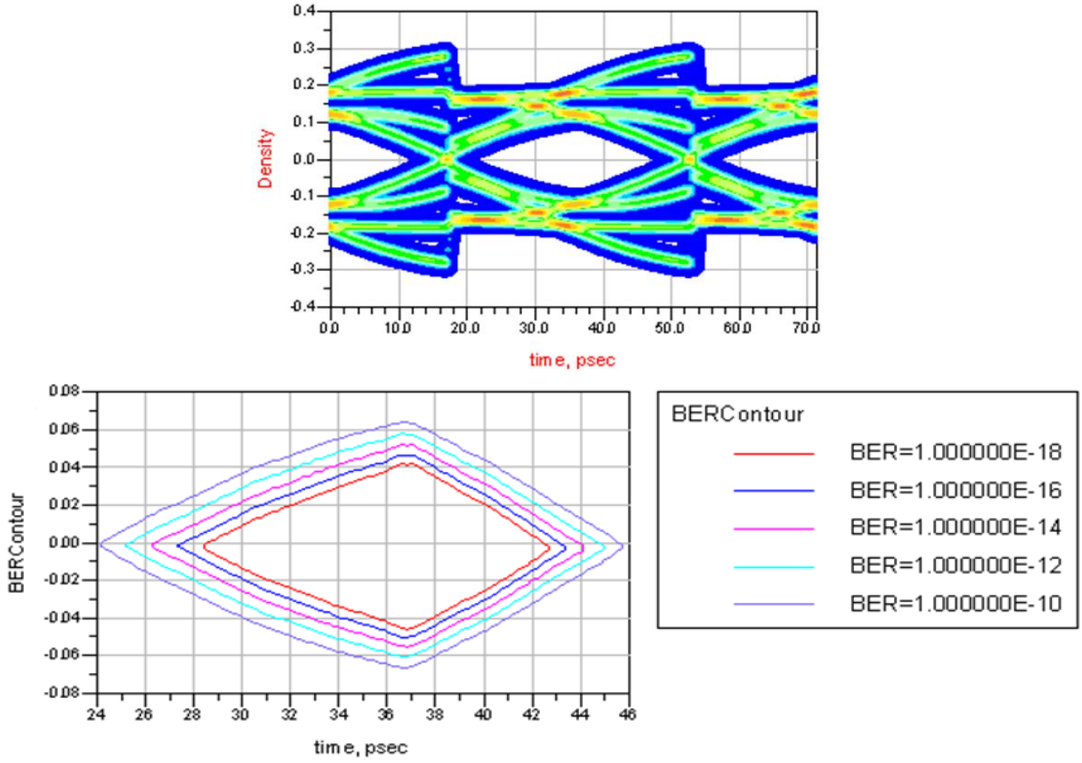

Figure 7.5: Eye plot and BER contour, at 28 Gbps for the 30dB loss channel in Figure 7.4

7.2 Verification and correlation

This model was verified in ADS, SiSoft Quantum Channel Designer (QCD) and multiple EDA platforms on both Linux and Windows. General matching has been achieved between different platforms with the design. TX output waveforms and RX internal waveforms before the slicers are correlated with the HSPICE simulation and achieved >20dB matching (averaged signal-to-delta ratio) across PVT corners. RX adaptation has been correlated with the algorithmic models from Mathworks Simulink. For future work, the IBIS-AMI model will be correlated and matched with the silicon data for TX output waveforms, RX adaptations and the predicted bit error ratio (BER).

8. Conclusions

This paper discussed the various challenges in IBIS-AMI model generation in terms of architecture partitioning, circuit abstraction, model building, model compilation and debugging. A practical IBIS-AMI model development flow was proposed using hierarchical partitioning, modularization, debugging hooks, scripting and automation. Engineering trade-offs such as accuracy vs. speed, custom models vs. library models and hierarchical levels are highlighted. The proposed model development flow was applied in a 20nm 32Gbps SerDes IBIS-AMI development. Using this flow, a high quality IBIS-AMI model was delivered in two months for 32-bit Windows system, 64-bit Windows system and 64-bit Linux system. The model runs 5 times faster than models generated from a legacy flow and achieves an excellent correlation with the design.

References

[1] “I/O Buffer Information Specification”, Version 5.1, IBIS Open Forum

[2] Kian Haur (Alfred) Chong, et al, “IBIS AMI Modeling of Retimer and Performance Analysis of Retimer based Active Serial Links”, DesignCon 2014

[3] Jason Liu, et al, “Channel Simulation Platform Creation in Matlab and IBIS-AMI Simulation Verification”, Asia IBIS Summit, Shanghai, China, November 2012

[4] Bob Sullivan, et al, “Simulating High-Speed Serial Channels with IBIS-AMI Models”, Application Note 5990-9111EN, Keysight Technologies, November 15, 2011

[5] Syed B. Hug, “An Introduction to IBIS Modeling”, Application Note 1111, Texas Instruments, June 1998