



February 2011
Channel Simulation

© **Agilent Technologies, Inc. 2000-2011**

5301 Stevens Creek Blvd., Santa Clara, CA 95052 USA

No part of this documentation may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

Acknowledgments

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries. Mentor products and processes are registered trademarks of Mentor Graphics Corporation. * Calibre is a trademark of Mentor Graphics Corporation in the US and other countries. "Microsoft®, Windows®, MS Windows®, Windows NT®, Windows 2000® and Windows Internet Explorer® are U.S. registered trademarks of Microsoft Corporation. Pentium® is a U.S. registered trademark of Intel Corporation. PostScript® and Acrobat® are trademarks of Adobe Systems Incorporated. UNIX® is a registered trademark of the Open Group. Oracle and Java and registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. SystemC® is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission. MATLAB® is a U.S. registered trademark of The Math Works, Inc.. HiSIM2 source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code in its entirety, is owned by Hiroshima University and STARC. FLEXIm is a trademark of Globetrotter Software, Incorporated. Layout Boolean Engine by Klaas Holwerda, v1.7 <http://www.xs4all.nl/~kholwerd/bool.html> . FreeType Project, Copyright (c) 1996-1999 by David Turner, Robert Wilhelm, and Werner Lemberg. QuestAgent search engine (c) 2000-2002, JObjects. Motif is a trademark of the Open Software Foundation. Netscape is a trademark of Netscape Communications Corporation. Netscape Portable Runtime (NSPR), Copyright (c) 1998-2003 The Mozilla Organization. A copy of the Mozilla Public License is at <http://www.mozilla.org/MPL/> . FFTW, The Fastest Fourier Transform in the West, Copyright (c) 1997-1999 Massachusetts Institute of Technology. All rights reserved.

The following third-party libraries are used by the NlogN Momentum solver:

"This program includes Metis 4.0, Copyright © 1998, Regents of the University of Minnesota", <http://www.cs.umn.edu/~metis> , METIS was written by George Karypis (karypis@cs.umn.edu).

Intel@ Math Kernel Library, <http://www.intel.com/software/products/mkl>

SuperLU_MT version 2.0 - Copyright © 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy). All rights reserved. SuperLU Disclaimer: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7-zip - 7-Zip Copyright: Copyright (C) 1999-2009 Igor Pavlov. Licenses for files are: 7z.dll: GNU LGPL + unRAR restriction, All other files: GNU LGPL. 7-zip License: This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. unRAR copyright: The decompression engine for RAR archives was developed using source code of unRAR program. All copyrights to original unRAR code are owned by Alexander Roshal. unRAR License: The unRAR sources cannot be used to re-create the RAR compression algorithm, which is proprietary. Distribution of modified unRAR sources in separate form or as a part of other software is permitted, provided that it is clearly stated in the documentation and source comments that the code may not be used to develop a RAR (WinRAR) compatible archiver. 7-zip Availability: <http://www.7-zip.org/>

AMD Version 2.2 - AMD Notice: The AMD code was modified. Used by permission. AMD copyright: AMD Version 2.2, Copyright © 2007 by Timothy A. Davis, Patrick R. Amestoy, and Iain S. Duff. All Rights Reserved. AMD License: Your use or distribution of AMD or any modified version of AMD implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. AMD Availability: <http://www.cise.ufl.edu/research/sparse/amd>

UMFPACK 5.0.2 - UMFPACK Notice: The UMFPACK code was modified. Used by permission. UMFPACK Copyright: UMFPACK Copyright © 1995-2006 by Timothy A. Davis. All Rights Reserved. UMFPACK License: Your use or distribution of UMFPACK or any modified version of UMFPACK implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at

your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. UMFPACK Availability: <http://www.cise.ufl.edu/research/sparse/umfpack> UMFPACK (including versions 2.2.1 and earlier, in FORTRAN) is available at <http://www.cise.ufl.edu/research/sparse> . MA38 is available in the Harwell Subroutine Library. This version of UMFPACK includes a modified form of COLAMD Version 2.0, originally released on Jan. 31, 2000, also available at <http://www.cise.ufl.edu/research/sparse> . COLAMD V2.0 is also incorporated as a built-in function in MATLAB version 6.1, by The MathWorks, Inc. <http://www.mathworks.com> . COLAMD V1.0 appears as a column-preordering in SuperLU (SuperLU is available at <http://www.netlib.org>). UMFPACK v4.0 is a built-in routine in MATLAB 6.5. UMFPACK v4.3 is a built-in routine in MATLAB 7.1.

Qt Version 4.6.3 - Qt Notice: The Qt code was modified. Used by permission. Qt copyright: Qt Version 4.6.3, Copyright (c) 2010 by Nokia Corporation. All Rights Reserved. Qt License: Your use or distribution of Qt or any modified version of Qt implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. Qt Availability: <http://www.qtsoftware.com/downloads> Patches Applied to Qt can be found in the installation at: \$HPEESOF_DIR/prod/licenses/thirdparty/qt/patches. You may also contact Brian Buchanan at Agilent Inc. at brian_buchanan@agilent.com for more information.

The HiSIM_HV source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code, is owned by Hiroshima University and/or STARC.

Errata The ADS product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsof" is now part of Agilent Technologies and is known as "Agilent EEsof". To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain "HP" or "HPEESOF" references.

Warranty The material contained in this document is provided "as is", and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this documentation and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

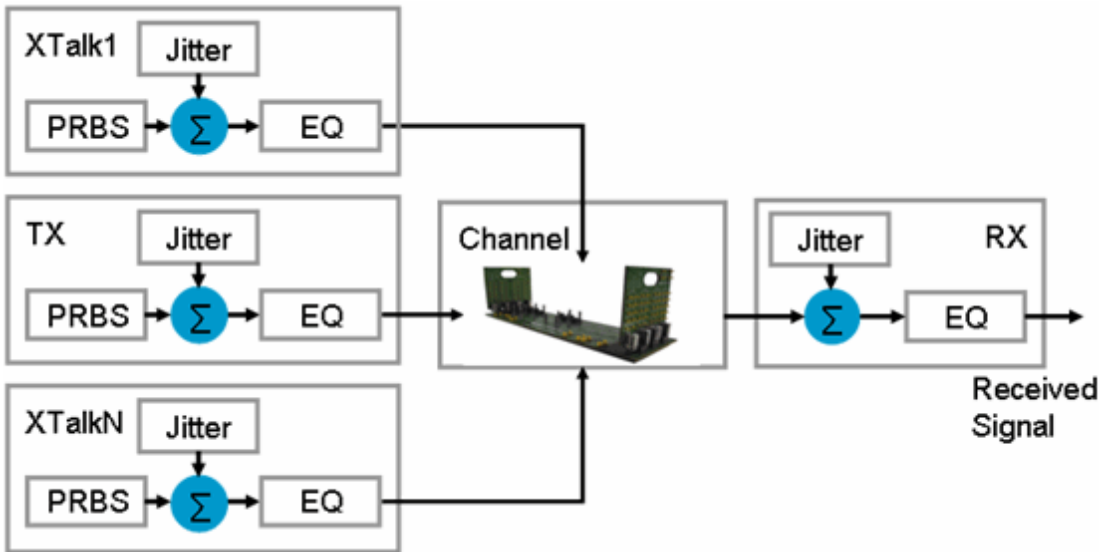
Technology Licenses The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license. Portions of this product include the SystemC software licensed under Open Source terms, which are available for download at <http://systemc.org/> . This software is redistributed by Agilent. The Contributors of the SystemC software provide this software "as is" and offer no warranty of any kind, express or implied, including without limitation warranties or conditions or title and non-infringement, and implied warranties or conditions merchantability and fitness for a particular purpose. Contributors shall not be liable for any damages of any kind including without limitation direct, indirect, special, incidental and consequential damages, such as lost profits. Any provisions that differ from this disclaimer are offered by Agilent only.

Restricted Rights Legend U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

About Channel Simulation	7
Parameters for Channel Simulation	8
Setting Up Analysis	8
Setting Up Convolution	8
Description of Channel Simulation	10
Bit-by-Bit Simulation	10
Statistical Simulation	12
Bit-by-bit vs. Statistical Simulations	12
Using Channel Simulation	14
Creating a Channel Simulation Design	14
Adding Equalization Effects	17
Adding Crosstalk Effects	19
Running Channel Simulations in Statistical Mode	20
Channel Simulation for AMI	22
Introduction to AMI	22
AMI Simulation Setup	23
AMI Remote Simulation	25
Examples of Channel Simulation	26
Simulating Various Designs Using Channel Simulation	26
Analyzing a PCIe Gen 2 System	26
Limitations of Channel Simulation	27
Troubleshooting Channel Simulation	28
Tx_SingleEnded (Channel Simulation Single-Ended Transmitter)	29
Tx_Diff (Channel Simulation Differential Transmitter)	35
Xtlk_SingleEnded (Channel Simulation Single-Ended Crosstalk Transmitter)	35
Xtlk_Diff (Channel Simulation Differential Crosstalk Transmitter)	40
Xtlk2_SingleEnded (Channel Simulation Single-Ended Crosstalk2 Transmitter)	41
Xtlk2_Diff (Channel Simulation Crosstalk2 Differential Transmitter)	45
Rx_SingleEnded (Channel Simulation Single-Ended Receiver)	46
Rx_Diff (Channel Simulation Differential Receiver)	55
Term_SingleEnded (Channel Simulation Single-Ended Termination)	56
Term_Diff (Channel Simulation Differential Termination)	57
Tx_AMI (IBIS-AMI Transmitter)	58
XtlkTx_AMI (IBIS-AMI Crosstalk Transmitter)	63
Rx_AMI (IBIS-AMI Receiver)	69
XtlkRx_AMI (IBIS-AMI Receiver)	72

About Channel Simulation

Channel simulation is designed for rapid signal integrity analysis of linear channels in high speed serial and parallel communication links. In typical scenarios, the ADS Channel Simulator is capable of processing million-bit patterns in approximately one minute, allowing for accurate analysis of eye diagram properties including density, width, height, bathtubs and BER contours. The channel simulator accounts for ISI, random jitter, crosstalk, encoding, equalization and other effects of interest to signal integrity designers. In addition to high-throughput simulations with long bit sequences, the Channel Simulator offers a statistical analysis mode for rapid calculation of statistical properties down to extremely low BERs. It is designed for system architectures of the following kind:



The channel is an arbitrary, linear, hierarchical ADS circuit. It typically consists of lumped RLC elements, Touchstone S-parameter files, transmission lines, multi-layer library models, HSPIICE netlists, W-elements, and similar circuit elements.

TX represents the transmitter, encompassing functionality such as PRBS generation, encoding, emphasis, and jitter injection. The ADS Channel Simulator includes dedicated TX models in single-ended and differential configurations.

XTLK 1 and XTLK N represent crosstalk transmitters. Any number of crosstalk transmitters may be included in a simulation. The Channel Simulator models both synchronous and random crosstalk, as discussed in *Xtlk2_SingleEnded (Channel Simulation Single-Ended Crosstalk2 Transmitter)* (cktsimchan).

RX models a receiver. Like the TX model, RX accounts for the effects of receive jitter, and includes models of FFE, DFE and CTLE equalizers, including fast algorithms for optimal tap coefficient calculation and adaptive filtering.

In ADS, the Channel Simulation controller, named *ChannelSim*, and its related components are available on the Simulation-ChannelSim palette. Channel Simulation uses the same license as *Transient and Convolution Simulation* (cktsimtrans).

Parameters for Channel Simulation

Channel Simulation parameters are accessible through the ChannelSim controller which is available on the Simulation-ChannelSim palette. They enable you to define aspects of the simulation listed in the following table:

Tab Name	Description	For details, see...
Analysis	Parameters that determine the analysis mode: Bit-by-Bit and Statistical	Setting Up Analysis
Convolution	Parameters that control the two phases of channel simulation: step response characterization and pulse superposition.	Setting Up Convolution
Display	Control the visibility of simulation parameters on the schematic.	<i>Displaying Simulation Parameters on the Schematic</i> (cktsim)

Setting Up Analysis

Analysis parameters determine the analysis mode, either Bit-by-bit or Statistical. In Bit-by-bit mode, the actual sequence specified in the Tx and Xtlk drivers is applied to the system for the duration specified in the Analysis tab. In Statistical mode, system properties are derived by statistical calculations and not by direct application of a bit sequence. In the limit as the length of the bit sequence becomes infinite, Statistical and Bit-by-bit simulations converge to the same results. For more information, refer to *Description of Channel Simulation* (cktsimchan).

Setup Dialog Name	Parameter Name	Description	Default
Bit-by-bit	Mode= <i>Bit-by-bit</i>	Choose Bit-by-bit analysis mode	Yes
Statistical	Mode= <i>Statistical</i>	Choose Statistical analysis mode	No
Number of bits	NumberOfBits	Number of bits to simulate in bit-by-bit mode	1000

Setting Up Convolution

Convolution parameters control the two phases of channel simulation:

- Step response characterization
- Pulse superposition (in bit-by-bit mode)

Refer to *Description of Channel Simulation* (cktsimchan) for the principles behind ADS channel simulation.

The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

Channel Simulation Convolution Parameters

Setup Dialog Name	Parameter Name	Description
Tolerance	ToleranceMode	Tolerances control the speed and accuracy of step response calculation. This parameter can usually be left at the default value (<i>ToleranceMode=Auto</i>).
Relax	=Relax	Uses looser tolerances for convolution analysis of the step response. Looser tolerances lead to faster simulations at the expense of simulation accuracy.
Auto	=Auto	Default. Uses default convolution tolerances and provides a balance between speed and accuracy.
Strict	=Strict	Most accurate step response calculation at the expense of slower simulations.
Enforce passivity	EnforcePassivity	Enforces passivity during convolution analysis of the step response, and can usually be left at its default value (<i>EnforcePassivity=yes</i>).
Advanced		Click Advanced on the Convolution tab, and enable Advanced Convolution Options. These can usually be left at their default values. For option descriptions, see the following section, Defining Advanced Convolution Options .

Defining Advanced Convolution Options

Advanced Convolution Options contains parameters to help optimize a channel simulation. Typically, these parameters can be left at their default values.

The following table describes the parameter details. Names listed in the *Parameter Name* column are used in netlists and on schematics.

Channel Simulation Advanced Convolution Options

Setup Dialog Name	Parameter Name	Description
Maximum impulse response length (bit)	MaxImpulseLength	The Channel Simulator monitors the channel step (or impulse) response for the duration specified by this parameter. If the response doesn't settle or if it isn't detected during this period, increase <i>MaxImpulseLength</i> . (Default: <i>MaxImpulseLength=1000</i>)
Number of time points per UI	NumberTimePtPerUI	Controls the sampling of the pulse response calculated in the channel characterization stage. Fewer samples lead to faster simulations during pulse response superposition. (Default: <i>NumberTimePtPerUI=32</i>)
Enforce strict passivity	EnforceStrictPassivity	Select this setting (<i>EnforceStrictPassivity=yes</i>) if <i>EnforcePassivity</i> (on the Convolution tab) fails to yield a passive step response.
Save characterization result	SaveToDataset	Makes the step response available for viewing in the data display by writing a variable <i>outn</i> to the dataset.

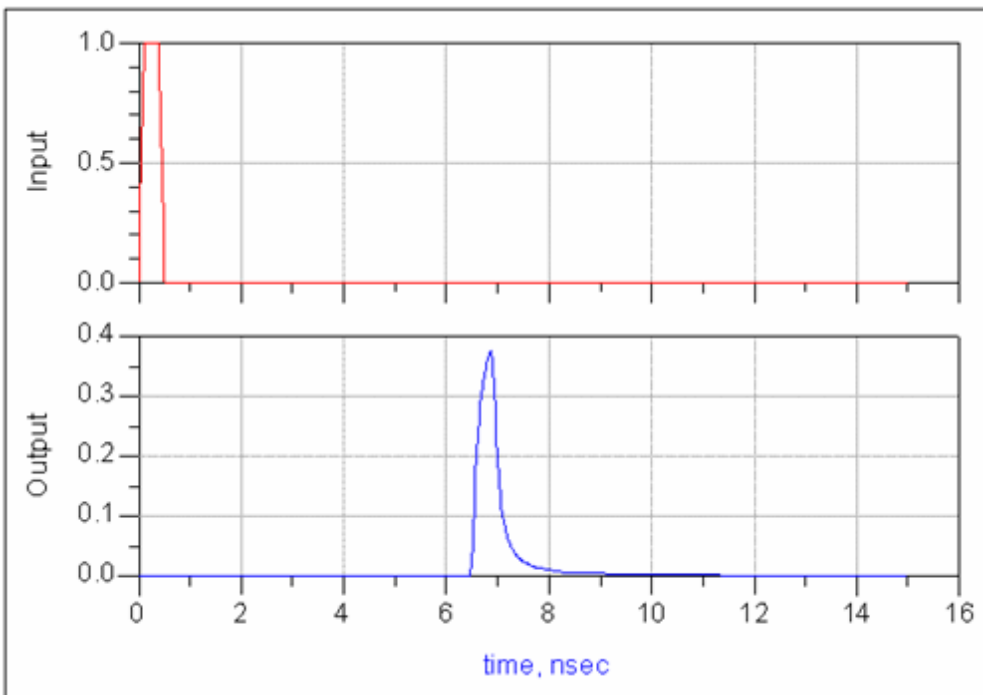
Description of Channel Simulation

The ADS Channel Simulator operates in one of two modes chosen by the user:

- [Bit-by-Bit Simulation](#)
- [Statistical Simulation](#)

Bit-by-Bit Simulation

The bit-by-bit mode computes the response to a specific bit sequence. In this mode, the Channel Simulator relies on linearity and time invariance of linear channels to achieve rapid simulations and high throughput. To understand the principles behind bit-by-bit channel simulation, consider the response $p_n(t)$ to an input NRZ pulse $r_n(t)$:



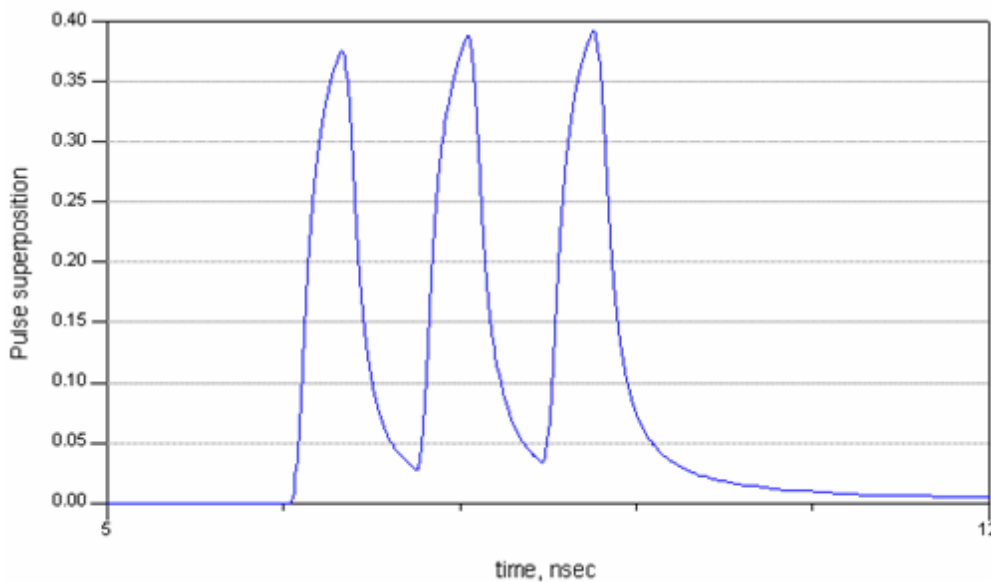
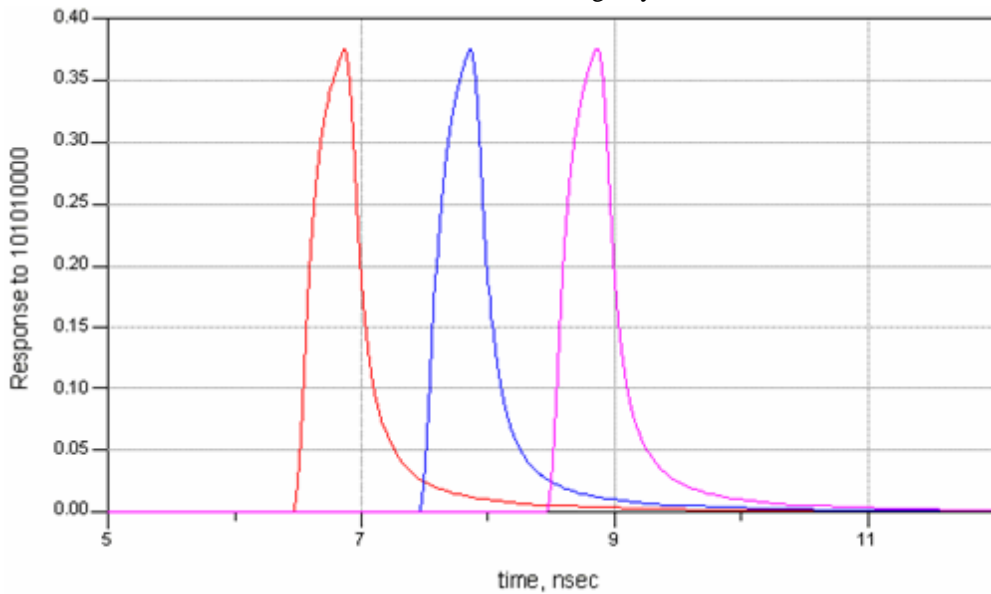
Assuming linearity and time invariance, the response to an arbitrary bit pattern

$$x(t) = \sum_n r_n(t - nT_b)$$

is given by the superposition of individual pulse responses

$$y(t) = \sum_n p_n(t - nT_b)$$

where T_b is the bit duration, or UI. The response to pattern 10101000 and pulse superposition are shown in the following figures:



Classical transient simulation results are overlaid on the plot labeled *Pulse superposition* showing a perfect match, as expected.

In bit-by-bit channel simulation the pulse response is obtained, in turn, by superposition of two step responses. The rising and falling edges of the step inputs are modulated by the various jitter components, as specified.

Channel simulation in bit-by-bit mode, therefore, consists of two steps:

1. Step characterization

During step characterization, the Channel Simulator invokes the ADS transient/convolution engine for accurate calculation of the system's step response. The simulator automatically detects step response duration, and accounts for the effects of transmit and receive equalizers, if any.

2. Pulse superposition

In this phase, the simulator adds up individual pulses and passes the output to eye probes for further processing. The simulator repeats this step for every eye probe

and every crosstalk driver in the circuit.

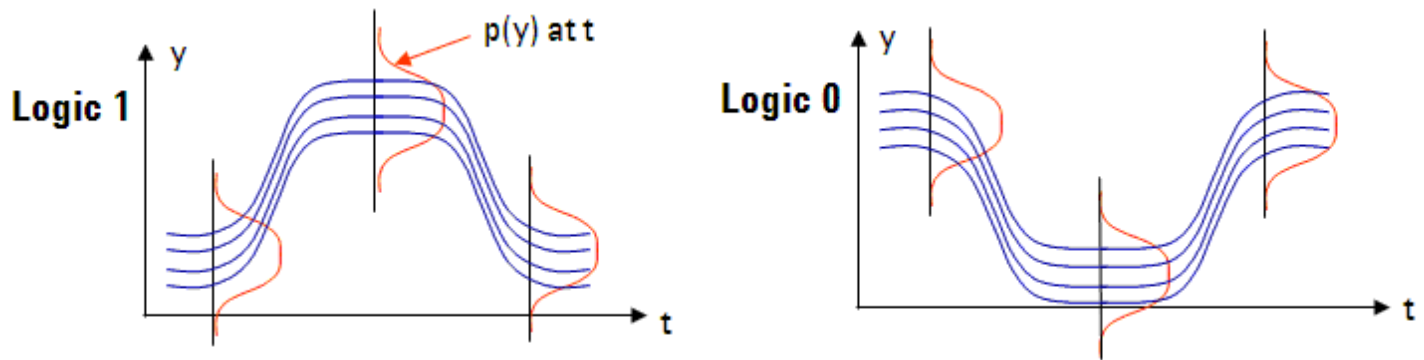
Statistical Simulation

In statistical channel simulation, system properties are derived from statistical calculations and not by brute-force superposition of pulse-reponses. The ADS implementation is based on the principles described in [Ref. 1, 2], with new and proprietary algorithms for accurate treatment of random and periodic jitter, DCD and other effects.

In this mode, the output

$$y(t) = \sum_n p_n(t-nT_b)$$

is viewed as a function of random variables b_k , where b_k is the input bit sequence. At a given time within the bit interval, the probability density function $p(y)$ is computed by statistical calculations, as shown below.



Similar to bit-by-bit mode, statistical simulation consists of two steps:

1. Step characterization
During step characterization, the Channel Simulator invokes the ADS transient/convolution engine for accurate calculation of the system's step response.
2. Statistical calculation
Statistical calculations are used to produce the eye density from the ISI distribution, taking into jitter specification, effects of crosstalk drivers, equalizers, encoding, etc. The Eye Probe component extracts all relevant quantities from the eye density distribution, including bathtubs, BER contours and other eye diagram metrics.

Bit-by-bit vs. Statistical Simulations

In the limit as the number of bits increases to infinity, the two channel simulation modes give identical results. Choose Statistical mode if you are interested in accurate simulations down to low BERs. Choose bit-by-bit simulation if you are interested in the response of your system to a specific bit sequence in the TX or crosstalk channel.

In bit-by-bit simulation, DFE/FFE RX equalizers can operate in adaptive mode. Adaptive DFE/FFE are not available in statistical simulation (you can still use DFE/FFE in optimized

or any other fixed-tap mode). To look at low-BER statistics with adaptive DFE/FFE, run bit-by-bit simulation and save DFE taps. You can then run statistical simulations by reading in the taps computed at the last time point of a bit-by-bit simulation.

Note that, with the exception of the Waveform measurement, which is available in bit-by-bit only, all *Eye_Probe* (ccsim) measurements are available in both channel simulation modes, including bathtub and contour plots. Statistical simulation offers more accurate results faster when you wish to calculate BERs down to low levels.

References

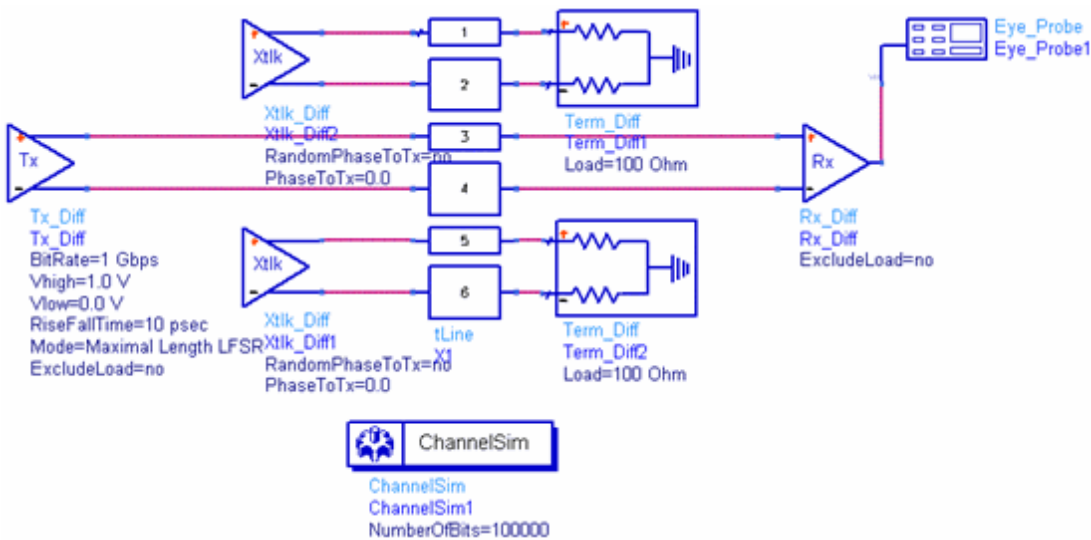
1. Anthony Sanders, Mike Resso, John D. Ambrosia. *Channel Compliance Testing Utilizing Novel Statistical Eye Methodology*, DesignCon 2004.
2. Fangyi Rao, Vuk Borich, Henock Abebe, Ming Yan. *Rigorous Modeling of Transmit Jitter for Accurate and Efficient Statistical Eye Simulation*, DesignCon 2010.

Using Channel Simulation

This section explains how to use the Channel Simulation controller with its related components. You'll learn how to create a design and specify parameter values to achieve the desired eye diagram results. Designs that demonstrate the following descriptions are located in *examples/SignalIntegrity/ChannelSimTutorial_wrk*.

Creating a Channel Simulation Design

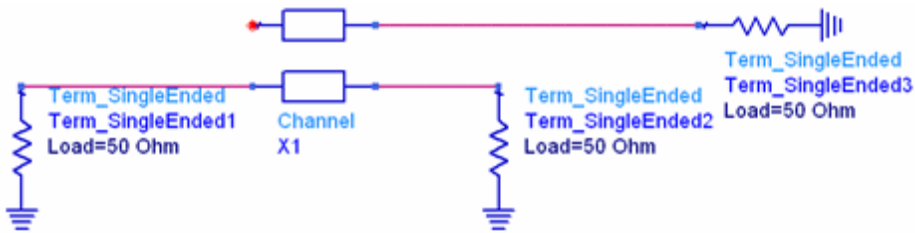
The following figure shows a design based on the general system architecture shown in *About Channel Simulation* (cktsimchan), as applied to ADS channel simulation:



The system consists of a coupled transmission line channel, a transmitter (*Tx_Diff*) (cktsimchan), two crosstalk transmitters (*Xtlk_Diff*) (cktsimchan), a receiver (*Rx_Diff*) (cktsimchan), and two terminations (*Term_Diff*) (cktsimchan). The Channel Simulator output is defined and measured by eye probe components (see *Eye_Probe* (ccsim)). The ChannelSim controller tells ADS to perform a channel simulation and defines the Channel Simulator parameters.

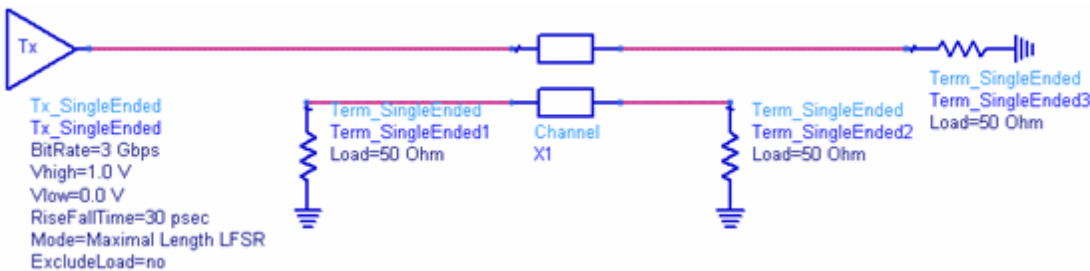
The following example demonstrates a simple channel simulation. (The purpose of the example is just to illustrate how to use Channel Simulator, and is not intended to be representative of a real-world chip-to-chip serial link. In contrast, the second, more complex, example *examples/SignalIntegrity/ChannelSimulatorPCIe2_wrk* shows a realistic PCI Express link.) The design, *DocExample*, is available in ADS and is located in the example workspace *examples/SignalIntegrity/ChannelSimTutorial_wrk*.

To get started, place a channel model on the schematic, as shown in the following figure:



The channel in this example represents a measured backplane model. The line is terminated by use of the Channel Simulator termination component. Termination models are provided for convenience only; equivalently, the line may be terminated with resistor models, Term elements, or other components.

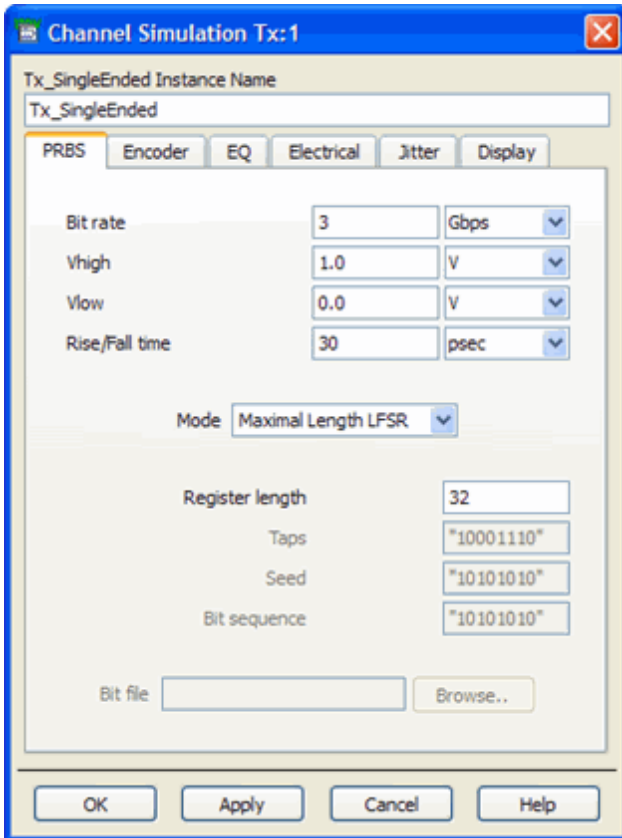
Next, connect a Tx model as shown in the following figure:



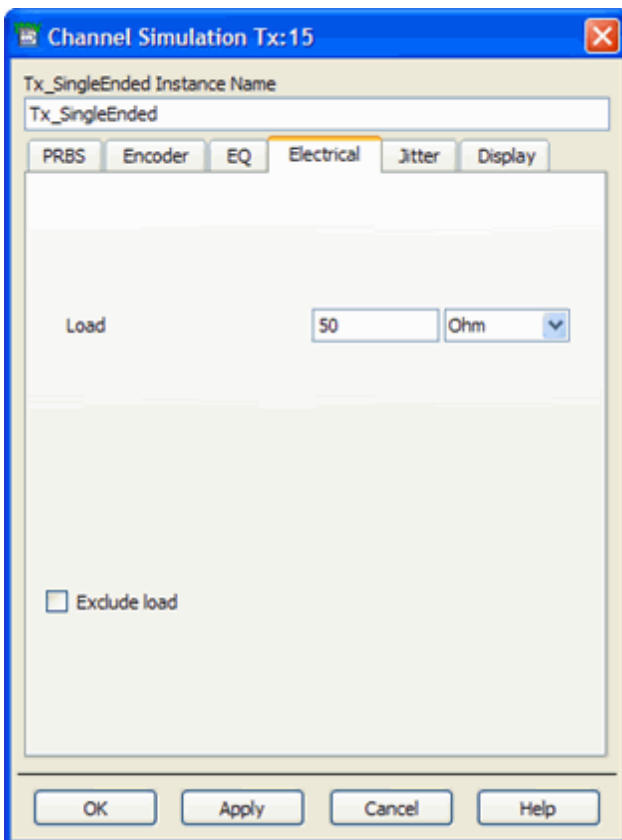
Double click the Tx model to open its setup dialog box. Select the PRBS tab and set the following values:

- Bit rate=3 Gbps
- Rise/Fall time=30 psec
- Register length=32

The PRBS tab should look like the following figure:

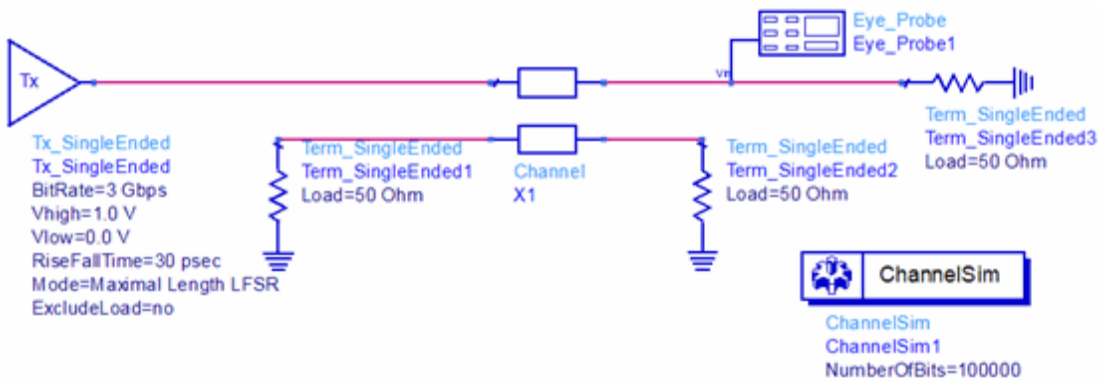


By default, the Tx source is ideal. To include a finite source impedance, select the Electrical tab, deselect *Exclude load* and set *Load* to 50 Ohms, as shown in the following figure:

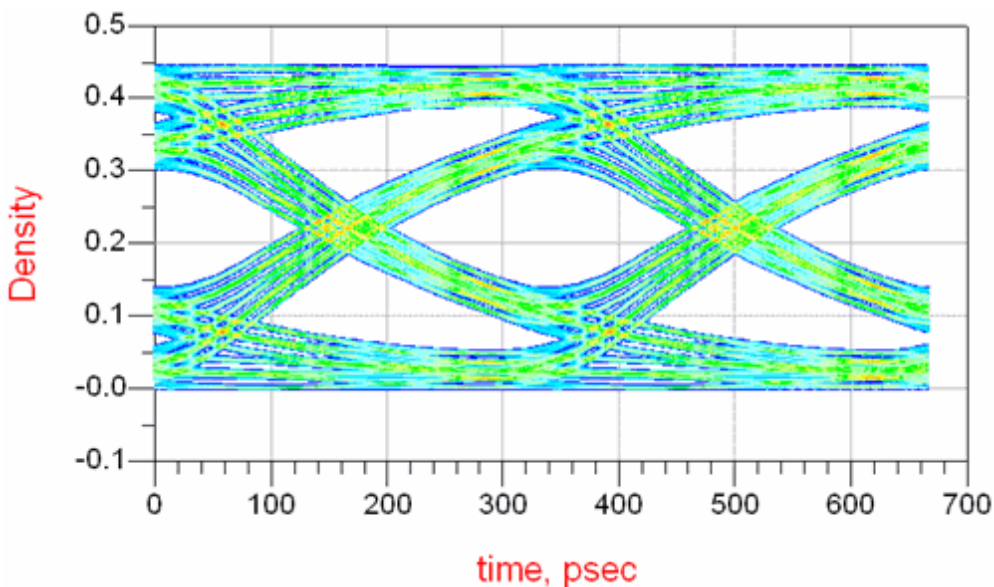


To complete the channel simulation set-up as shown in the following figure, attach an Eye_Probe component before the Term_SingleEnded3 component, and add a ChannelSim controller to the schematic. Set the following values:

- Eye_Probe component: set *Data rate* to 3 Gbps.
- ChannelSim controller: on the Analysis tab, choose *Bit-by-bit* simulation and set *Number of bits* to the number of bits you wish to simulate. In this case, the length of the simulated sequence is 100,000 bits.



Simulate the design, and plot the eye density in the data display. The simulation takes approximately ten seconds on a typical workstation. Your results should look similar to the following:



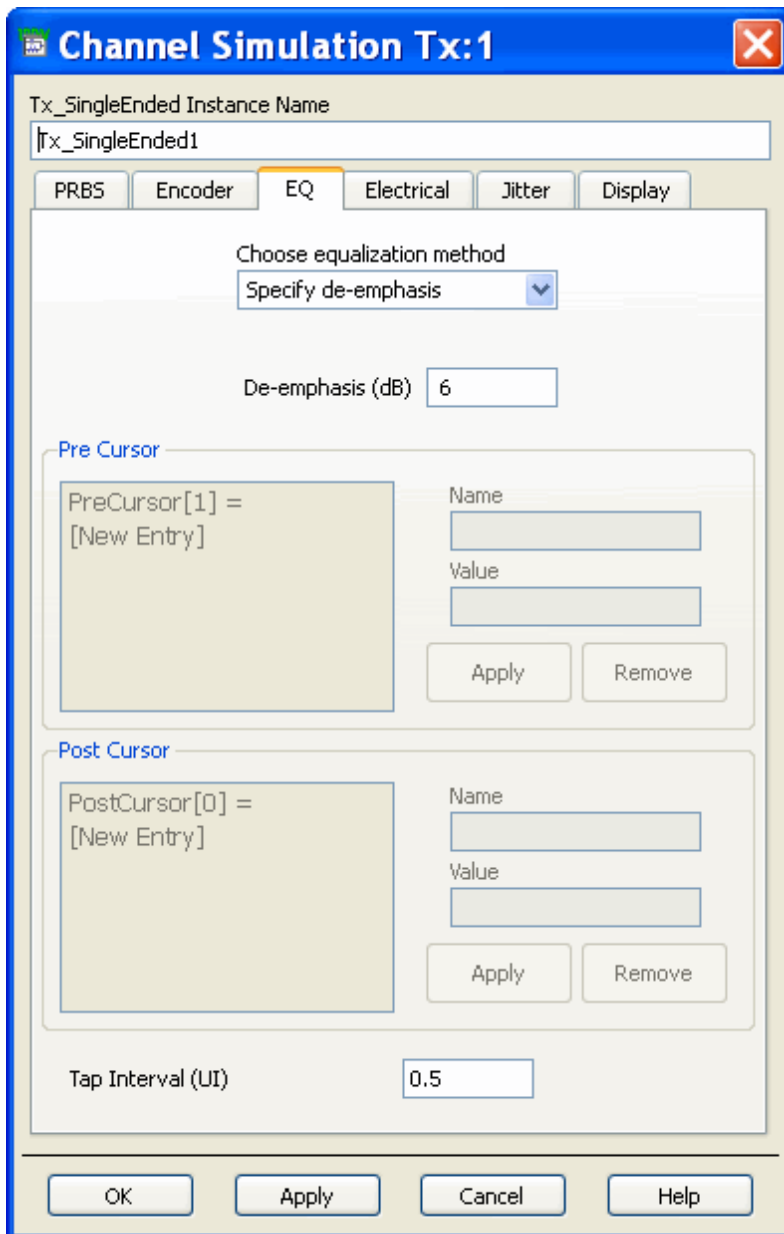
Adding Equalization Effects

The Channel Simulator makes it easy to design transmitter emphasis filters to improve eye performance. To model a 6 dB de-emphasis filter, on the Tx component's EQ tab, choose *Specify de-emphasis* under *Choose equalization method* and set *De-emphasis* to 6 dB. Alternatively, choose *Specify FIR taps* and set the following parameters:

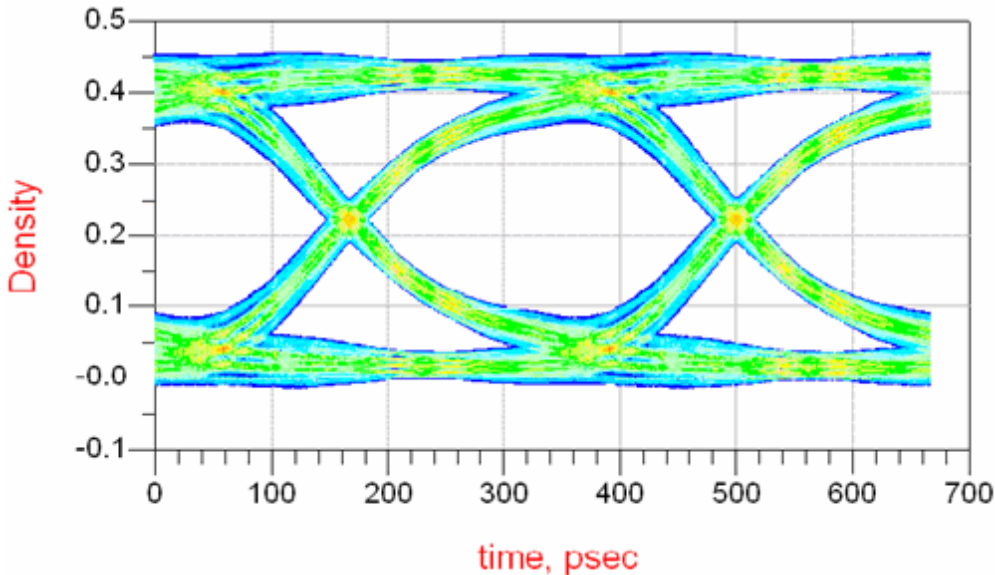
- PostCursor[0]=1.5
- PostCursor[1]=-0.5

- Tap Interval=0.5 UI

The EQ tab should look like the following figure:

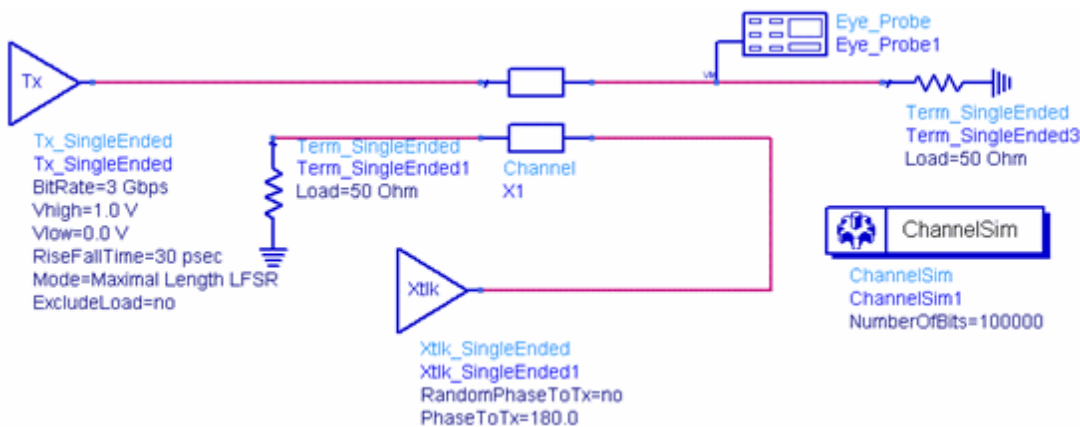


The effects of equalization on the eye opening are shown in the following figure:

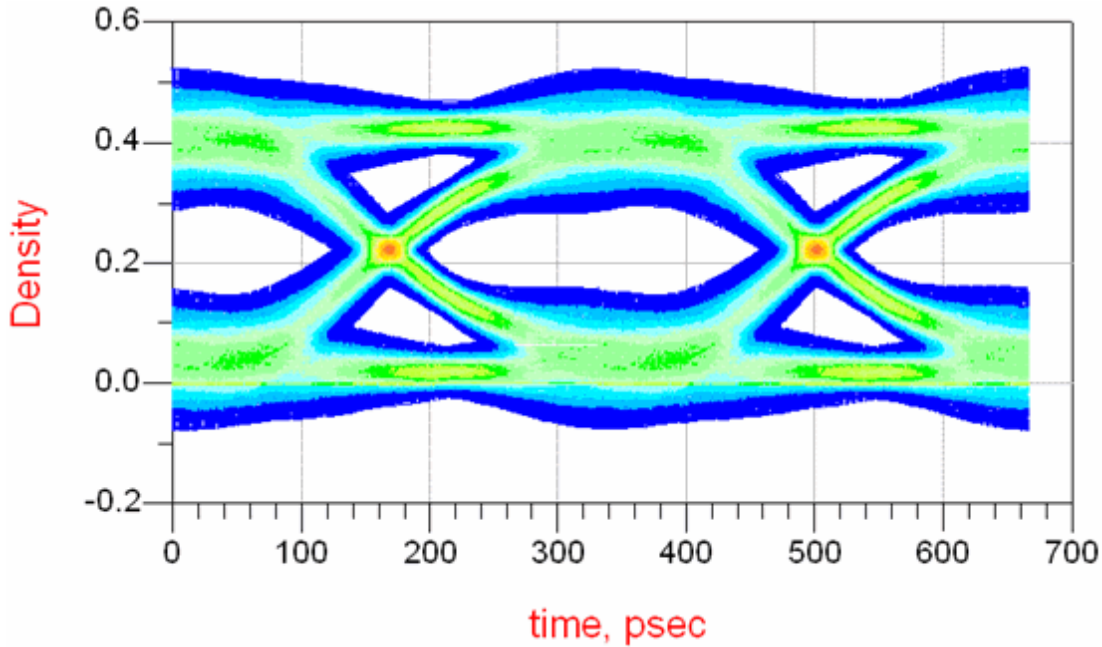


Adding Crosstalk Effects

Like equalization, channel simulation makes it straightforward to include the effects of crosstalk. To study crosstalk, insert one or more Xtlk components from the Simulation-ChannelSim palette. The following figure shows an example:



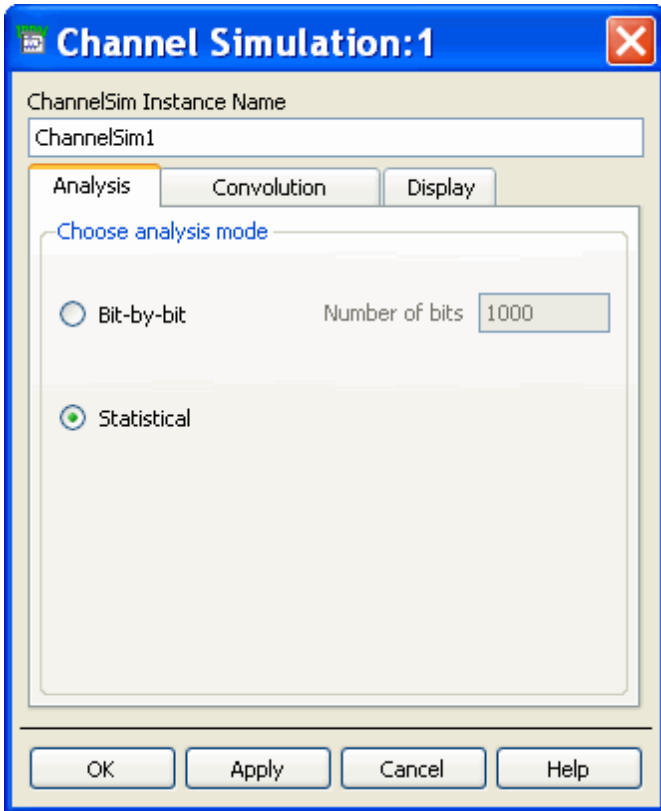
Xtlk components are *smart* in that, by default, they inherit the properties of the Tx component, such as bit rate, rise/fall times, jitter, etc. It is possible to override the inherited properties on the Xtlk component's dialog box. Often, however, all that is required is to specify the type of crosstalk to include in the simulation. The Channel Simulator models both synchronous and random (asynchronous) crosstalk. In this simulation, we look at the effects of 180 degree synchronous crosstalk on the output eye diagram as shown in the following figure:



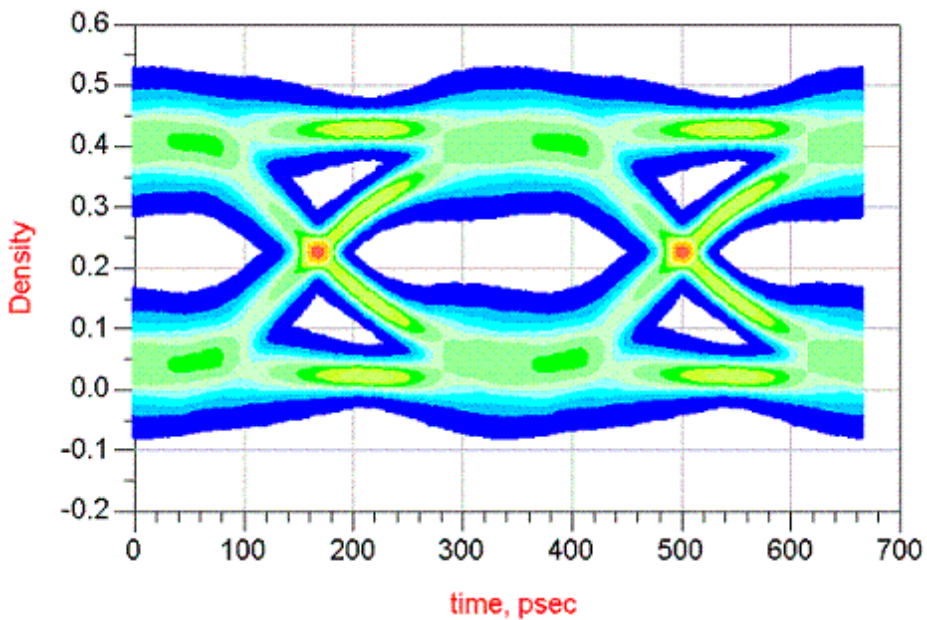
Running Channel Simulations in Statistical Mode

The Channel Simulator operates in bit-by-bit or in statistical mode. In bit-by-bit mode, a bit sequence is applied to the system as specified in the TX and crosstalk components. In statistical mode, the output is calculated by mathematical calculations using probability distribution functions for channel ISI, crosstalk, jitter, etc. For a very large number of bits, the two modes give identical answers. Statistical simulations are useful for low-BER simulations where the application of bit-by-bit simulation isn't practical. Bit-by-bit simulation is useful when analyzing the response to a specific bit sequence, or when running adaptive equalizer simulations.

You can choose from bit-by-bit or statistical simulation by setting the analysis mode parameter in the *Analysis* tab of the Channel Simulator controller, as shown below:



For an example of statistical simulations, open design *Statistical* in *examples/SignalIntegrity/ChannelSimTutorial_wrk*. This design is the same as *DocExample*, except it simulates in statistical mode. Note that results are very similar, as expected and as shown below:



Channel Simulation for AMI

This section describes about AMI models and its simulation setup.

Introduction to AMI

AMI (Algorithmic Modeling Interface) is the modeling interface for SERDES behavioral models which simulate SERDES functionalities such as equalization and CDR. AMI is introduced as an extension to the IBIS standard in version 5.0. The AMI portion is specified in IBIS file as part of the IBIS model. An AMI model acts as a DSP block which takes an input signal waveform and outputs a modified waveform. AMI models are developed by SERDES vendors to match and represent the actual chip behavior. Vendors deliver models in forms of DLL or/and shared object to protect their IP.

An AMI model consists of the following files:

- .ibs file
- .ami parameter file
- DLL or shared object file

The .ami file specifies model parameters.

AMI Parameter

There are following types of AMI parameters:

Parameter	Description
Reserved	Reserved parameters are common parameters shared by all models. Their names, types and meanings are defined by the AMI standard.
Model specific	Model specific parameters are private to the model. Model makers can define any number of model specific parameters under any name and any type.

AMI Parameter Mode

Each AMI parameter has one of the following four usage types:

Usage type	Description
Info	Info parameters are only used by simulators.
In	In parameters are only used by models.
InOut	These parameters are both used and updated by models. Their returned values will be used by simulators.
Out	These parameters are for models to returned values used by simulators.



Note

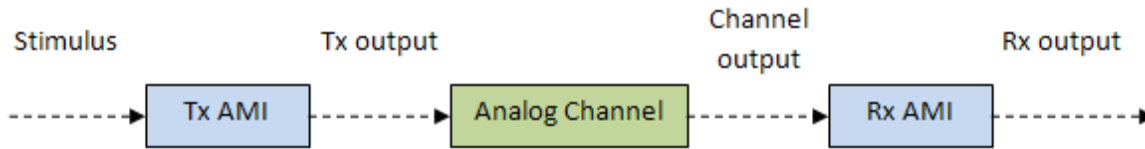
Only values of parameters of In and InOut types can be set by model users.

AMI models are applied in channel simulations to take into account effects of SERDES devices on channel performance. As shown below, the AMI methodology divides the

channel system into three parts:

- Tx AMI block
- Analog channel
- Rx AMI block

The analog channel includes the transmitter back-end, the physical channel and the receiver front-end.



During simulation the stimulus signal is first processed by the Tx AMI model. The Tx output signal is subsequently injected into the analog channel. The channel output is then processed by the Rx AMI model. The Rx output is used to calculate the eye and BER. Signals in AMI simulations are assumed to be differential.

AMI Simulation Setup

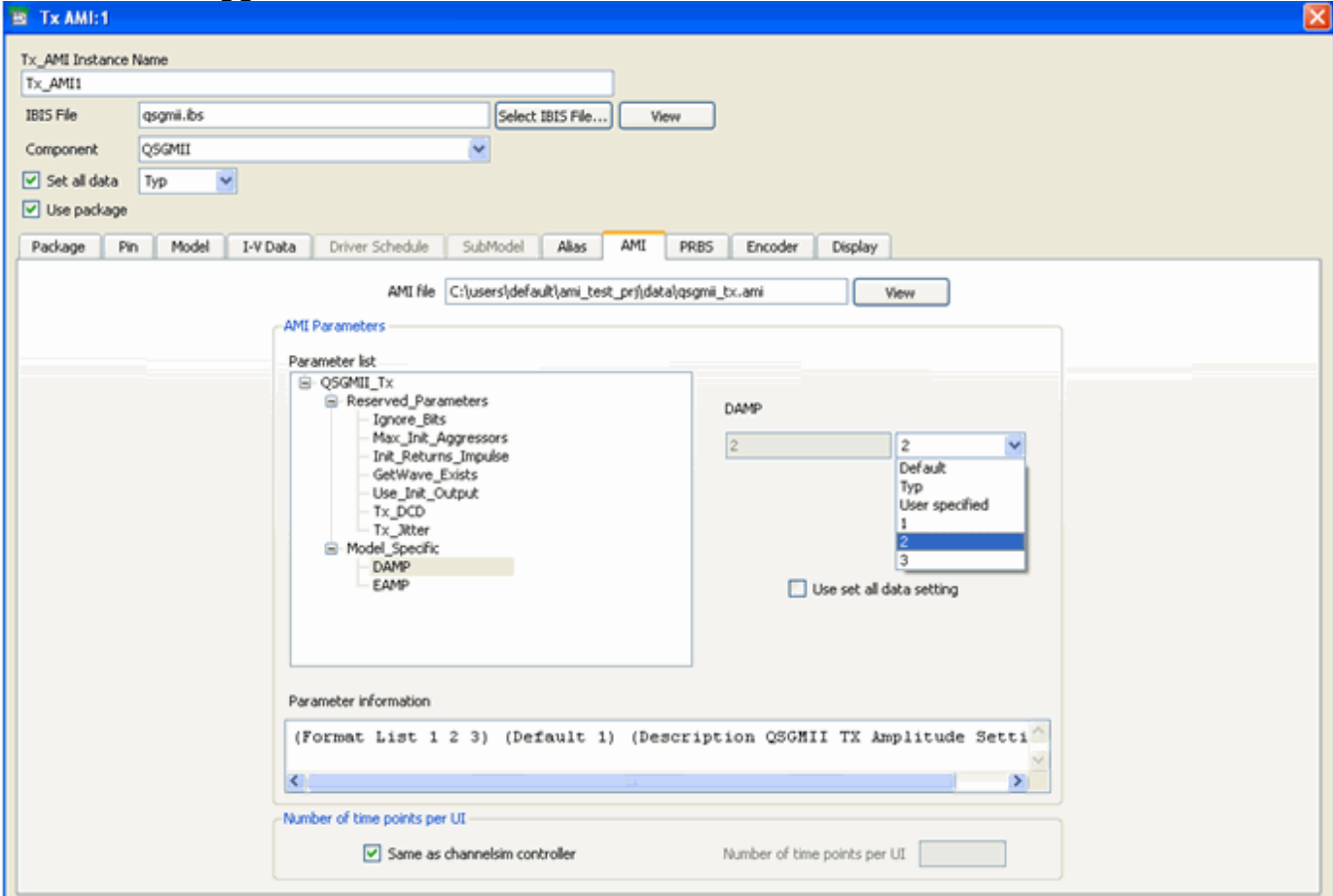
The setup of AMI simulation is identical to that of regular channel simulation. ADS provides AMI components of transmitter *Tx_AMI* (cktsimchan), receiver *Rx_AMI* (cktsimchan), crosstalk transmitter *XtlkTx_AMI* (cktsimchan) and crosstalk receiver *XtlkRx_AMI* (cktsimchan). Their icons are listed in the *Simulation-ChannelSim* palette and can be clicked on and placed in the design to create the corresponding component. The model is loaded to the component by selecting the IBIS file using the **Select IBIS File** button located at the top of the component setup dialog box shown below. After the model is loaded all AMI parameters will be listed in the *AMI* tab. The default parameter value is determined by the setting of the **Set all data** field at the top of the dialog box and the parameter format according to the following table:

	Value	Range	List	Corner	Increment	Steps
Typ	value	typ value	typ value	typ value	typ value	typ value
Min	value	min value	default/typ value*	default/typ value*	min value	min value
Max	value	max value	default/typ value*	default/typ value*	max value	max value
Fast	value	default/typ value*	default/typ value*	fast value	default/typ value*	default/typ value*
Slow	value	default/typ value*	default/typ value*	slow value	default/typ value*	default/typ value*

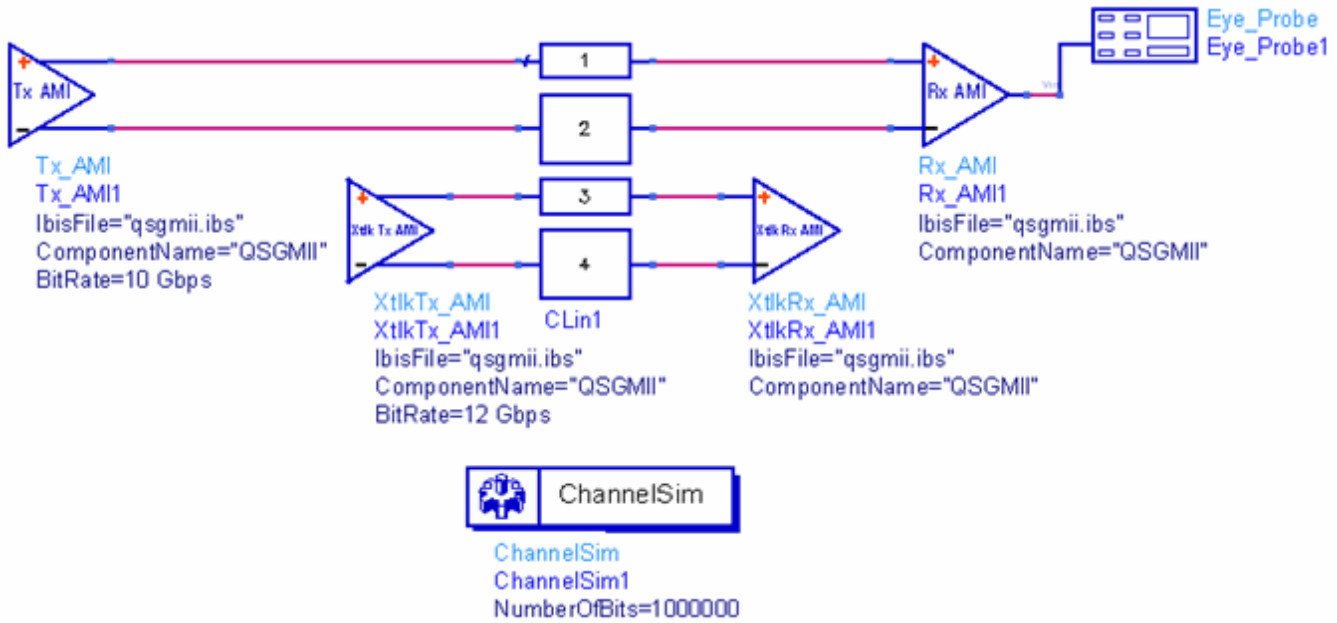
* Default value is used if default is defined. Otherwise typical value is used.

Parameter values can also be manually specified or selected using the drop-down list of allowed values next to the parameter list after un-checking the **Use set all data setting** option. Variable names defined in a *VAR* block can be entered as parameter values by using **User specified** option in the drop-down list for sweep, optimization or batch mode simulations. As mentioned above, only values of parameters of In and InOut usage types can be set by user. The number of time points per unit interval (UI) can be specified at the bottom of the dialog box. By default it is the same as the *NumberTimePtPerUI* parameter of the channel simulation controller. Different numbers of time points per UI can be used for different components. Bit rate, bit pattern and encoder of transmitter and

crosstalk aggressor can be set in *PRBS* and *Encoder* tabs. In AMI simulations, asynchronous aggressor with the same data rate as the victim is modeled by a small offset in the aggressor bit rate.



The design must contain a transmitter, a receiver and an eye probe *Eye Probe* (ccsim) connected to the receiver output node. Crosstalk can be included in the simulation. Crosstalk channels can be terminated by either *XtlkRx_AMI* component, or regular IBIS model, or any linear circuit component such as R, L and C. The following figure shows a design consist of a transmitter, a receiver, a crosstalk transmitter, a crosstalk receiver and two coupled differential channels.



The setup of channel simulation controller and eye probe is described in *Using Channel Simulation* (cktsimchan). ADS supports both bit-by-bit and statistical simulations for AMI models. In bit-by-bit mode the number of bits must be specified. Simulation results of eye measurements selected in the eye probe are saved to dataset and can be viewed with data display.

AMI Remote Simulation

AMI remote simulation is not supported on Windows. It works only when all of the following conditions are met.

- The server and the client operating systems are either Unix or Linux.
- The IBIS file is specified using the full network path.
- The IBIS file is accessible from the server.
- The AMI file is (consistently with IBIS specification) located in the same directory as the IBIS file.
- The AMI shared object library for the server platform must exist and is located in the same directory as the IBIS file.

Examples of Channel Simulation

ADS includes two channel simulation example workspaces, located in the *examples/SignalIntegrity* directory:

- *ChannelSimTutorial_wrk*
- *ChannelSimulatorPCIe2_wrk*

See their descriptions below.

Simulating Various Designs Using Channel Simulation

The workspace *ChannelSimTutorial_wrk* contains the following cells which demonstrate various applications:

- *DocExample* includes the example described in *Using Channel Simulation* (cktsimchan).
- *FileSweep* provides an example of sweeping S-parameter files through a channel simulation using the *DataFileList* (cktsimbatch) component.
- *Tuning* demonstrates the ADS tuning features in conjunction with the ChannelSim controller.
- *BatchEQ* demonstrates the use of Batch Simulation to sweep through several sets of equalizer coefficients and study eye diagram response. See *Batch Simulation* (cktsimbatch) for more information.
- *Statistical* demonstrates the use of Statistical simulation mode.
- *OptimalEqualizers* demonstrates the use of FFE/DFE with automatically optimized tap coefficients.
- *AdaptiveEqualizers* shows the operation of adaptive DFE equalization.
- *EyeMasks* demonstrates the use of eye masks.

Analyzing a PCIe Gen 2 System

The workspace *ChannelSimulatorPCIe2_wrk* demonstrates a channel simulation of a PCIe Gen 2 system:

- *PCIe_channel_1* is a PCIe channel analysis, with a single TX_Diff component and no crosstalk effects.
- *PCIe_channel_2* demonstrates the effects of crosstalk.
- *PCIe_channel_3* includes the effects of crosstalk and 8B/10B encoding.
- *PCIe_channel_4* shows the effects of crosstalk, 8B/10B encoding, and receiver jitter.

Limitations of Channel Simulation

The Channel Simulator relies on linearity and time invariance to achieve fast simulation speed and high bit throughput. For linear, time-invariant systems, channel simulation results are identical, within negligible numerical differences, to those obtained by a transient/convolution simulation. ADS does not prevent the inclusion of nonlinear elements in channel simulations so the user must ensure linearity. In some cases, mildly nonlinear systems give acceptable results. In general, only linear element systems apply to channel simulations. Just as ADS does not prevent the inclusion of nonlinear elements in simulations, it doesn't pose any restrictions on the method of describing linear circuit elements. Linear circuit elements may be included in any format, including ADS built-in transmission lines, Touchstone files, Verilog-A modules, HSPICE and Spectre netlists, and others.

The Channel Simulator relies on eye probe elements for output and eye diagram processing. Unlike the transient/convolution simulator, the channel simulator doesn't output node voltage waveforms up to a specified level of circuit hierarchy. Node waveforms at the eye probe nodes may be stored by selecting the *Eye_Probe* or *EyeDiff_Probe* components' *Waveform* measurement. For simulations of very long bit sequences, the output of node waveforms is not recommended because of large storage requirements. For details about the eye probe components, see *Eye_Probe* (ccsim) and *EyeDiff_Probe* (ccsim).

Troubleshooting Channel Simulation

This section discusses issues you may encounter while using the channel simulator.

How do I make simulations run even faster?

1. Simulation speed has a linear dependence on the number of eye probes and transmitters (TX and Xtlk) in the circuit. Reduce the number of eye probes for faster simulations.
2. Ensure that the TX rise/fall time is set to a realistic value. TX rise time determines the transient analysis time step in pulse characterization and the maximum frequency up to which passive devices are evaluated in convolution.
3. Reduce the Advanced Convolution Option *Number of time points per UI* (on the ChannelSim controller's setup dialog Convolution tab, click *Advanced*).
4. Loosen the convolution tolerance in the ChannelSim controller.

I don't see the waveform at named schematic nodes in the dataset.

The Channel Simulator does not store circuit waveforms at named schematic nodes. To view a waveform, attach an eye probe to the node of interest and select the *Waveform* measurement.

Why is only one TX component allowed on the schematic?

Any number of crosstalk transmitters are allowed. TX components are unique because they define global simulation parameters such as data rate, and provide default values for most crosstalk driver parameters.

The response of my circuit is non-passive, even though passivity is enforced in the ChannelSim controller.

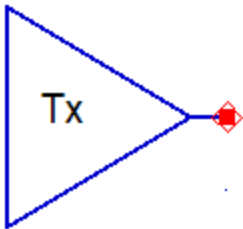
Occurrences of passivity violations in ADS are extremely rare. In some cases, many-port S-parameter files may exhibit non-passive behavior even after passivity enforcement. The usual remedy is to enable the Advanced Convolution Option *Enforce strict passivity* (on the ChannelSim controller's setup dialog Convolution tab, click *Advanced*). Strict passivity enforcement will apply stricter passivity enforcement rules in the region where S-parameters are extrapolated beyond the maximum frequency in the data file.

The eye density plot indicates zero output at the eye probe

If channel delay is long, increase the Advanced Convolution Option *Maximum impulse response length* (on the ChannelSim controller's setup dialog Convolution tab, click *Advanced*). By default, the Channel Simulator measures the step response of the system for the duration of 1000 bits. If the response is not detected within this period, the Channel Simulator may produce zero output.

Tx_SingleEnded (Channel Simulation Single-Ended Transmitter)

Symbol



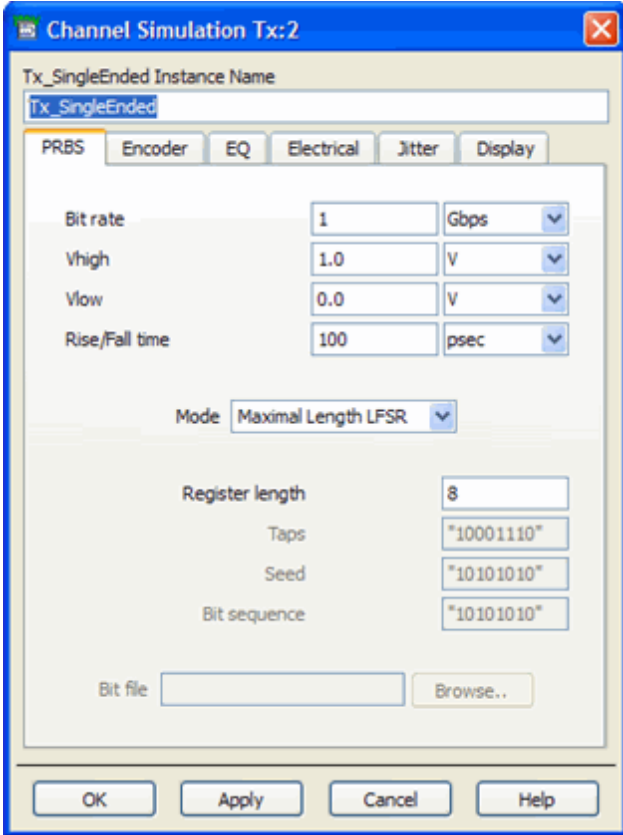
The Tx model encompasses the functionality of a PRBS source, encoder, and equalizer. The source may be an ideal voltage source (i.e. zero source impedance) or have a specified source impedance, as defined by its *Electrical* parameters. Optionally, the source may inject jitter in several different forms.

The Tx parameters are organized into the following functional groups enabling you to specify the source characteristics:

- **PRBS** specifies the Tx excitation source.
- **Encoder** selects 8B/10B or no encoding.
- **Equalization** designs transmitter emphasis filters.
- **Electrical** sets the source impedance.
- **Jitter** selects the type of jitter applied to the Tx source.
- **Display** controls the visibility of component parameters on the schematic (see *Displaying Simulation Parameters on the Schematic (cktsim)*).

PRBS Parameters

Use the PRBS tab to specify the Tx excitation source in several different formats:



Tx_SingleEnded PRBS Parameters

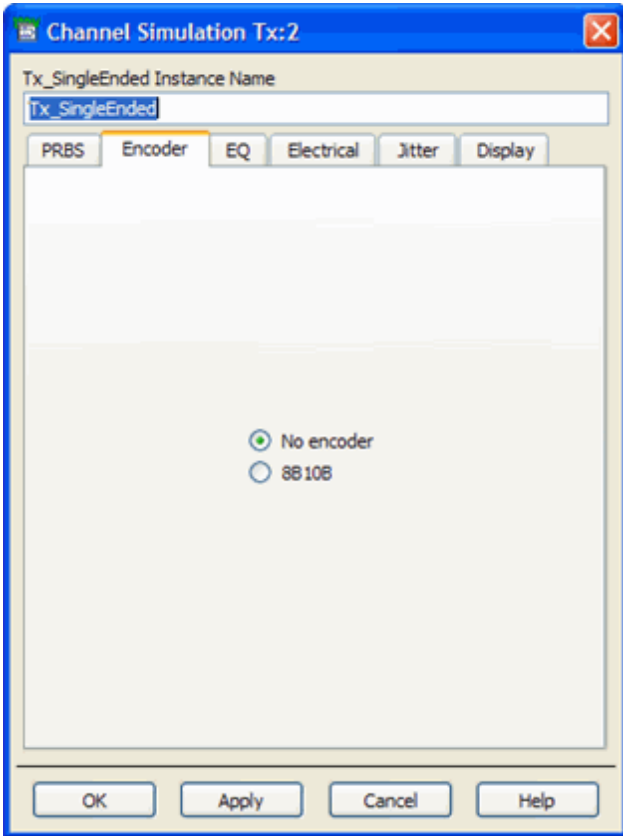
Setup Dialog Name	Parameter Name	Description	Units	Default
Bit rate	BitRate	Data rate of the PRBS source. This parameter also establishes the simulation data rate.	bps	1 Gbps
Vhigh	Vhigh	Logic-1 voltage level (see note)	V	1 V
Vlow	Vlow	Logic-0 voltage level (see note)	V	0 V
Rise/Fall time	RiseFallTime	Source rise and fall times. The rise and fall times must be in the range between 0.01 UI and 0.5 UI.	sec	10 ps
Mode	Mode	PRBS pattern mode. Select one of the following: <ul style="list-style-type: none"> Maximal Length LFSR User Defined LFSR User Defined Sequence Bit File. 	None	Maximal Length LFSR
Register length	RegisterLength	Length of shift register, in bits, in maximal length mode.	None	8
Taps	Taps	LFSR taps in user-defined LFSR mode.	None	"10001110"
Seed	Seed	LFSR seed in user-defined LFSR mode.	None	"10101010"
Bit sequence	BitSequence	Specifies a user-defined bit sequence. The pattern repeats if the simulation extends beyond the length of the user-defined sequence.	None	"10101010"
Bit file	BitFile	The pattern is specified in a text file. The sequence is specified as an array of 1s and 0s in row or column format.	None	None

Notes

1. The specified voltage is the internal voltage. If a source impedance is specified, the voltage at the output node will be subject to the resulting source/load voltage divider.
2. For Maximal Length LFSR mode, seed is fixed value depending on the Register length.

Encoder Parameters

Use the Encoder tab to select 8B/10B encoding.

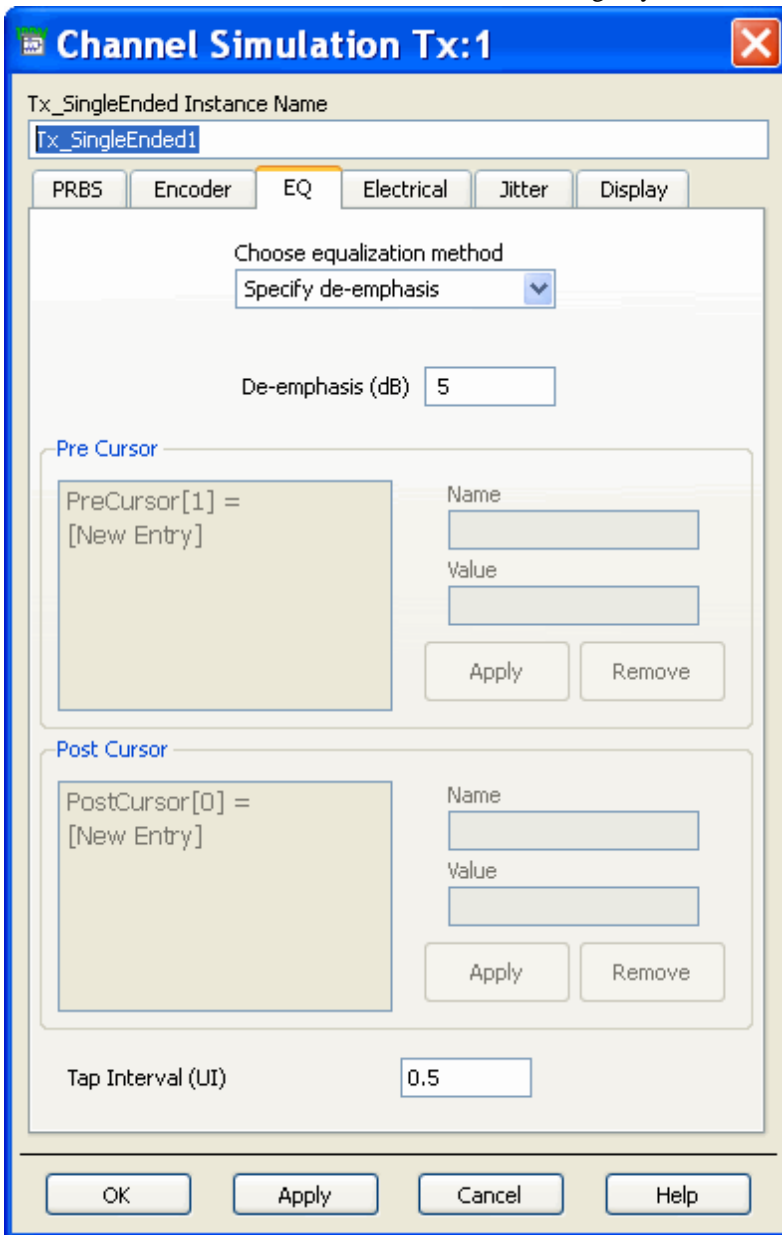


Tx_SingleEnded Encoder Parameters

Setup Dialog Name	Parameter Name	Description
No encoder	Encoder=No encoder	By default, no encoding is applied to the PRBS sequence.
8B10B	Encoder=8B10B	Select this option to apply 8B/10B encoding to the PRBS sequence.

Equalization Parameters

Use the EQ tab to design transmitter emphasis filters.



Tx_SingleEnded EQ Parameters

Setup Dialog Name	Parameter Name	Description
Choose equalization method	EQMode	Equalization mode: <i>Specify FIR taps, Specify de-emphasis or None</i>
De-emphasis (dB)	Deemphasis	dB de-emphasis when Equalization mode = <i>Specify de-emphasis</i>
Pre Cursor	PreCursor[n]	Sets the n^{th} pre-cursor tap, starting from $n=1$.
Post Cursor	PostCursor[n]	Sets the n^{th} post-cursor tap, starting from $n=0$.
Tap Interval (UI)	TapInterval	Sets the tap delay, measured in UI.

Tx equalization can be specified in terms of FIR taps, or in terms of db de-emphasis where 2-tap equalization is automatically applied by the simulator based on the de-emphasis setting. You can easily turn equalization on and off by selecting Equalization mode = *None*.

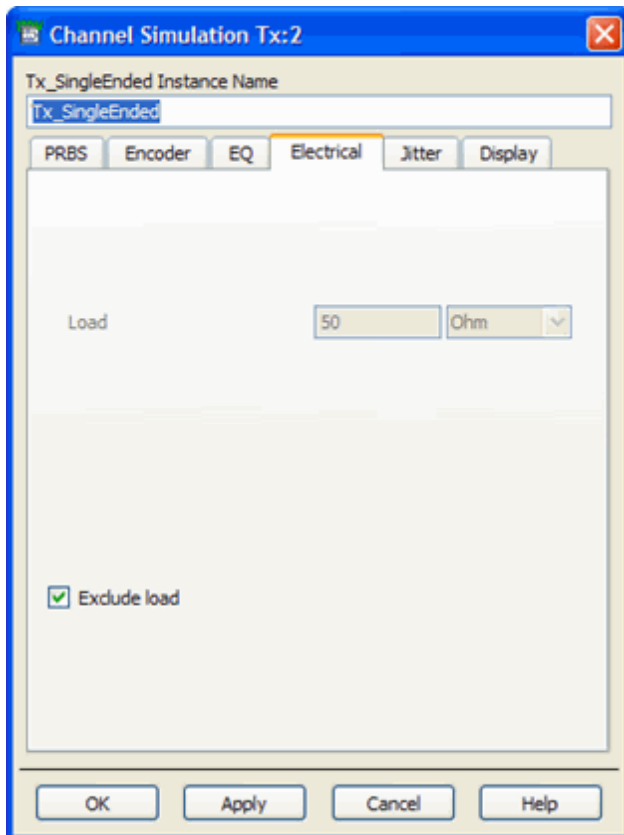
When Equalization mode = *Specify FIR taps*, the Tx model applies an FIR equalizer to the Tx PRBS source. The n^{th} sample $y[n]$ of the equalized Tx source $x[n]$ is given by:

$$y[n] = \text{PostCursor}[0]*x[n] + \text{PostCursor}[1]*x[n-\text{Delta}] + \text{PostCursor}[2]*x[n-2*\text{Delta}] + \dots + \text{PreCursor}[1]*x[n+\text{Delta}] + \text{PreCursor}[2]*x[n+2*\text{Delta}] \dots$$

The number of pre- and post-cursor taps is unlimited. The tap spacing is specified as a fraction of UI.

Electrical Parameters

Use the Electrical tab to set the source impedance for the circuit.



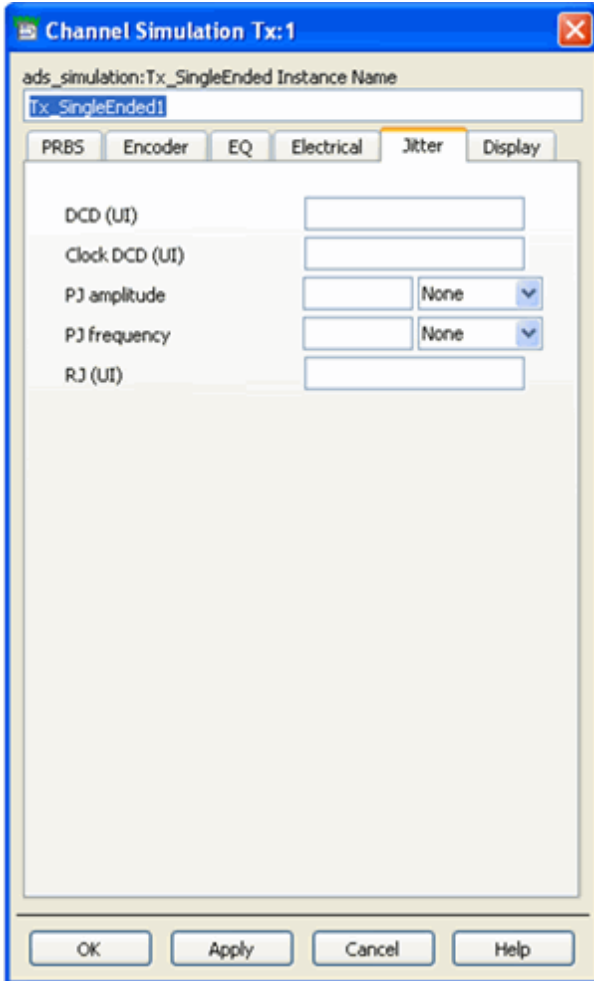
Tx_SingleEnded Electrical Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
Load	Load	Specify a finite source impedance.	Ohm	50
Exclude load	ExcludeLoad	Select <i>Exclude load</i> (<i>ExcludeLoad=yes</i>) to make the Tx an ideal voltage source (i.e. zero source impedance).	None	yes

The electrical properties of the source are controlled by the Electrical properties tab. By default the Tx PRBS source – optionally encoded, filtered, and jittered – is applied to the circuit as an ideal independent voltage source. The Tx model also enables you to specify a finite source impedance, so you don't need to include it explicitly in your schematic.

Jitter Parameters

Use the Jitter tab to apply random, periodic, and DCD jitter to the Tx source.



Tx_SingleEnded Jitter Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
DCD (UI)	DCD	Duty-cycle distortion in data signal as a fraction of UI. With DCD all 1 bits are longer (or shorter) than all 0 bits.	None	0.0
Clock DCD (UI)	ClockDCD	Duty-cycle distortion in clock signal as a fraction of UI. It's also called F/2 jitter. With Clock DCD all even bits are longer (or shorter) than all odd bits.	None	0.0
PJ amplitude	PJAmplitude	Periodic jitter amplitude	sec	0.0
PJ frequency	PJFrequency	Periodic jitter frequency	Hz	0.0
RJ (UI)	RJ	RMS random jitter as a fraction of UI	None	0.0

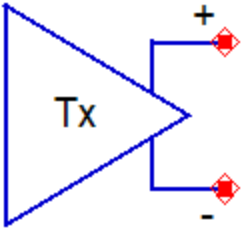
Notes

1. *One (and only one)* instance of a Tx model (in single-ended or differential configuration) must be included in the schematic to perform a channel simulation. Additional transmitters are viewed by the simulator as crosstalk drivers and should be modeled by one of the Xtlk components (see *Xtlk_SingleEnded* (cktsimchan) and *Xtlk_Diff* (cktsimchan)).

2. The Tx model defines the system's data rate for the purposes of channel simulation.
3. Crosstalk transmitters, by default, inherit Tx parameters such as voltage levels, rise/fall times, and jitter.

Tx_Diff (Channel Simulation Differential Transmitter)

Symbol

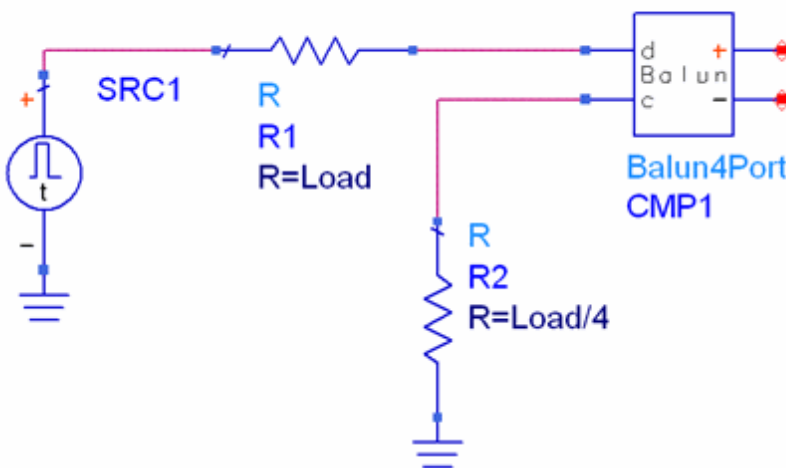


Notes

1. Tx_Diff is a differential transmitter model.
2. All Tx_Diff model parameters are the same as the single-ended version of the Tx model (see *Tx_SingleEnded*) (cktsimchan)).
3. The Tx_Diff model is identical to the single-ended version except in its electrical properties. The electrically equivalent model of the differential source is shown in the following figure. Internally to the model, the ADS 4-port balun model is used for single-ended-to-differential conversion (see *Balun4Port* (ccsys)).

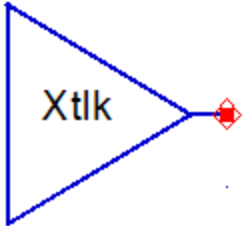
Single-ended input

Differential Tx output



Xtlk_SingleEnded (Channel Simulation Single-Ended Crosstalk Transmitter)

Symbol



Note

This component is obsolete after ADS 2009 Update 1. It has been replaced by Xtlk2_SingleEnded. Xtlk2_SingleEnded offers more features and flexibility in defining the crosstalk excitation, including sequence generation, equalization and encoding.

Existing designs created before ADS 2009 Update 1 will import correctly and will use Xtlk_SingleEnded as before. New designs will use the new component, Xtlk2_SingleEnded. It is possible, though strongly discouraged, to invoke the old model by typing Xtlk_SingleEnded in the ADS Component History box.

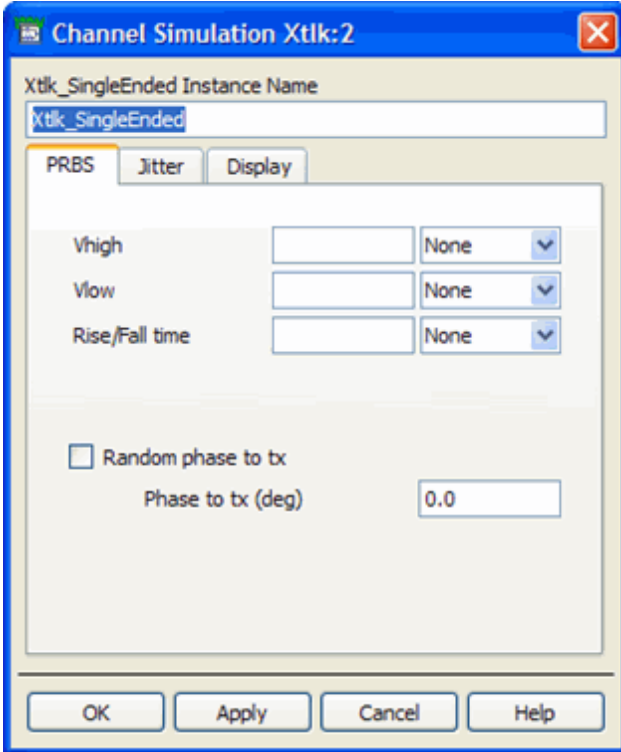
Xtlk components are very similar to the Tx components (see Tx components *Tx_SingleEnded* (cktsimchan) and *Tx_Diff* (cktsimchan)). Unlike Tx, which is required and must be unique on the schematic, Xtlk drivers are optional. In addition, multiple Xtlk drivers are supported and there is no limit to the number of Xtlk components allowed in a channel simulation.

The Xtlk parameters are organized into the following functional groups enabling you to specify the crosstalk characteristics:

- **PRBS** specifies the Xtlk excitation source.
- **Jitter** selects the type of jitter applied to the Xtlk source.
- **Display** controls the visibility of component parameters on the schematic (see *Displaying Simulation Parameters on the Schematic* (cktsim)).

PRBS Parameters

Use the PRBS tab to specify the Xtlk excitation source in several different formats.

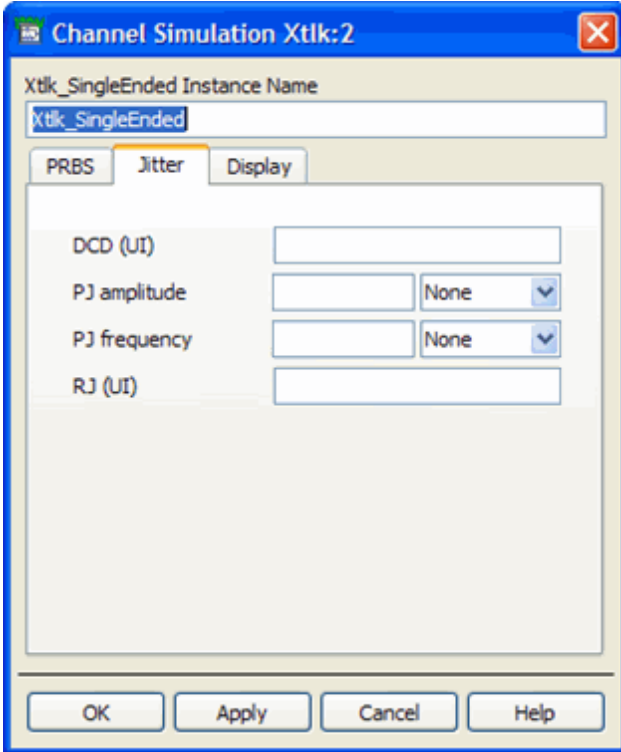


Xtlk_SingleEnded PRBS Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
Vhigh	Vhigh	Logic-1 voltage level	V	Inherited from Tx
Vlow	Vlow	Logic-0 voltage level	V	Inherited from Tx
Rise/Fall time	RiseFallTime	Source rise and fall times. The rise and fall times must be in the range between 0.01 UI and 0.5 UI.	sec	Inherited from Tx
Random phase to tx	RandomPhaseToTx	Select to enable random crosstalk modeling.	None	No
Phase to tx (deg)	PhaseToTx	Sets the phase relationship between the crosstalk and Tx drivers for synchronous crosstalk modeling. 1 UI = 360 degrees.	None	0.0

Jitter Parameters

Use the Jitter tab to apply random, periodic, and DCD jitter to the Xtlk source.



Xtlk_SingleEnded Jitter Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
DCD (UI)	DCD	Duty-cycle distortion as a fraction of UI	None	Inherited from Tx
PJ amplitude	PJAmplitude	Periodic jitter amplitude	sec	Inherited from Tx
PJ frequency	PJFrequency	Periodic jitter frequency	Hz	Inherited from Tx
RJ (UI)	RJ	RMS random jitter as a fraction of UI	None	Inherited from Tx

Notes

1. Xtlk components inherit most of their properties from the Tx used in the circuit, including voltage levels, source impedance, PRBS parameters, encoding, equalization, and jitter. Many parameters may be overridden, as shown in the parameter tables above.
2. The Channel Simulator models both synchronous and random (asynchronous) crosstalk. For synchronous crosstalk, the crosstalk excitation has a fixed phase relationship to the driver. Synchronous crosstalk contribution is obtained by a pulse superposition method similar to the calculation of Tx response, as outlined in *Description of Channel Simulation* (cktsimchan). Random crosstalk is calculated by statistical methods, by convolving the probability density functions of the forward channel and crosstalk responses.
3. Xtlk component parameters are analogous to Tx parameters. Often, the type of crosstalk (synchronous or random) is all that is necessary to specify a Xtlk component.

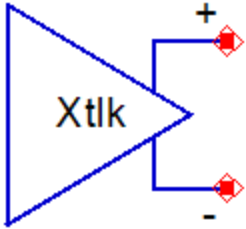


Note

When a Xtlk parameter is left unspecified, it assumes the value of that parameter from the Tx component.

Xtlk_Diff (Channel Simulation Differential Crosstalk Transmitter)

Symbol



Note

This component is obsolete after ADS 2009 Update 1. It has been replaced by Xtlk2_Diff. Xtlk2_Diff offers more features and flexibility in defining the crosstalk excitation, including sequence generation, equalization and encoding.

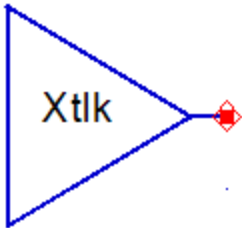
Existing designs created before ADS 2009 Update 1 will import correctly and will use Xtlk_Diff as before. New designs will use the new component, Xtlk2_Diff. It is possible, though strongly discouraged, to invoke the old model by typing Xtlk_Diff in the ADS Component History box.

Notes

1. The Xtlk_Diff is a differential crosstalk transmitter.
2. The differential crosstalk transmitter differs from the single-ended crosstalk model only in electrical properties (see *Xtlk_SingleEnded* (cktsimchan)). A 4-port balun is used for single-ended-to-differential conversion internally to the model. See *Tx_Diff* (cktsimchan) for a figure of the equivalent circuit.
3. All other parameters are the same as the single-ended version of the Xtlk.

Xtlk2_SingleEnded (Channel Simulation Single-Ended Crosstalk2 Transmitter)

Symbol



Xtlk components are very similar to the Tx components (see Tx components *Tx_SingleEnded* (cktsimchan) and *Tx_Diff* (cktsimchan)). Unlike Tx, which is required and must be unique on the schematic, Xtlk drivers are optional. In addition, multiple Xtlk drivers are supported and there is no limit to the number of Xtlk components allowed in a channel simulation.

By default, Xtlk components inherit most of their properties from the Tx used in the circuit, including voltage levels, source impedance, PRBS parameters, encoding, equalization, and jitter. Those properties may be overridden, as shown in the parameter tables below.

The Channel Simulator models both synchronous and random (asynchronous) crosstalk. For synchronous crosstalk, the crosstalk excitation has a fixed phase relationship to the driver, and synchronous crosstalk contribution is obtained by a pulse superposition method similar to the calculation of Tx response, as outlined in *Description of Channel Simulation* (cktsimchan). Random crosstalk is calculated by statistical methods, by convolving the probability density functions of the forward channel and crosstalk responses.

Most Xtlk component parameters are analogous to Tx parameters. Often, the type of crosstalk (synchronous or random) is necessary to specify a Xtlk component, but the component gives you the flexibility to override virtually all Tx parameters to customize its behavior.

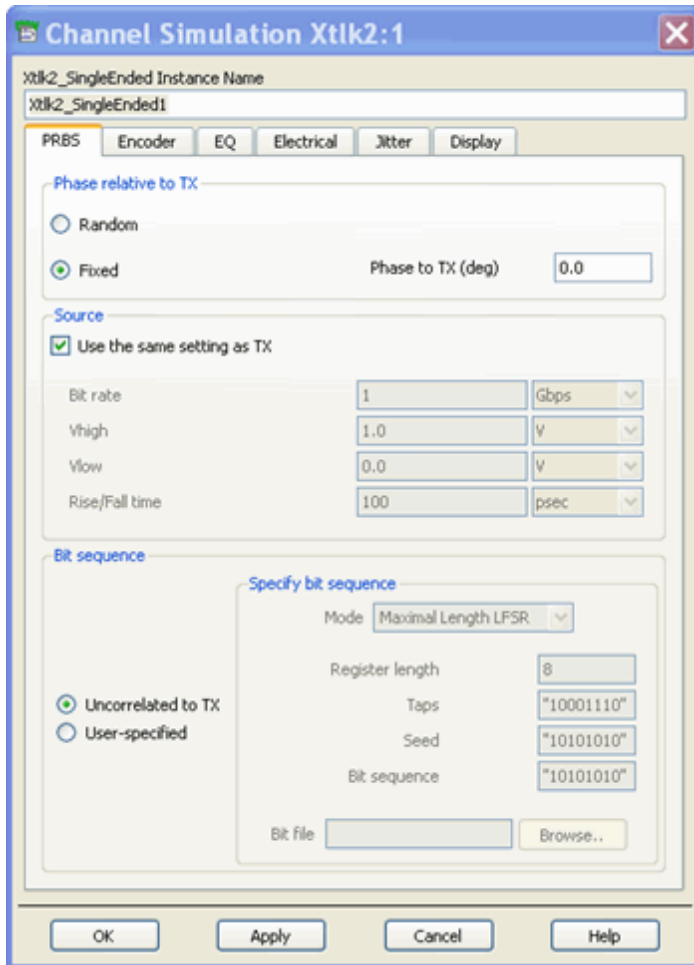
The Xtlk parameters are organized into the following functional groups enabling you to specify the crosstalk characteristics:

- **PRBS** specifies the Xtlk excitation source.
- **Encoder** lets you select the encoding scheme.
- **EQ** specifies FIR equalizer coefficients or transmit de-emphasis.
- **Electrical** defines the electrical properties of the driver, such as equivalent impedance.
- **Jitter** allows you to inject random, periodic or DCD jitter.
- **Display** controls the visibility of component parameters on the schematic (see *Displaying Simulation Parameters on the Schematic* (cktsim)).

Except for PRBS properties, the remaining parameters tabs (Encoder, EQ, Electrical and Jitter) are virtually identical to Tx (see Tx components *Tx_SingleEnded* (cktsimchan) and *Tx_Diff* (cktsimchan))

PRBS Parameters

Use the *PRBS* tab to specify the Xtlk excitation source in several different formats.



Xtlk_SingleEnded PRBS Parameters

PRBS parameters are organized in three different categories.

1. Phase parameters determine the type of crosstalk to be modeled, i.e., random or synchronous. In synchronous crosstalk mode, you can adjust the phase of the crosstalk source relative to TX.

Parameter Name	Description	Units	Default
Random	The crosstalk source has a random (asynchronous) phase relationship to TX.	None	No
Fixed	The crosstalk source has a fixed (synchronous) phase relationship to TX	None	Yes
Phase to TX	Fixed phase offset (in degrees) between the crosstalk and TX sources (1UI = 360 degrees)	None	0.0

2. Source parameters determine the data rate of the crosstalk source, voltage levels and rise/fall times. By default, the parameters are automatically inherited from TX and often do not require adjustment. The Bit rate parameter may be set to a value different from TX; this setting is only active when the source is in asynchronous mode.

Parameter Name	Description	Units	Default
Use the same settings as TX	When enabled, all parameters in this group are inherited from the TX	None	Yes
Bit rate	Bit rate of the crosstalk source	Bps	Inherited from Tx
Vhigh	Logic-1 voltage level (see note)	V	Inherited from Tx
Vlow	Logic-0 voltage level (see note)	V	Inherited from Tx
Rise/Fall time	Source rise and fall times. The rise and fall times must be in the range between 0.01 UI and 0.5 UI.	sec	Inherited from Tx



Note

The specified voltage is the internal voltage. If a source impedance is specified, the voltage at the output node will be subject to the resulting source/load voltage divider.

3. Bit sequence parameters allow you to specify the crosstalk bit sequence. These settings apply only to the Bit-by-bit channel simulation mode and are ignored when the mode is statistical. By default, the crosstalk sequence is random and uncorrelated to the TX sequence, but you have complete freedom in defining your desired crosstalk excitation.

Parameter Name	Description	Units	Default
Uncorrelated to TX	When enabled, automatically chooses a random sequence uncorrelated to TX	None	Yes
User-specified	When enabled, lets you specify a PRBS sequence in several different modes	None	No

Refer to *Tx_SingleEnded* (cktsimchan) for description of PRBS sequence parameters.

Encoder Parameters

By default, the crosstalk source uses the same parameters as TX. Uncheck **Use the same settings as TX** to override any of the properties. For information on encoding options, refer to *Tx_SingleEnded* (cktsimchan).

EQ Parameters

By default, the crosstalk source uses the same equalization as TX. Uncheck **Use the same settings as TX** to use different equalization in the crosstalk driver. For information on EQ options, refer to *Tx_SingleEnded* (cktsimchan).

Electrical Parameters

By default, the crosstalk source has the same electrical properties as TX. Uncheck **Use the same settings as TX** to set different electrical properties for crosstalk. For information on Electrical parameters, refer to *Tx_SingleEnded* (cktsimchan).

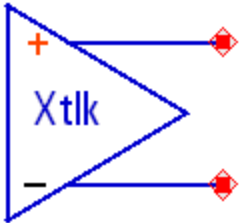
Jitter Parameters

By default, the crosstalk source has the same jitter properties as TX. Uncheck **Use the same settings as TX** to set different jitter properties for crosstalk. For information on jitter parameters, refer to *Tx_SingleEnded* (cktsimchan).

Notes

Xtlk2_Diff (Channel Simulation Crosstalk2 Differential Transmitter)

Symbol

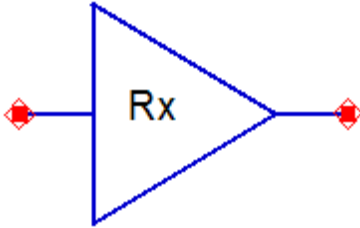


Notes

1. Xtlk2_Diff is a differential crosstalk transmitter.
2. The differential crosstalk transmitter differs from the single-ended crosstalk model only in electrical properties (see *Xtlk2_SingleEnded* (cktsimchan)). A 4-port balun is used for single-ended-to-differential conversion internally to the model. See *Tx_Diff* (cktsimchan) for equivalent circuit.
3. All other parameters are the same as the single-ended version of the crosstalk model.

Rx_SingleEnded (Channel Simulation Single-Ended Receiver)

Symbol



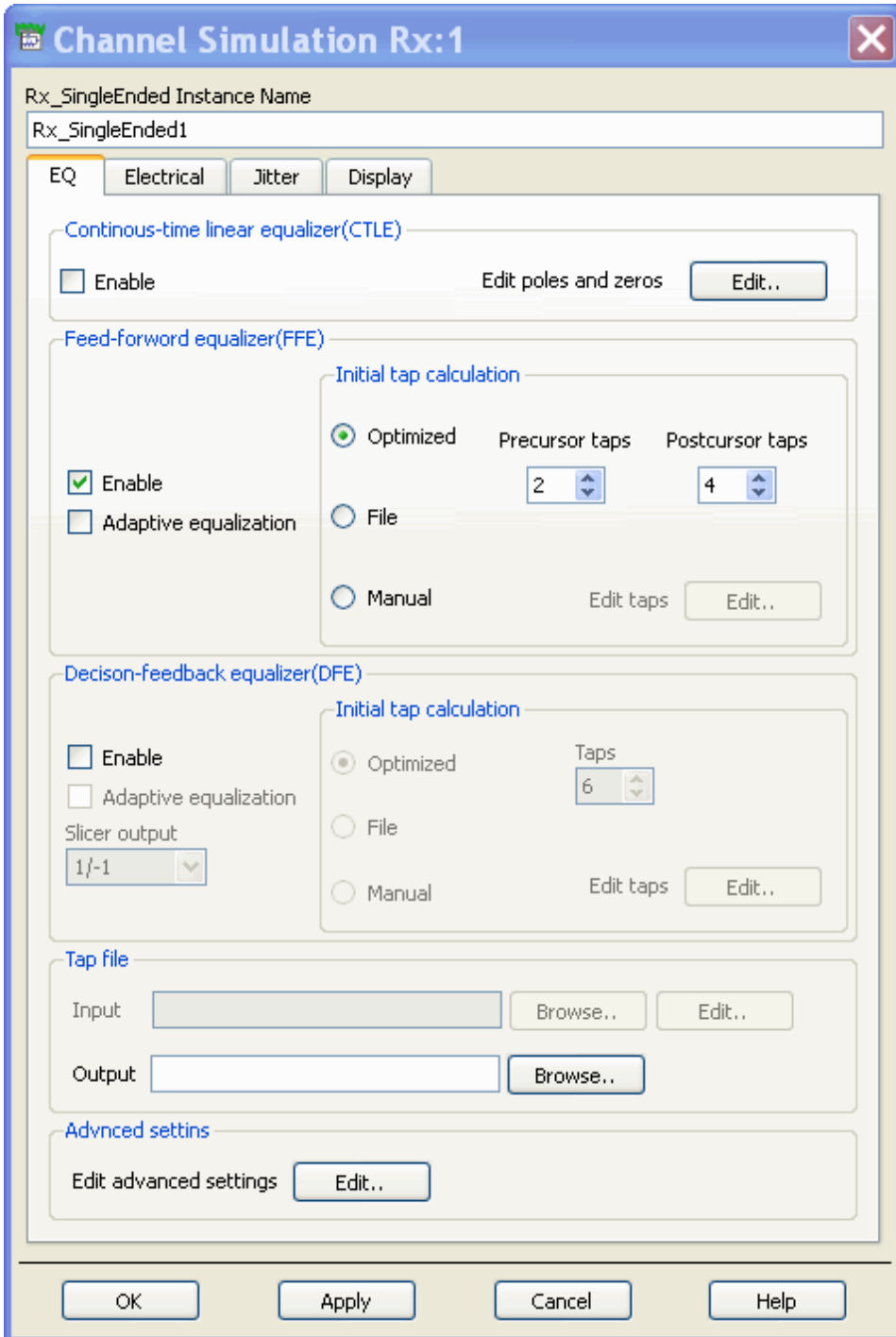
The Rx component models a receiver. It includes equalization capabilities and random jitter injection. It also offers several options for modeling the receiver load. Up to one receiver component may be included in a simulation.


The Rx parameters are organized into the following functional groups enabling you to specify the receiver characteristics:

- **Equalization** specifies continuous-time linear (CTLE), feed-forward (FFE) or decision-feedback (DFE) equalizers.
- **Electrical** sets the receiver load impedance.
- **Jitter** specifies random jitter injected into the Rx.
- **Display** controls the visibility of component parameters on the schematic (see *Displaying Simulation Parameters on the Schematic* (cktsim)).

Equalization Parameters

The Rx model offers equalization choices that include (CTLE), (FFE) and (DFE). Use the EQ tab, then use the *Enable* checkboxes to select an equalizer, as shown below:



 It is possible to select cascade configurations of CTLE-DFE and FFE-DFE.

Rx_SingleEnded EQ CTLE Parameters

To activate CTLE equalization, select *Enable* in the CTLE parameter group. With this option, the RX equalizer is a Laplace-domain linear filter defined by its complex poles and zeros. Click *Edit poles and zeros* to set the CTLE poles and zeros:

The parameters in the CTLE parameter group are described below:

Parameter Name	Description	Units	Default	Notes
Zero[n]	Position of nth zero in s domain, where n starts from 1	rad/s	Empty	
Pole[n]	Position of nth pole in s domain, where n starts from 1	rad/s	Empty	
Pre-factor	Transfer function factor	None	1.0	This is a multiplier in the transfer function. It is not power gain nor gain in dB nor voltage gain.

The filter transfer function is given by:

$$H(s) = \text{Pre-factor} * N(s) / D(s),$$

where,

$$N(s) = (s - \text{Zero}[1]) * (s - \text{Zero}[2]) * \dots$$

$$D(s) = (s - \text{Pole}[1]) * (s - \text{Pole}[2]) * \dots$$

For complex poles and zeros, only one conjugate is specified using ADS complex number notation. For example, to specify the transfer function:

$$H(s) = 1e16 / (s^2 + 2*s*1e8 + 5*1e16), \text{let Pole}[1] = 1e8 * (-1 + 2*j) \text{ or Pole}[1] = 1e8 * (-1 - 2*j).$$

As another example, consider the transfer function:

$$H(s) = N(s)/D(s),$$

where,

$$N(s) = (s + 1e8)/1e8$$

$$D(s) = (s + 1e9)*(s+5e9)/(1e9*5e9)$$

To specify this transfer function, let

$$\text{Zero}[1] = -1e8$$

$$\text{Pole}[1] = -1e9$$

$$\text{Pole}[2] = -5e9$$

$$\text{Pre-factor} = 1e9*5e9/1e8$$

This setting of Pre-factor will yield the desired DC gain of 1. The relationship between the Pre-factor and the complex amplitude DC gain can be derived by setting $s = 0$:

$$H(0) = \text{Pre-factor}*N(0)/D(0)$$

As an alternative to the 1e8-type format, the poles and zeros can be specified as:

$$\text{Zero}[1] = -0.1G$$

$$\text{Pole}[1] = -1G$$

$$\text{Pole}[2] = -5G$$

Poles must be in the left half of the s domain, i.e. the real part must be < 0 .

Rx_SingleEnded EQ FFE Parameters

To activate FFE equalization, select *Enable* in the FFE parameter group. The FFE may be used in fixed and adaptive modes. The FFE is adaptive if *Adaptive equalization* is enabled in the FFE parameter group, otherwise the tap coefficients are fixed throughout a channel analysis. The fixed tap coefficients (or initial tap coefficients in the case of adaptive FFE) may be set in one of three ways:

- Automatically optimized to yield maximum eye opening
- Specified in a file
- Manually specified

The parameters in the FFE parameter group are described below:

Parameter Name	Description	Units	Default
Enable	Enables FFE equalization. This parameter is inactive if CTLE is enabled, but it may be used in tandem with DFE	None	No
Adaptive equalization	Turns on adaptive FFE	None	No
Optimized	Taps are automatically optimized to maximize eye opening	None	Yes
Precursor taps	The number of pre-cursor taps in Optimized FFE mode	None	2
Postcursor taps	The number of post-cursor taps in Optimized FFE mode	None	4
File	Taps will be read from a file	None	No
Manual	Taps are manually specified	None	No

To manually specify FFE parameters, click *Edit taps* and enter pre and post-cursor tap coefficients. For more information on file-based tap specification, refer to [File-based tap specification](#) below. For more information on adaptive equalization settings, refer to [Advanced equalization parameters](#).

Rx_SingleEnded EQ DFE Parameters

DFE equalization parameters mirror FFE. DFE may be used in tandem with CTLE or FFE, and works in fixed and adaptive modes. To activate DFE, select *Enable* in the DFE parameter group. The DFE will adapt if *Adaptive equalization* is enabled, otherwise the tap coefficients will be fixed throughout a channel analysis run. As with FFE, you can set DFE coefficients using automatic tap optimization, from file, or manually. The parameters in the DFE parameter group are described below:

Parameter Name	Description	Units	Default
Enable	Enables DFE equalization. You can use DFE in combination with CTLE or FFE	None	No
Adaptive equalization	Turns on adaptive DFE	None	No
Slicer output	Sets DFE slicer output to 1/-1 or 1/0	None	1/-1
Optimized	Taps are automatically optimized to maximize eye opening	None	Yes
Taps	The number of DFE taps in Optimized DFE mode	None	6
File	Taps will be read from a file	None	No
Manual	Taps are manually specified	None	No

File-based Tap Specification

The starting FFE/DFE tap coefficients may be specified in text files. In addition, the calculated FFE/DFE taps may be stored in files for re-use. Storing tap coefficients in a file is useful, for example, when the optimized FFE/DFE mode is used to calculate the taps.

To read tap coefficients from a file, select the file-based FFE or DFE mode and click *Browse* next to *Input* tap file parameter to select a file. Tap files use a simple format, an example of which is shown below:

```
PreCursor
-3.292119
PostCursor
6.592708
```

-2.806157
 0.505568
 DFETap
 0.003872
 0.007379
 -0.010099
 -0.014041

To specify pre-cursor FFE taps, list them after the keyword *PreCursor*. To specify post-cursor FFE taps, list them after the keyword *PostCursor*. DFE taps use the keyword *DFETap*. This example uses a file-based FFE-DFE combination. If FFE or DFE are not in use, or they don't use the file-based tap specification mode, the corresponding sections are ignored and may be omitted.

To store FFE or DFE taps, specify a file name in *Output* box of the Tap file parameter group. This can be used to store the calculated taps in the Optimized tap mode, or to store the final set of taps when using equalizers in adaptive mode.

Advanced equalization parameters

Parameter Name	Description	Units	Default
AdaptiveAlgorithm	Adaptive Algorithm	None	No
Alpha	step size for LMS and ZF algorithm	None	1e-3
Lambda	weighting factor for RLS algorithm	None	0.999
Delta	small positive constant for RLS algorithm	None	1e-3
TargetMSE	reference MSE in dB for stopping updating coefficients when RLS equalizer reaches this MSE	dB	-40
WaitingPeriod	Waiting Period for adaptive equalization	bit	100

If the adaptive equalization is selected, the equalizer taps will be adapted during the simulation based on the *AdaptiveAlgorithm*. The error signal is from the decision signal of

the equalized signal: $e_k = \tilde{I}_k - \hat{I}_k$

where,

\tilde{I}_k is the detected output sequence from DFE slicer

\hat{I}_k is the equalized output sequence.

LMS Algorithm

The criterion most commonly used in the optimization of the equalizer coefficients is the minimization of the mean square error (MSE) between the desired equalizer output and the actual equalizer output. MSE minimization can be accomplished recursively by use of the stochastic gradient algorithm introduced by Widrow, called the LMS algorithm. This algorithm is described by the coefficient update equation

$$C_{k+1} = C_k + \alpha e_k X_k$$

where

C_k is the vector of the equalizer taps at the kth iteration

X_k represents the equalizer signal vector.

α is parameter *Alpha*.

RLS Algorithm

The convergence rate of the LMS algorithm is slow because a single parameter α controls the rate of adaptation. A fast converging algorithm is obtained if a recursive least squares (RLS) criterion is adopted for adjustment of the equalizer coefficients.

The RLS iteration algorithm follows.

Calculate Kalman gain vector:

$$K_k = \frac{P_{k-1} \times X_k}{\lambda + X_k^T \times P_{k-1} \times X_k}$$

Update inverse of the correlation matrix:

$$P_k = \frac{1}{\lambda} \times \left[P_{k-1} - K_k \times X_k^T \times P_{k-1} \right]$$

Update coefficients:

$$C_k = C_{k-1} + P_k \times X_k \times e_k$$

where

λ is parameter Lamda

P_k is a diagonal matrix with initial value $\Delta \times I$ (where I is a diagonal matrix).

Δ is parameter *Delta*.

The updating of coefficients in RLS algorithm will be halted when the MSE averaged over 100 consecutive symbols is less than a reference MSE defined by *TargetMSE*.

ZF Algorithm

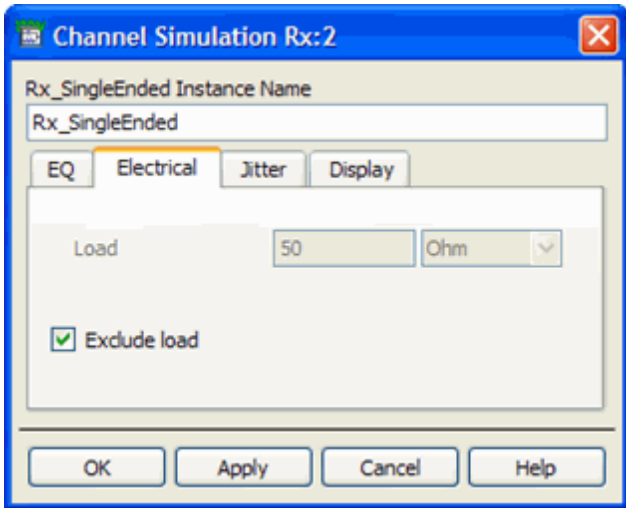
The zero-forcing (ZF) solution is achieved by forcing the cross-correlation between the error sequence and the desired information sequence to be zero.

The coefficients are updated as: $C_{k+1} = C_k + \alpha e_k \tilde{I}_k$
where

\tilde{I}_k is the detected output sequence.

Electrical Parameters

Use the Electrical tab to set the receiver load impedance.



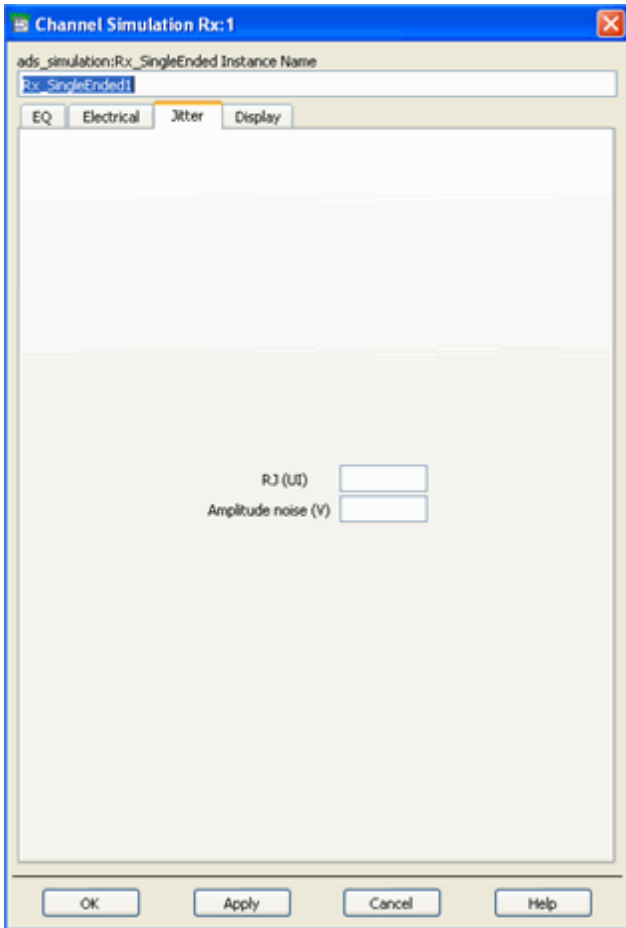
Rx_SingleEnded Electrical Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
Load	Load	Specify a finite load impedance	Ohm	50
Exclude load	ExcludeLoad	Select <i>Exclude load</i> (<i>ExcludeLoad=yes</i>) to exclude the loading effects of the Rx (infinite resistance to ground).	None	Yes

By default, the Rx component is electrically open-circuited. Optionally, you may include an arbitrary resistance from the input terminal to ground.

Jitter Parameter

Use the Jitter tab to set the optional RJ injection.

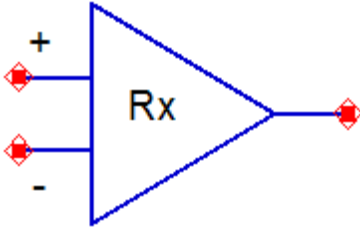


Rx_SingleEnded Jitter Parameter

Setup Dialog Name	Parameter Name	Description	Units	Default
RJ (UI)	RJ	Random jitter as a fraction of UI	None	0.0
Amplitude noise (V)	AmpNoise	RMS of amplitude noise in V	V	0 V

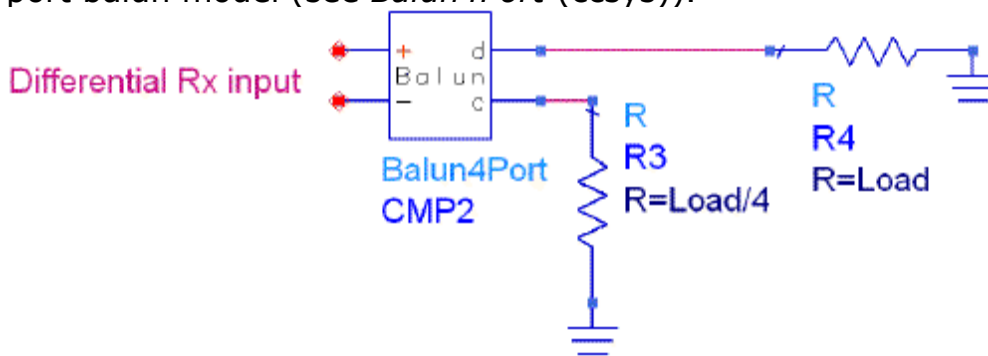
Rx_Diff (Channel Simulation Differential Receiver)

Symbol



Notes

1. Rx_Diff is a differential receiver (Rx) model.
2. All Rx_Diff model parameters are the same as the single-ended version of the Rx model (see *Rx_SingleEnded* (cktsimchan)).
3. The differential load is modeled as shown in the following figure, using the ADS 4-port balun model (see *Balun4Port* (ccsys)).



Term_SingleEnded (Channel Simulation Single-Ended Termination)

Symbol



Parameters

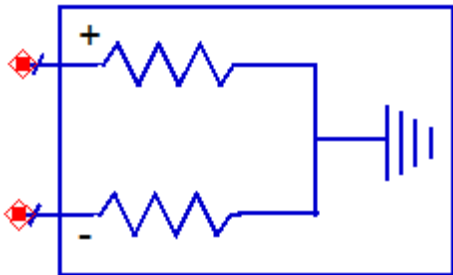
Setup Dialog Name	Parameter Name	Description	Units	Default
Load	Load	Resistive load to ground	Ohm	50.0

Notes

1. The Term_SingleEnded model is a single-ended termination component used to supply a resistive load to ground.
2. Term_SingleEnded is provided for convenience only. Other resistive models, such as *R* and *Term*, may also be used to terminate the channel.

Term_Diff (Channel Simulation Differential Termination)

Symbol

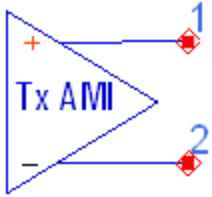


Notes

1. Term_Diff is a differential termination used to supply a resistive load to ground.
2. The Term_Diff model's parameter (*Load*) is the same as the *Load* parameter used in the Term_SingleEnded model (see *Term_SingleEnded* (cktsimchan)).
3. The electrical equivalent of this model is the same as that of the Rx_Diff model (see *Rx_Diff* (cktsimchan)).

Tx_AMI (IBIS-AMI Transmitter)

Symbol



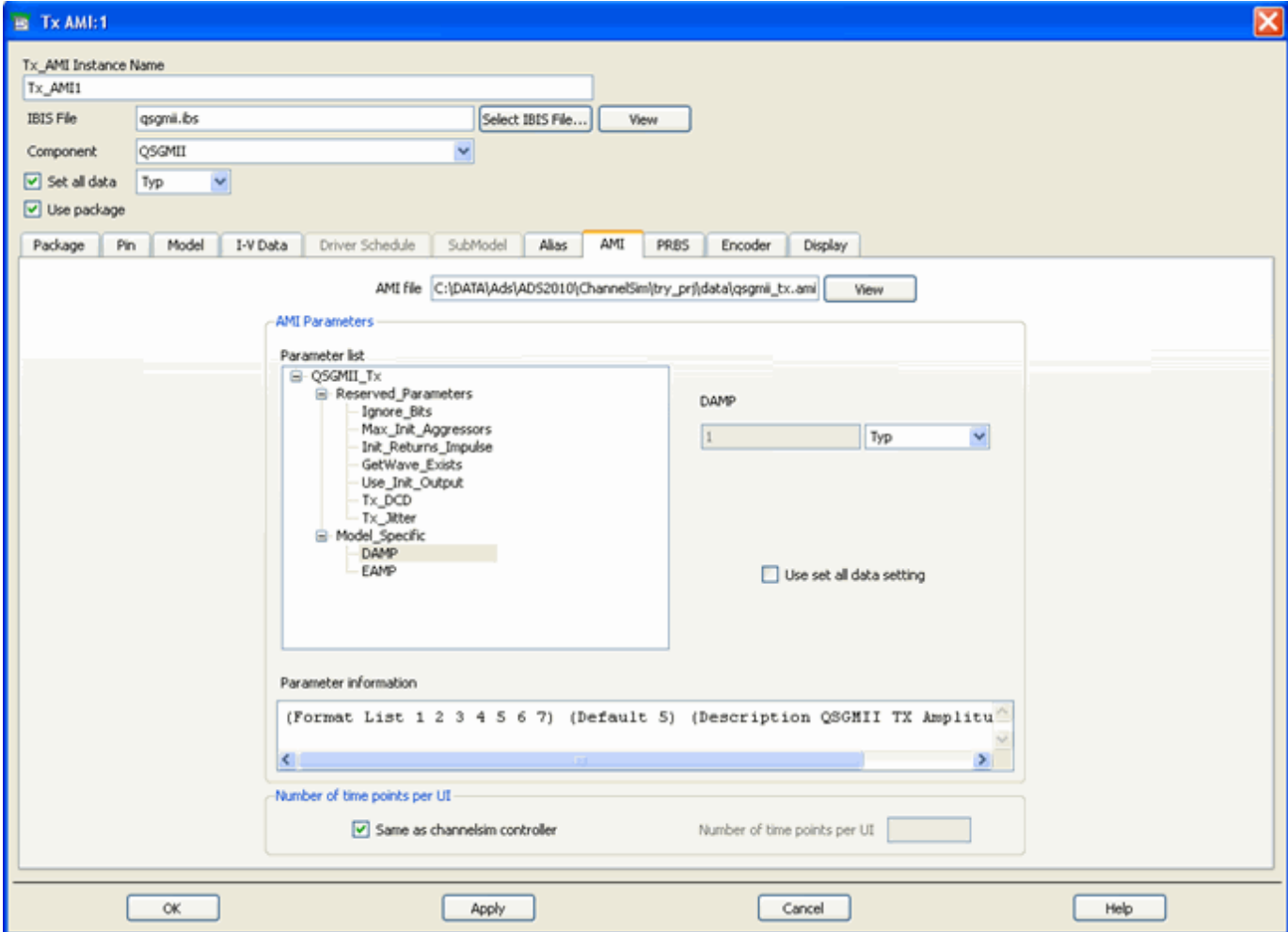
The Tx_AMI component encompasses the functionalities of a PRBS source, encoder, and an IBIS component. The source produces a digital signal of user-specified bit sequence. The *IBIS Models* (ibis) provides the transmitter's back-end analog behavioral model and must include the [Algorithmic Model] keyword for the selected IBIS model. The [Algorithmic Model] keyword contains details about the AMI parameter file and the shared object library file, both provided by the transmitter's vendor. During simulation, the transmitter equalization filter behavior is controlled by the shared object library.

The Tx_AMI parameters are organized into the following functional groups enabling you to specify the transmitter characteristics:

- **IBIS** tabs and more fields in the Tx_AMI dialog allow you to select IBIS file, component, pin, model, as well as desired corner values in the same way as for any other IBIS component. See *IBIS Models* (ibis) for more details.
- **AMI** tab in the Tx_AMI dialog allows you to view all the AMI parameters. Some of the AMI parameters can be modified, while others cannot.
- **PRBS** specifies the Tx_AMI excitation source.
- **Encoder** selects 8B/10B or no encoding.
- **Display** controls the visibility of component parameters on the schematic (see *Displaying Simulation Parameters on the Schematic* (cktsim)).

AMI Parameters

Use the AMI tab to view or set AMI parameters:

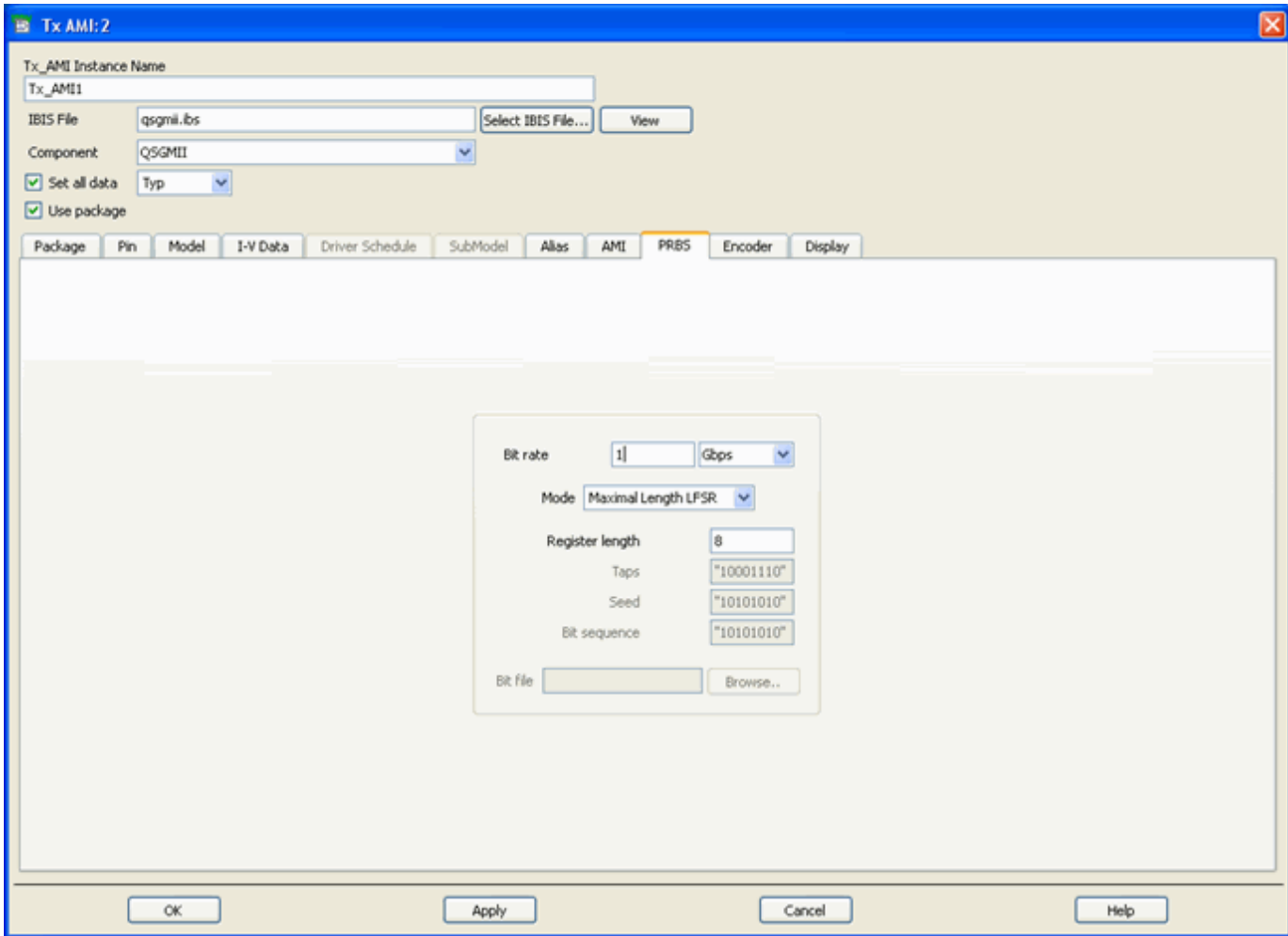


Tx_AMI AMI Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
Highlighted item in the Parameter list window	AMIvarName[i]	As defined in the AMI file. Index 'i' refers to the order in the AMI file.	As defined in the AMI file.	As defined in the AMI file.
Highlighted item in the Parameter list window	AMIvarValue[i]	As defined in the AMI file. Index 'i' refers to the order in the AMI file.	As defined in the AMI file.	As defined in the AMI file.
Number of time points per UI: Same as channelsim controller	UseControllerSampleRate	Flag to inherit the value of Number of time points per UI as set in the Advanced Convolution Option dialog in the ChannelSim controller.	None	yes
Number of time points per UI	NumberTimePtPerUI	Number of time points per unit interval – takes effect only if the Same as channelsim controller box is not checked.	None	32

PRBS Parameters

Use the PRBS tab to specify the Tx_AMI excitation source in several different formats:

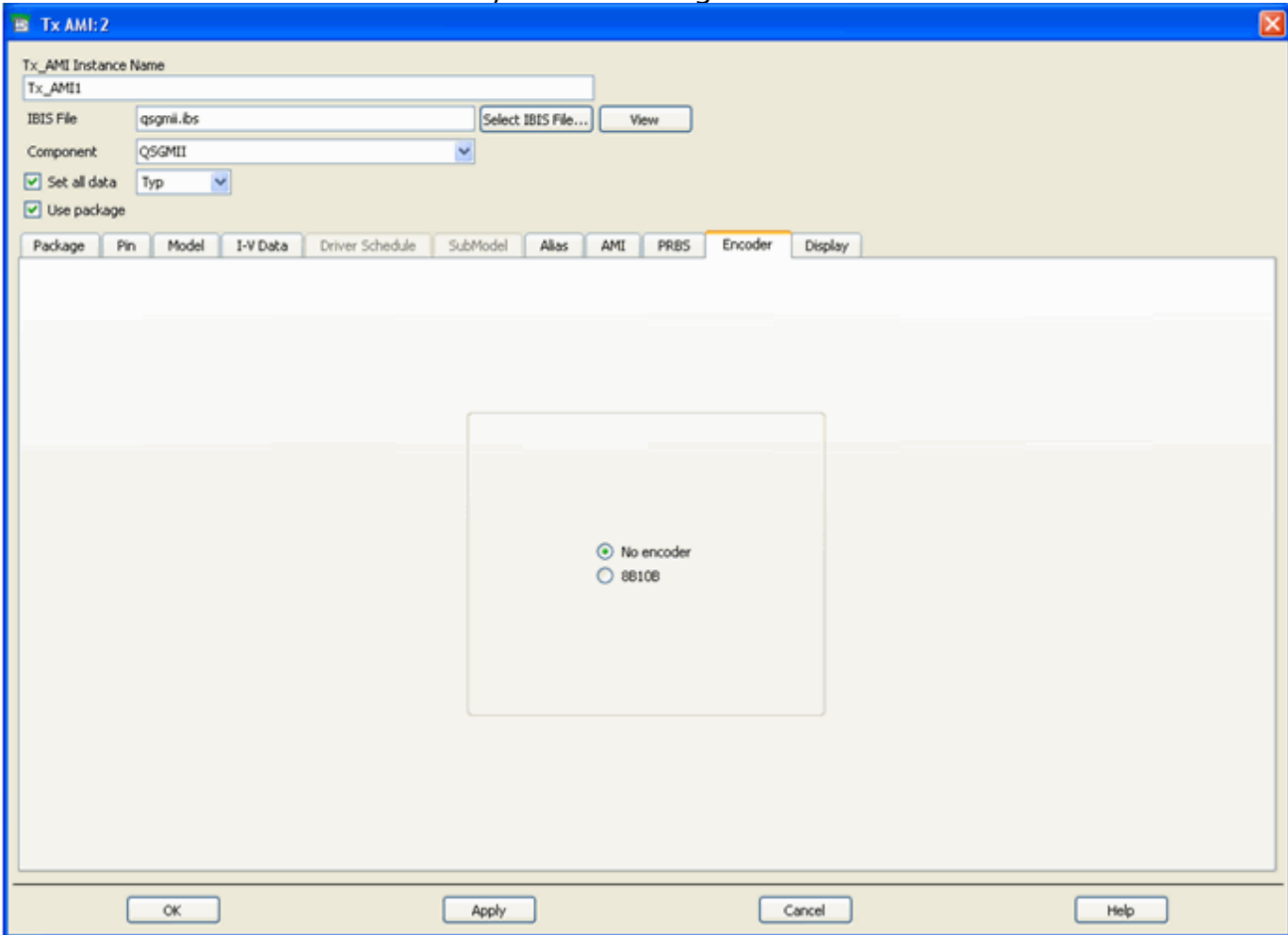


Tx_AMI PRBS Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
Bit rate	BitRate	Data rate of the PRBS source. This parameter also establishes the simulation data rate.	bps	1 Gbps
Mode	Mode	PRBS pattern mode. Select one of the following: <ul style="list-style-type: none"> Maximal Length LFSR User Defined LFSR User Defined Sequence Bit File 	None	Maximal Length LFSR
Register length	RegisterLength	Length of shift register, in bits, in maximal length mode.	None	8
Taps	Taps	LFSR taps in user-defined LFSR mode.	None	"10001110"
Seed	Seed	LFSR seed in user-defined LFSR mode.	None	"10101010"
Bit sequence	BitSequence	Specifies a user-defined bit sequence. The pattern repeats if the simulation extends beyond the length of the user-defined sequence.	None	"10101010"
Bit file	BitFile	The pattern is specified in a text file. The sequence is specified as an array of 1s and 0s in row or column format.	None	None

Encoder Parameters

Use the Encoder tab to select 8B/10B encoding:



Tx_AMI Encoder Parameters

Setup Dialog Name	Parameter Name	Description
No encoder	Encoder=No encoder	Select this option if no encoding is to be applied to the PRBS sequence (See Note 7).
8B10B	Encoder=8B10B	Select this option to apply 8B/10B encoding to the PRBS sequence.

Notes/Equations

1. The Tx_AMI component requires an IBIS file and, within that file, the keyword [Algorithmic Model] must be specified for the selected IBIS component/pin/model.
2. The **AMI file** field and the **View** button are provided for user convenience only. The AMI file is specified in the IBIS file and this field is not editable.
3. Only *In* and *InOut* types of AMI parameters are sent back to the simulator. The values of those parameters can be modified by the user.
4. See *Channel Simulation Controller AMI Simulation Setup* (cktsimchan) for more details on how to set the values of AMI parameters, what options are available, etc.
5. The source jitter is controlled by a predefined AMI parameter *Tx_Jitter*. If the

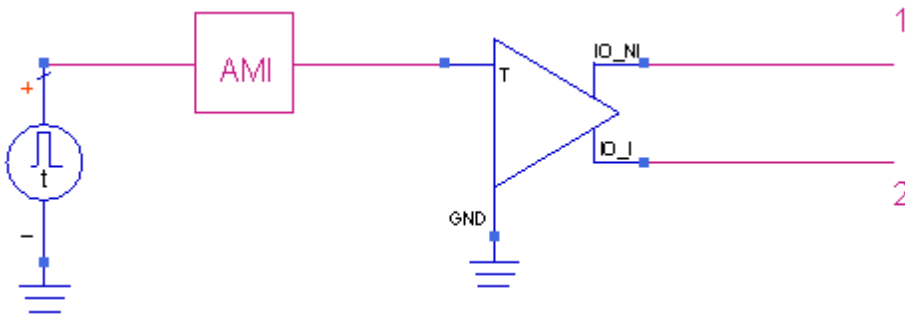
parameter is not present then no jitter is added.

6. The source duty-cycle distortion is controlled by a predefined AMI parameter Tx_DCD . If the parameter is not present then no distortion is assumed.
7. By default, no encoding is applied to the PRBS sequence (the **No encoder** option radio button is selected).

Circuit Diagram

The Tx_AMI component schematic diagram is shown below.

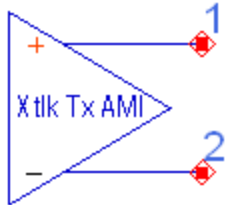
Figure: Tx_AMI schematic diagram



The IBIS component inside of Tx_AMI is a differential IBIS transmitter buffer. Any of the supported output type buffers can be used. The buffer common-mode is zero. Enabling of I/O and tri-state buffers and providing power supply are handled internally.

XtlkTx_AMI (IBIS-AMI Crosstalk Transmitter)

Symbol



The XtlkTx_AMI component encompasses the functionalities of a PRBS source, encoder, and an IBIS component. The source produces a digital signal of user-specified bit sequence. The *IBIS Models* (ibis) provides the transmitter's back-end analog behavioral model and must include the [Algorithmic Model] keyword for the selected IBIS model. The [Algorithmic Model] keyword contains details about the AMI parameter file and the shared object library file, both provided by the transmitter's vendor. During simulation, the transmitter equalization filter behavior is controlled by the shared object library.

XtlkTx_AMI components are very similar to the Tx_AMI components. Unlike Tx_AMI, which is required and must be unique on the schematic, XtlkTx_AMI drivers are optional. In addition, multiple XtlkTx_AMI drivers are supported. However, there is a limit to the number of XtlkTx_AMI components allowed in AMI simulation. The limit is established by the value of the AMI reserved parameter "Max_Init_Aggressors" in the AMI file of the Rx_AMI. The default is zero.

By default, the XtlkTx_AMI components inherit most of their properties from the Tx_AMI used in the circuit, including PRBS parameters and encoding. Those properties may be overridden, as shown in the parameter tables below.

The Channel Simulator models both synchronous and asynchronous crosstalk. For synchronous crosstalk, the crosstalk excitation has a fixed phase relationship to the driver. In AMI methodology the synchronous crosstalk has an identical bit rate to the main channel, and the asynchronous crosstalk has a different bit rate.

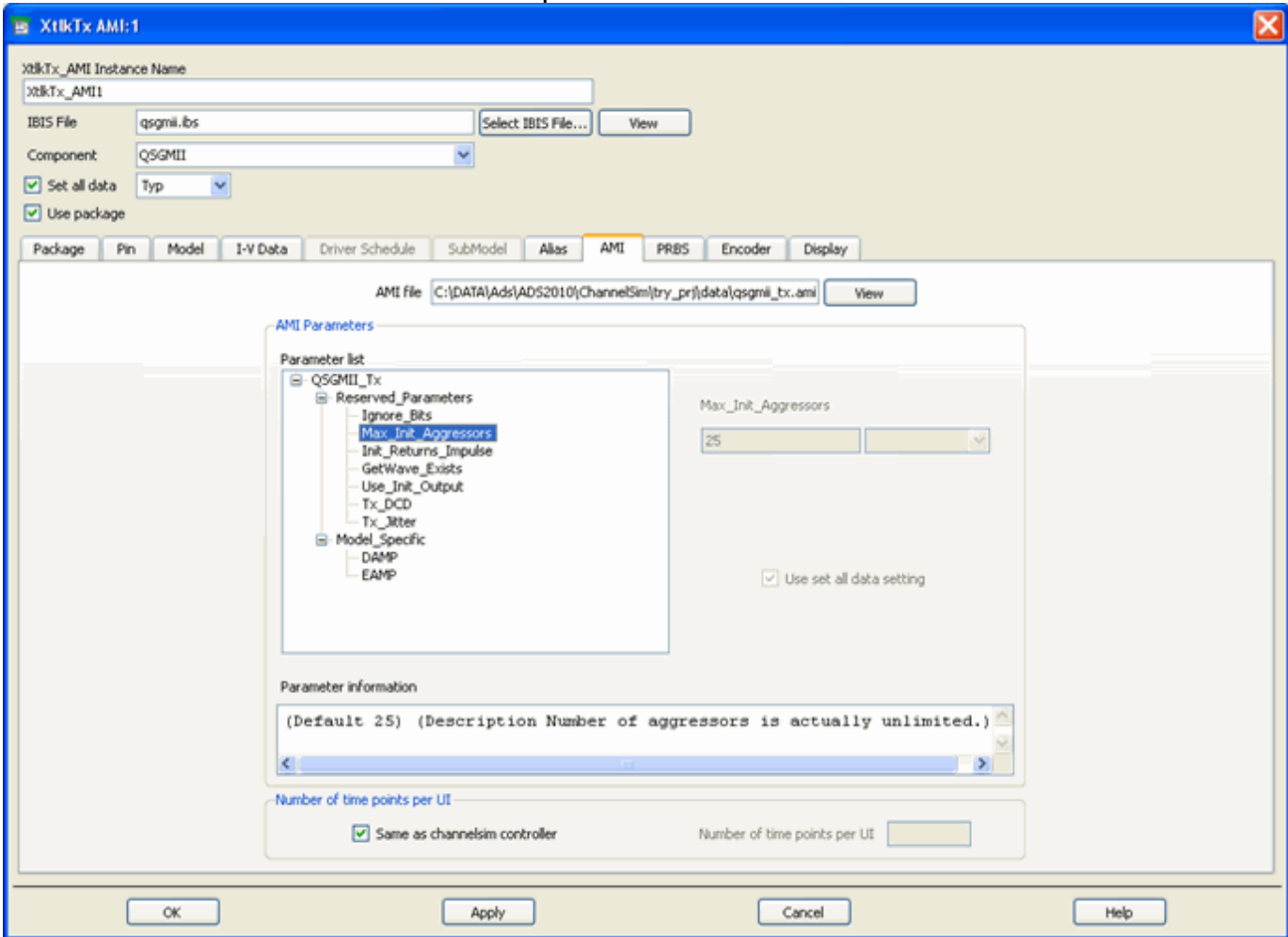
The XtlkTx_AMI parameters are organized into the following functional groups enabling you to specify the transmitter characteristics:

- **IBIS** tabs and more fields in the XtlkTx_AMI dialog allow you to select IBIS file, component, pin, model, as well as desired corner values in the same way as for any other IBIS component. See *IBIS Models* (ibis) for more details.
- **AMI** tab in the XtlkTx_AMI dialog allows you to view all the AMI parameters. Some of the AMI parameters can be modified, while others cannot.
- **PRBS** specifies the XtlkTx_AMI excitation source.
- **Encoder** selects 8B/10B or no encoding.
- **Display** controls the visibility of component parameters on the schematic (see *Displaying Simulation Parameters on the Schematic* (cktsim)).

Note
 Except for PRBS synchronization properties, all the parameter tabs are almost identical to those for the Tx_AMI component.

AMI Parameters

Use the AMI tab to view or set AMI parameters:

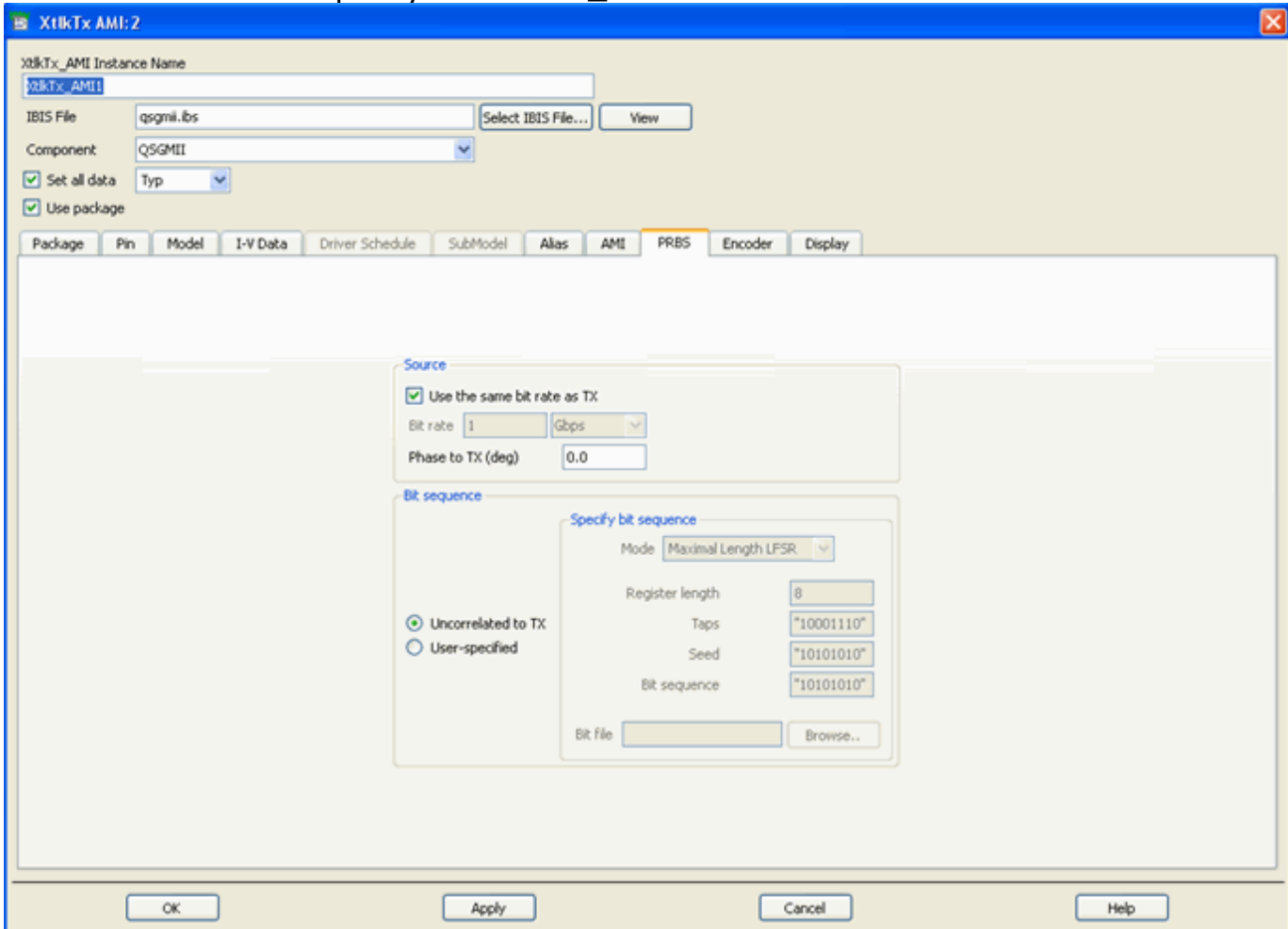


XtlkTx_AMI AMI Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
Highlighted item in the Parameter list window	AMIVarName[i]	As defined in the AMI file. Index 'i' refers to the order in the AMI file.	As defined in the AMI file.	As defined in the AMI file.
Highlighted item in the Parameter list window	AMIVarValue[i]	As defined in the AMI file. Index 'i' refers to the order in the AMI file.	As defined in the AMI file.	As defined in the AMI file.
Number of time points per UI: Same as channelsim controller	UseControllerSampleRate	Flag to inherit the value of Number of time points per UI as set in the Advanced Convolution Option dialog in the ChannelSim controller.	None	yes
Number of time points per UI	NumberTimePtPerUI	Number of time points per unit interval – takes effect only if the Same as channelsim controller box is not checked.	None	32

PRBS Parameters

Use the PRBS tab to specify the XtkTx_AMI excitation source in several different formats:

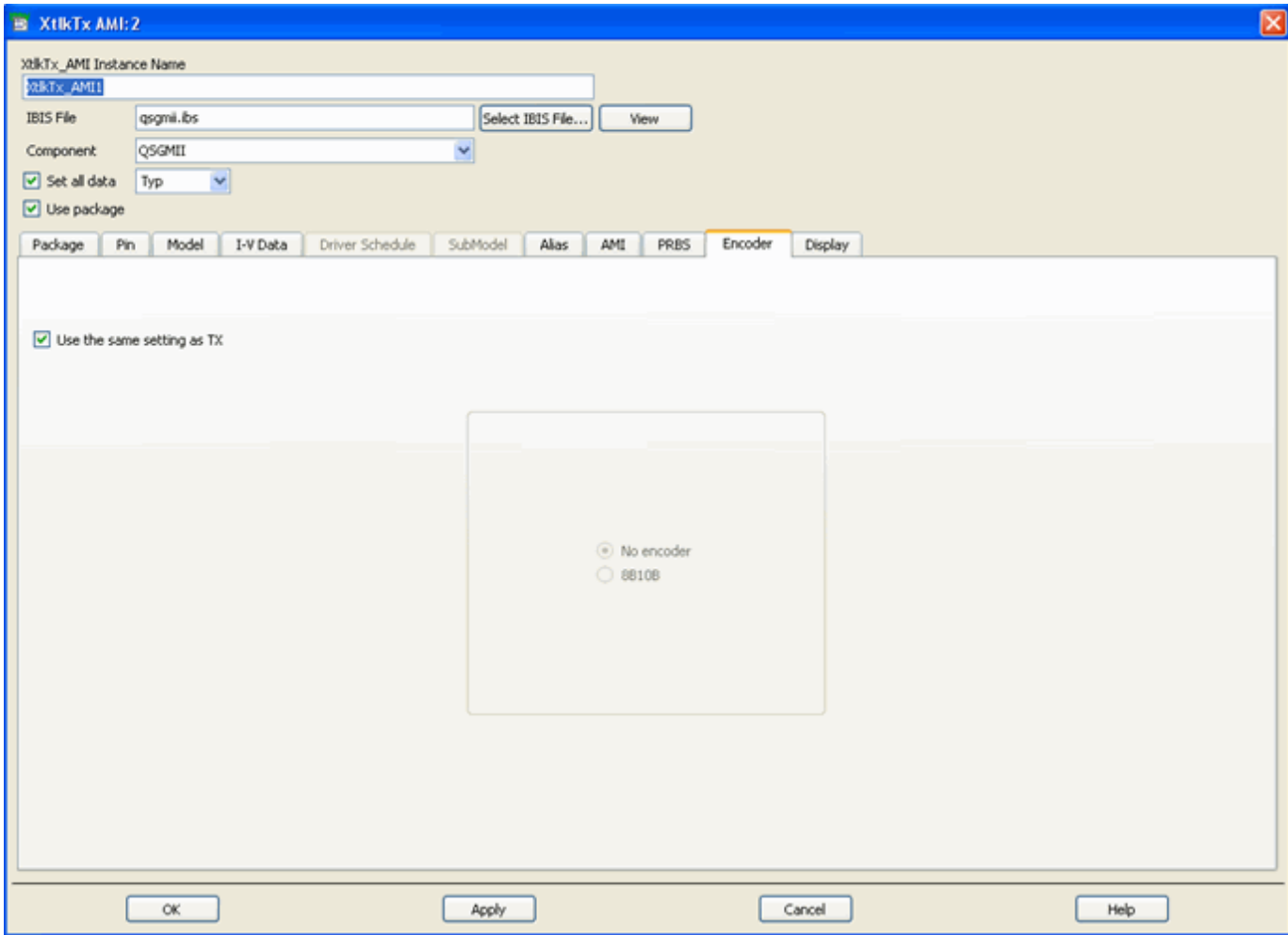


XtkTx_AMI PRBS Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
Use the same bit rate as TX	SameSourceSettingAsTx	Flag to inherit the bit rate from Tx_AMI.	None	yes
Bit rate	BitRate	Data rate of the PRBS source. This parameter can be set if the box "Use the same bit rate as TX" is unchecked.	bps	1 Gbps
Phase to TX (deg)	PhaseToTx	Fixed phase offset (in degrees) between this crosstalk source and the Tx_AMI source (1UI = 360 degrees).	degrees	0.0
Bit Sequence: Uncorrelated to TX	BitSequenceMode = Uncorrelated to TX	When enabled, automatically chooses a random sequence uncorrelated to TX	None	Yes
Bit Sequence: User-specified	BitSequenceMode = User specified	When enabled, lets you specify a PRBS sequence in several different modes	None	No
Mode	PRBSMode	PRBS pattern mode. Select one of the following: <ul style="list-style-type: none"> Maximal Length LFSR User Defined LFSR User Defined Sequence Bit File 	None	Maximal Length LFSR
Register length	RegisterLength	Length of shift register, in bits, in maximal length mode.	None	8
Taps	Taps	LFSR taps in user-defined LFSR mode.	None	"10001110"
Seed	Seed	LFSR seed in user-defined LFSR mode.	None	"10101010"
Bit sequence	BitSequence	Specifies a user-defined bit sequence. The pattern repeats if the simulation extends beyond the length of the user-defined sequence.	None	"10101010"
Bit file	BitFile	The pattern is specified in a text file. The sequence is specified as an array of 1s and 0s in row or column format.	None	None

Encoder Parameters

Use the Encoder tab to select 8B/10B encoding.



XtlkTx_AMI Encoder Parameters

Setup Dialog Name	Parameter Name	Description	Default
Use the same setting as TX	SameEncoderSettingAsTx	Flag to use the same setting as the one selected for the main channel IBIS AMI transmitter Tx_AMI.	yes
No encoder	Encoder=No encoder	Select this option if no encoding is to be applied to the PRBS sequence. Takes effect only if the box "Use the same setting as TX" is not checked.	See Note 8
8B10B	Encoder=8B10B	Select this option to apply 8B/10B encoding to the PRBS sequence. Takes effect only if the box "Use the same setting as TX" is not checked.	See Note 8

Notes/Equations

1. The XtlkTx_AMI component requires an IBIS file and, within that file, the keyword [Algorithmic Model] must be specified for the selected IBIS component/pin/model.
2. The **AMI file** field and the **View** button are provided for user convenience only. The AMI file is specified in the IBIS file and this field is not editable.
3. Only *In* and *InOut* types of AMI parameters are sent back to the simulator. The values of those parameters can be modified by the user.
4. See *Channel Simulation Controller AMI Simulation Setup* (cktsimchan) for more details on how to set the values of AMI parameters, what options are available, etc.
5. The source jitter is controlled by a predefined AMI parameter *Tx_Jitter*. If the

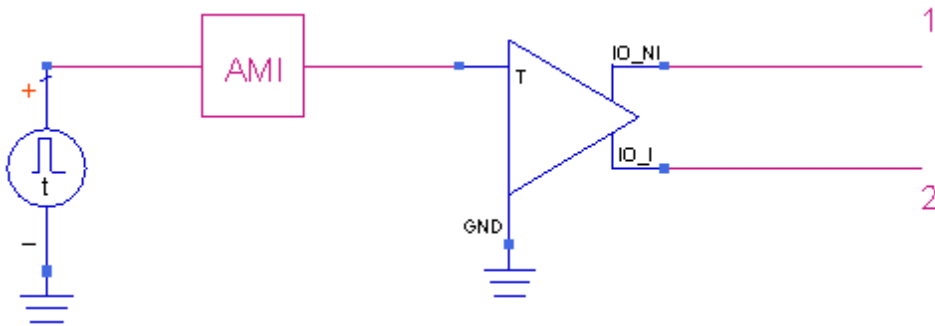
parameter is not present then no jitter is added.

6. The source duty-cycle distortion is controlled by a predefined AMI parameter Tx_DCD . If the parameter is not present then no distortion is assumed.
7. If the box **Use the same bit rate as TX** is checked, or if the **Bit rate** is specified as equal to that of the Tx_AMI component then the crosstalk is synchronous.
8. If you want to model asynchronous crosstalk with the same bit rate as Tx_AMI, add a small offset to this component's bit rate after un-checking the box **Use the same bit rate as TX**.
9. If the **Use the same setting as TX** option is unchecked, no encoding is applied to the PRBS sequence by default (the **No encoder** option radio button is selected).

Circuit Diagram

The XtlkTx_AMI component schematic diagram is shown below.

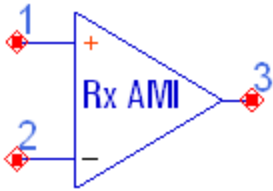
Figure: XtlkTx_AMI schematic diagram



The IBIS component inside of XtlkTx_AMI is a differential IBIS transmitter buffer. Any of the supported output type buffers can be used. The buffer common-mode is zero. Enabling of I/O and tri-state buffers and providing power supply are handled internally.

Rx_AMI (IBIS-AMI Receiver)

Symbol



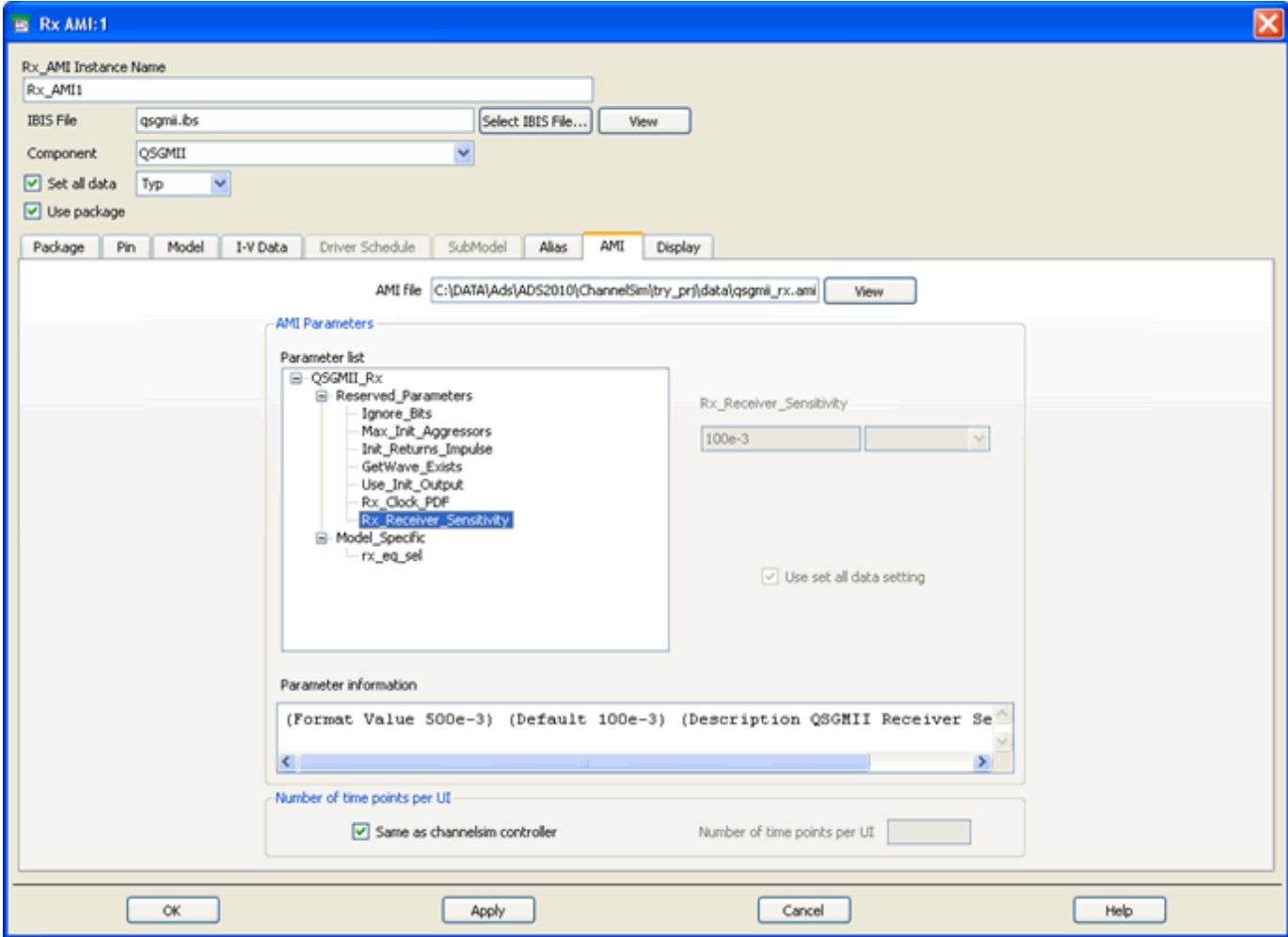
The Rx_AMI component encompasses the functionalities of an IBIS component and digital signal processing of the receiver. The *IBIS model* (ibis) provides the receiver's front-end analog behavioral model and must include the [Algorithmic Model] keyword for the selected IBIS model. The [Algorithmic Model] keyword contains details about the AMI parameter file and the shared object library file, both provided by the receiver's vendor. During simulation, the receiver equalization and clock data recovery behavior is controlled by the shared object library.

The Rx_AMI parameters are organized into the following functional groups enabling you to specify the receiver characteristics:

- **IBIS** tabs and more fields in the Rx_AMI dialog allow you to select IBIS file, component, pin, model, as well as desired corner values in the same way as for any other IBIS component. See *IBIS Models* (ibis) for more details.
- **AMI** tab in the Rx_AMI dialog allows you to view all the AMI parameters. Some of the AMI parameters can be modified, while others cannot.
- **Display** controls the visibility of component parameters on the schematic (see *Displaying Simulation Parameters on the Schematic* (cktsim)).

AMI Parameters

Use the AMI tab to view or set AMI parameters



Rx_AMI AMI Parameters

Setup Dialog Name	Parameter Name	Description	Units	Default
Highlighted item in the Parameter list window	AMIVarName[i]	As defined in the AMI file. Index 'i' refers to the order in the AMI file.	As defined in the AMI file.	As defined in the AMI file.
Highlighted item in the Parameter list window	AMIVarValue[i]	As defined in the AMI file. Index 'i' refers to the order in the AMI file.	As defined in the AMI file.	As defined in the AMI file.
Number of time points per UI: Same as channelsim controller	UseControllerSampleRate	Flag to inherit the value of Number of time points per UI as set in the Advanced Convolution Option dialog in the ChannelSim controller.	None	yes
Number of time points per UI	NumberTimePtPerUI	Number of time points per unit interval – takes effect only if the Same as channelsim controller box is not checked.	None	32

Notes/Equations

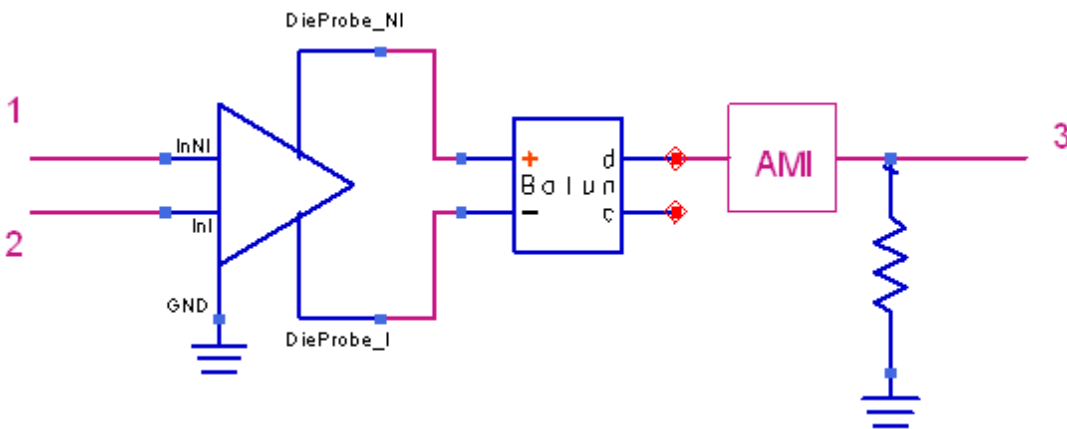
1. The Rx_AMI component requires an IBIS file and, within that file, the keyword [Algorithmic Model] must be specified for the selected IBIS component/pin/model.
2. The **AMI file** field and the **View** button are provided for user convenience only. The AMI file is specified in the IBIS file and this field is not editable.

3. Only *In* and *InOut* types of AMI parameters are sent back to the simulator. The values of those parameters can be modified by the user.
4. See *Channel Simulation Controller AMI Simulation Setup* (cktsimchan) for more details on how to set the values of AMI parameters, what options are available, etc.

Circuit Diagram

The Rx_AMI component schematic diagram is shown below.

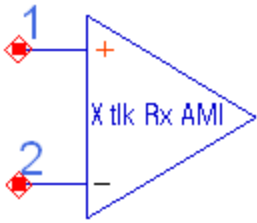
Figure: Rx_AMI schematic diagram



The IBIS component inside of Rx_AMI is a differential IBIS receiver buffer. Any of the supported input/terminator type buffers can be used. The buffer common-mode is zero. Disabling of I/O buffers and providing power supply are handled internally. The ADS 4-port balun component (see *Balun4Port* (ccsys)) converts the IBIS on-chip plus/minus voltages into a differential signal, which after AMI processing becomes available for being probed by an Eye_Probe component.

XtlkRx_AMI (IBIS-AMI Receiver)

Symbol



The XtlkRx_AMI component is simply just an IBIS component. The *IBIS model* (ibis) may or may not include the [Algorithmic Model] keyword for the selected IBIS model. The [Algorithmic Model] keyword, if present, is ignored. It is not required that the crosstalk channel is terminated with the XtlkRx_AMI component.

The XtlkRx_AMI parameters are organized into the following functional groups enabling you to specify the transmitter characteristics:

- **IBIS** tabs and more fields in the XtlkRx_AMI dialog allow you to select IBIS file, component, pin, model, as well as desired corner values in the same way as for any other IBIS component. See *IBIS Models* (ibis) for more details.
- **Display** controls the visibility of component parameters on the schematic (see *Displaying Simulation Parameters on the Schematic* (cktsim)).

The IBIS component inside of XtlkRx_AMI is a differential IBIS receiver buffer. Any of the supported input/terminator type buffers can be used. The buffer common-mode is zero. Disabling of I/O buffers and providing power supply are handled internally.