Agilent Technologies

Advanced Design System 2011.01

Feburary 2011
Sinks

Acknowledgments

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.

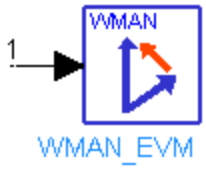
Errata The ADS product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsof" is now part of Agilent Technologies and is known as "Agilent EEsof". To avoid broken functionality and to maintain backward compatibility for 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).

WMAN_EVM



Description: WMAN EVM measurement
Library: Sinks
Class: TSDF_WMAN_EVM

Parameters

Name	Description	Default	Unit	Type	Range
RLoad	load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	physical temperature, in degrees C, of load resistance. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
FCarrier	carrier frequency	1.9 GHz	Hz	real	(0, ∞)
Start	start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
AverageType	average type: Off, RMS (Video)	RMS (Video)		enum	
FramesToAverage	number of frames that will be averaged if AverageType is RMS (Video)	20		int	[1, ∞)
DataSubcarrierModulation	modulation format of the data subcarriers: Auto Detect, FCH, BPSK, QPSK, QAM 16, QAM 64	Auto Detect		enum	
GuardInterval	guard interval time, expressed as a fraction of the FFT time length	0.25		real	[0, 1]
NominalBandwidth	nominal channel bandwidth of the input signal	7.0 MHz	Hz	real	[2, 370e6]
FsBW_Ratio	ratio between the FFT sampling frequency and the nominal bandwidth: Auto, _57/50, _8/7, _86/75, _316/275, _144/125, Other	Auto		enum	
OtherFsBW_Ratio	sampling frequency to nominal bandwidth ratio when FsBW_Ratio is set to Other	8/7		real	[0.5, 2]

SearchLength	search length	4.8 msec	sec	real	(0, ∞)
PulseSearch	search for burst in the input signal: NO, YES	YES		enum	
ResultLengthType	used together with ResultLength parameter, see help for details: Auto Select, Try FCH, Manual Override	Auto Select		enum	
ResultLength	in terms of symbol-times, see help for details	60		int	[1, 1367]
MeasurementOffset	measurement offset (symbol-times)	0		int	[0, ∞)
MeasurementInterval	measurement interval (symbol-times)	60		int	[1, ∞)
SubchannelIndex	specify the subchannel to check the possible uplink subchannel preamble	16		int	[1, 31]
SymbolTimingAdjust	amount of time (expressed as a percent of the FFT time length) to back away from the end of the symbol time when deciding the part of the symbol that the FFT will be performed on	-3.125		real	[-100*GuardInterval, 0]
TrackAmplitude	pilot amplitude tracking: NO, YES	NO		enum	
TrackPhase	pilot phase tracking: NO, YES	YES		enum	
TrackTiming	pilot timing tracking: NO, YES	NO		enum	
EqualizerTraining	specify how the equalizer is initialized, or trained: CH Estimation Sequence Only, CH Estimation Sequence & Data	CH Estimation Sequence Only		enum	
SubcarrierSelect	carriers that will be analyzed: All, Single Carrier, Pilots Only	All		enum	
CarrierIndex	index of carrier to be analyzed when SubcarrierSelect = Single carrier	1		int	
Debug	display instant RCE and other information when set to YES: NO, YES	NO		enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	timed

Notes/Equations

1. This component performs an EVM measurement for an 802.16 OFDM signal.

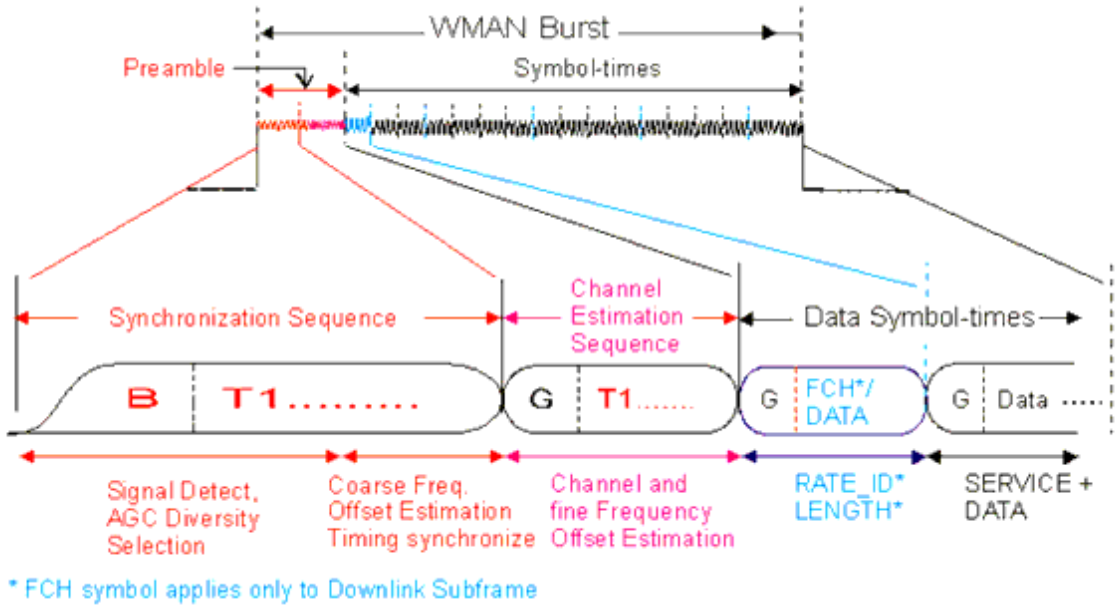
Note
The input signal must be a timed RF (complex envelope) signal or the component errors out.

This measurement provides results for RCErms_percent, RCE_dB, RCEpk_percent, PilotRCE_dB, CPErms_percent, FrequencyError_Hz, IQ_Offset_dB, and SyncCorrelation. To use these results in an AEL expression or in the *Goal* expression in an optimization setup, you must prefix them with the instance name of the component followed by a period (for example *W1.RCE_dB*).

Note
The 802.16 OFDM standard uses *RCE* (*relative constellation error*) instead of *EVM* (*error vector magnitude*); while the names are different, the calculated values are exactly the same. *CPE* means *common pilot error*.

The following notes provide a brief description of the algorithm (which is the same as that used in Agilent 89600 VSA) and parameters used in this component.
The WMAN burst structure is shown in the following figure. Many of the terms used in the following notes such as the preamble, channel estimation sequence, data symbol, guard-band (G, also called guard interval or cyclic prefix) are shown in this figure.

WMAN Burst Structure



2. Starting at the time instant specified by the Start parameter, a signal segment of SearchLength is acquired. This signal segment is searched in order for a complete burst to be detected. The burst search algorithm looks for both a *burst on* and a *burst off* transition. In order for the burst search algorithm to detect a burst, an idle part must exist between consecutive bursts and the bursts must be at least 15 dB above the noise floor.
   If the acquired signal segment does not contain a complete burst, the algorithm does not detect any burst and the analysis that follows most likely produces incorrect results. Therefore, SearchLength must be long enough to acquire at least one complete burst. Because the time instant specified by the Start parameter can be soon after the beginning of a burst, we recommend that SearchLength be set to a

value approximately equal to 2 × burstLength + 3 × idle, where burstLength is the duration of a burst in seconds and idle is the duration of the idle part in seconds. If it is known that Start is close to the beginning of a burst then SearchLength can be set to burstLength + 2 × idle. If the duration of the burst or the idle part is unknown, then a TimedSink component can be used to record the signal and the signal can be plotted in the data display. By observing the magnitude of the signal's envelope versus time you can determine the duration of the burst and the idle interval.

3. If AverageType = Off, only one burst is detected, demodulated, and analyzed.
   If AverageType = RMS (Video), after the first burst is analyzed the signal segment corresponding to it is discarded and new signal samples are collected from the input to fill in the signal buffer of SearchLength. When the buffer is full again a new burst search is performed; when a burst is detected it is demodulated and analyzed. These steps repeat until FramesToAverage bursts are processed.
   If a burst is misdetected for any reason, results from its analysis are discarded. The EVM results obtained from all successfully detected, demodulated, and analyzed bursts are averaged to give the final result.
4. DataSubcarrierModulation specifies the data subcarrier modulation format. This parameter does not affect the demodulation of the pilot subcarriers, which are always BPSK modulation.
   - When DataSubcarrierModulation = Auto Detect The algorithm uses the information detected within the 802.16 OFDM burst to automatically determine the data subcarrier modulation format.
   - When DataSubcarrierModulation = FCH The algorithm uses the information within the frame control header (FCH) to determine the data subcarrier modulation format. If the FCH symbol is not present, the data subcarrier modulation format is auto-detected instead.
   - When DataSubcarrierModulation = BPSK, QPSK, QAM 16, or QAM 64 The algorithm ignores the format determined from the OFDM burst and sets the demodulator to the modulation format specified.
5. GuardInterval specifies the guard interval (also called cyclic extension) length for each symbol-time, as a fraction of the FFT time period. The value must match the guard interval length actually used in the input signal in order for the demodulation to work properly.
6. NominalBandwidth specifies the nominal channel bandwidth defined in the 802.16 standard. The actual signal bandwidth is slightly less than this and the OFDM FFT sample rate is slightly more. NominalBandwidth, combined with FsBW_Ratio (and the knowledge of the FFT length) determines the OFDM subcarrier spacing.
7. FsBW_Ratio and OtherFsBW_Ratio specify the ratio (*n* in the standard) between OFDM FFT sample rate (*Fs* in the standard) and the nominal channel bandwidth (*BW* in the standard). The *ratio* is generally a number slightly larger than one.
   - When FsBW_Ratio = Auto, the *ratio* is detected automatically based on the bandwidth. The value of OtherFsBW_Ratio is ignored.
   - When FsBW_Ratio = _57/50, _8/7, _86/75, _316/275, or _144/125, the value of OtherFsBW_Ratio is ignored.
   - When FsBW_Ratio = Other, the value of OtherFsBW_Ratio is used. An arbitrary value between 0.5 and 2.0 is allowed.
8. PulseSearch controls how the algorithm searches for the preambles.
   - When PulseSearch = YES, the algorithm searches for the preambles at the beginning of the burst. This is the default and recommended setting.
   - When PulseSearch = NO, the algorithm tries to search for the preambles over

the entire signal. This is only needed when some 802.16 OFDM signals do not appear to be made up of RF bursts; these signals appear to be continually on, with preambles embedded within the signal.

9. ResultLengthType and ResultLength control how much data is demodulated.
   - When ResultLengthType = Auto Select, the measurement result length is determined automatically. In this case, the ResultLength defines a Max Result Length for the burst in symbol-times; that is, the measurement compares the number of symbol-times detected within the burst to the Max Result Length and uses the smaller value as the measurement result length.
   - When ResultLengthType = Try FCH, the algorithm first tries to use the Frame Control Header to determine the measurement result length. If the FCH symbol is not present, the measurement result length is auto-detected based on the RF pulse length.
   - When ResultLengthType = Manual Override, the measurement result length is set to ResultLength and the number of symbol-times detected within the burst is ignored.

   The following table summarizes how Auto Select, Try FCH, and Manual Override modes determine the measurement result length. The table lists the measurement result lengths actually used for three different values of ResultLength (30, 26 and 20 symbol-times). It is assumed that the input burst is 26 symbol-times long.

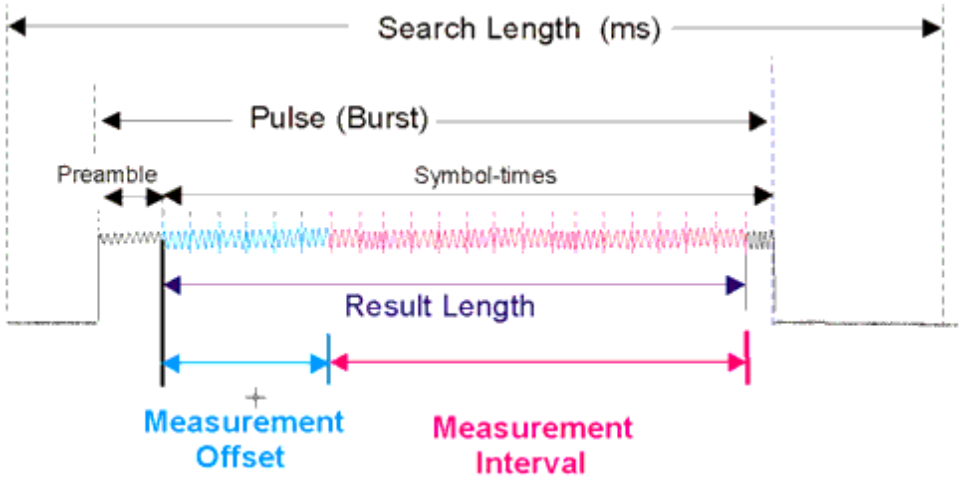
ResultLength Parameter Settings

ResultLengthType	ResultLength	Measurement Result Length Actually Used
Auto Select / Try FCH	20	20
Auto Select / Try FCH	26	26
Auto Select / Try FCH	30	26
Manual Override	20	20
Manual Override	26	26
Manual Override	30	30

   Note that when ResultLengthType = Manual Override and ResultLength = 30 (larger than the actual burst size) the algorithm demodulates the full 30 symbol-times even though this is 4 symbol-times beyond the burst width.

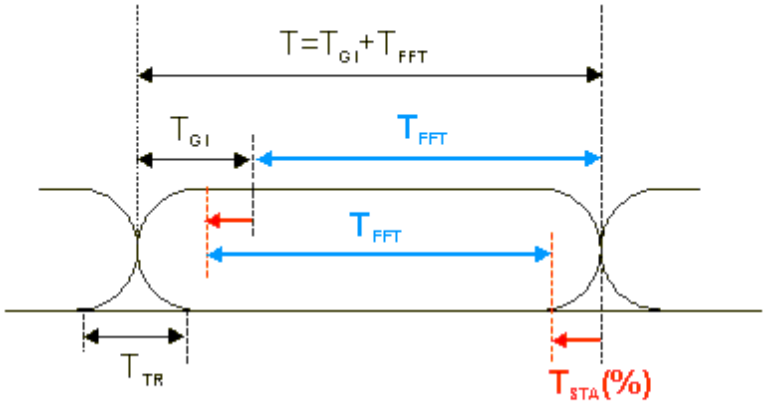
10. With the MeasurementInterval and MeasurementOffset parameters you can isolate a specific segment of the ResultLength for analysis. Only the segment specified by these two parameters is analyzed in order to get the EVM results. The following figure shows the interrelationship between the SearchLength, ResultLength, MeasurementInterval, and MeasurementOffset.

Interrelationship between SearchLength, ResultLength, MeasurementInterval, and MeasurementOffset.



11. SubchannelIndex controls analysis of 802.16-2004 uplink subchannels. When demodulating an 802.16 OFDM signal, the algorithm checks for several different kinds of preambles, including Long preamble (found on Downlink subframes), Short preamble (found on most Uplink bursts), and a Subchannel preamble (found on Uplink subchannel bursts). The type of preamble found determines the type of demodulation that is done. However, there are 31 different possible uplink subchannels specified in the 802.16-2004 standard, so the algorithm checks for only one of the possible subchannels. This parameter controls which of the subchannels the analyzer checks for. The default value is 16, which is a subchannel that uses all 200 subcarriers.
12. SymbolTimingAdjust adjusts the symbol timing used for demodulation. Normally, when demodulating an OFDM symbol, the guard interval is skipped and an FFT is performed on the last portion of the symbol-time. However, this means that the FFT includes the transition region between this symbol and the following symbol. To avoid this, it is generally beneficial to back away from the end of the symbol-time and use part of the guard interval.
SymbolTimingAdjust controls how far the FFT part of the symbol is adjusted away from the end of the symbol-time. The value is in terms of percent of the used (FFT) part of the symbol-time. Note that the SymbolTimingAdjust parameter value is negative, because the FFT start time is moved back by this parameter. The following figure demonstrates this concept. When setting SymbolTimingAdjust, do not back away from the end of the symbol-time too much because this can make the FFT include corrupt data from the transition region at the beginning of the symbol-time.

SymbolTimingAdjust Definition



T = Symbol Time
T_{GI} = Guard Interval
T_{FFT} = FFT/IFFT Time Period
T_{TR} = Symbol Transition Time
T_{STA} = Symbol Timing Adjust (%)

13. TrackAmplitude, TrackPhase, and TrackTiming specify how pilot tracking is used to correct for imperfections in the equalizer response (calculated from the burst preamble) and for imperfections that change over the length of the burst.
    - When TrackAmplitude = YES, the algorithm applies pilot subcarrier amplitude error correction to the pilot and data subcarriers.
    - When TrackPhase = YES, the algorithm applies pilot subcarrier phase error correction to the pilot and data subcarriers.
    - When TrackTiming = YES, the algorithm applies pilot subcarrier timing error correction to the pilot and data subcarriers.
14. EqualizerTraining specifies how the equalizer is trained. When demodulating the 802.16 OFDM signal, the algorithm uses an equalizer to correct for linear impairments in the signal path, such as multi-path.
    - When EqualizerTraining = CH Estimation Sequence Only The equalizer is trained by looking at the channel estimation sequence in the preamble of the OFDM burst. After this initialization, the equalizer coefficients are held constant while demodulating the rest of the burst.
      Advantages of this method: it models what a typical OFDM receiver would do, so the measured RCE (EVM) more accurately reflects the signal quality seen by a typical OFDM receiver; and it complies with the description in the *Transmit constellation error and test method* section of the 802.16 standard.
      Disadvantage of this method: the measured RCE (EVM) value can be higher for signals whose impairments change during the burst than it would be if the equalizer were trained over the entire burst.
    - When EqualizerTraining = CH Estimation Sequence & Data The equalizer is trained by analyzing the entire OFDM burst, including the channel estimation sequence (in the preamble) and the data symbols. This type of training generally gives a more accurate estimate of the true response of the transmission channel.
      Advantages of this method: the equalizer coefficients typically reflect the linear channel impairment with greater accuracy, as the dataset used to train the equalizer is larger and is less affected by turn-on transient effects in the burst; and the RCE (EVM) is typically lower because the equalizer is less impacted by

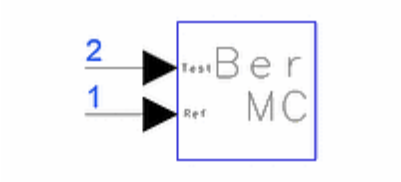
noise and some other forms of distortion.
Disadvantage of this method: it is less likely to accurately reflect the performance of a typical OFDM receiver. Because this type of equalizer calculation is more complicated and therefore more expensive to implement, it is less likely to be used in practical receivers.

15. SubcarrierSelect and CarrierIndex specify what subcarrier data is analyzed.
    - When SubcarrierSelect = All, all subcarriers (−100 through 100, skipping 0) is analyzed.
    - When SubcarrierSelect = Single Carrier, the subcarrier number specified by CarrierIndex is analyzed.
    - When SubcarrierSelect = Pilots Only, all pilot subcarriers is analyzed.

References

1. IEEE Std 802.16-2004, "Part 16: Air Interface for Fixed Broadband Wireless Access Systems" 2004.
2. Agilent 89600 VSA software, Option B7S 802.16 OFDM. http://www.agilent.com/find/89600 .

berMC



Description: Error Probability measurement using Monte Carlo Method, 2 inputs
Library: Sinks
Class: TSDF_berMC
Derived From: baseTimeSink

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
ControlSimulation	if set to YES, 'Stop' time determines how long the simulation will run: NO, YES	YES		enum	
SymbolTime	symbol time	1.0	sec	real	[TStep, ∞)†
EstRelVariance	estimation relative variance (if set to 0.0 simulation will run until Stop is reached)	0.0		real	[0, 1\]]
NumThreshold	number of threshold	1		int	[1, ∞)
ThresholdSetting	type of threshold settings: automatically, manually	automatically		enum	
Threshold	threshold values when user selects manually setting	0		real array	
DelayBound	upper bound of delay for Synchronizing inputs; if DelayBound = -1, the Synchronizer is turned off	-1.0	sec	real	{-1} or (0, ∞)††
berOutput	Type of ber output: ber vs time, ber vs time every 10 symbols, ber vs time every 100 symbols, ber vs time every 1000 symbols, ber only	ber only		enum	
StatusUpdatePeriod	Status update period in number of bits	1000		int	[1, ∞)

† TStep is the simulation time step for the component input signals.
†† If DelayBound is set to -1, berMC will assume test and reference signals are already synchronized.

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed
2	input2	test signal input	timed

Notes/Equations

1. Basic Principle
   berMC uses the Monte Carlo method for probability of error measurement of linear as well as nonlinear communication systems.

The measurement is based on comparing test data to reference data waveforms, symbol by symbol.

To measure symbol error probability:

- Synchronize output test data (test data) with reference data.
- Sample reference and test data. Each sampling time must be at the center of each symbol. The sampling time interval is one symbol time.
- Compare the test data sample to the reference data sample.
- Detect an error event by using thresholds. An error occurs if the value of the test sample is not the same as reference sample in the threshold detection, otherwise, there is no error.
- Continue counting error events. Assume the number of error events is n, and the total number of test (or reference) data samples is N. Error probability = n/N.

2. Data Normalization
   To simplify detection, the parameter ThresholdSetting can be set to *automatically* and all detection thresholds are normalized from -1 to 1. The threshold range results in symbol ranges that are equally spaced between −1 and +1. This means test and reference data must be normalized from -1 to 1.
   If ThresholdSetting = *manually*, test data must be adjusted in the same range of reference data.
3. Synchronizing Test and Reference Data
   Test and reference data can be synchronized automatically or by the designer.
   Automatic Synchronization

   Set DelayBound > exact delay between test and reference data. Typically, set DelayBound = 5× to 10× exact delay. This approach does not require knowing the exact delay between test and reference data. Auto-synchronization estimates the delay based on calculating the cross-correlation function between the two waveforms.

   User Synchronization

   - Set DelayBound = −1.
   - Find the exact delay between test and reference data.
   - Place a Delay component between the reference data source and the reference input pin of the berMC component.
     Continue synchronization with these parameters.
   - Start is used for specifying a sampling start time for the reference signal, and also for positioning test and reference samples. As mentioned, each sampling time must be at the center of each symbol. For example, for the reference signal from ADS data source if DelayBound = −1, Start must be set to Delay + SymbolTime/2, where the exact delay between test and reference data is Delay, and symbol time is SymbolTime. Otherwise, Start must be set to SymbolTime/2.
   - Stop is used for setting N total test samples (or reference samples). N = Stop/SymbolTime. A larger N results in a more accurate error probability.
   - NumThreshold is determined by signal levels; assume the signal level is M, NumThreshold = M − 1.
   - ThresholdSetting options:
     automatically setting thresholds for uniform threshold interval.
     manually setting thresholds for uniform or non-uniform interval.

- Threshold: Threshold values when you select the *manually* setting.
- If berOutput = ber only, berMC provides the BER estimation value;
  if berOutput = ber vs time, berMC provides the BER estimation value and the BER Vs Time plot.

4. Delay between reference signal and test signal. If set DelayBound > 0, this component provides an estimation of the delay. The delay value can be found in the data display server.
5. The EstRelVariance parameter can be used to control the quality of the estimate that berMC generates (for details, refer to *PE Measurement Concepts* under the *Sinks, PE Estimator Usage* topic). If set to 0, then it is basically ignored and berMC bases its estimate on data collected from Start to Stop.
   When EstRelVariance is set to a positive value (≤ 1), then berMC bases its estimate on data collected from Start until the estimate's relative variance drops below EstRelVariance or until Stop is reached (in this case Stop provides an upper bound on how long the simulation should run). In this mode of operation, and since the exact time the simulation finishes is unknown, berMC reports (on the status window) the relative variance of its current estimate every time 1000 symbols are processed.
   Note that the equation for the estimation relative variance given (in *PE Measurement Concepts* under the *Sinks, PE Estimator Usage* topic) assumes that the errors happen randomly (as in the case of an AWGN channel) and not in bursts (as in the case of a fading channel).

6. Symbol Error Probability and Bit Error Probability
   berMC can measure the symbol error probability or symbol error rate (SER) for a single channel. For a system with only one channel, such as PAM or BPSK, the bit error probability or bit error rate (BER) can be calculated by

   $$BER = SER / L$$

   where L is the number of bits per symbol (for example, for a BPSK system L = 1, for an 8-level PAM system L = 3). Note that the above formula assumes that Gray coding was used to map the bits to symbols.
   For systems with I and Q channels, such as QAM, QPSK, π/4-DQPSK, you must place two berMC components to measure the SER for each channel, SERi and SERq. Then the SER for the whole system can be calculated by SER = SERi + SERq − SERi × SERq
   This formula assumes that both berMC components used to measure SERi and SERq process the same number of symbols in the simulation (see note 7 on how to make sure this is true).
   To calculate the BER for a system with two channels, first calculate the BER for each channel BERi and BERq using the equations

   $$BERi = SERi / L \text{ and } BERq = SERq / L$$

   where L is now the number of bits per symbol in each channel (for example, for QPSK L = 1, for 16-QAM L = 2, for 64-QAM L = 3). Again Gray coding is assumed in the mapping of bits to symbols. Note that this does not work for systems with non-rectangular constellations such as 8-PSK, 32-QAM, 128-QAM. berMC cannot be used to measure SER or BER for these systems. After BERi and BERq have been calculated, the BER for the whole system is given by

BER = (BERi + BERq) / 2

7. As mentioned in note 6, for systems with two channels it is important that both berMC components used to measure the SER for each channel process the same number of symbols. If this is not true, the equations given in note 6 to determine the SER and BER of the whole system from the SERi and SERq gives biased results.
   The two approaches to make both berMC sinks process the same number of symbols during the simulation are:
   - if EstRelVariance is not used (it is set to 0) set the Start parameters of both sinks to the same value and the Stop parameters of both sinks to the same value. The ControlSimulation parameter must be set to YES for both sinks.
   - if EstRelVariance is used (it is set to a positive value ≤ 1), then set the Start parameters for both sinks to the same value and make one sink to control the simulation and the other sink not to control the simulation.
     For example, if you want a relative variance of 0.01, maximum symbols processed 1.0e6, and start processing data after 8.5 msec, then, assuming that symbol time is 1 msec:
     Parameters of the first sink must be set as follows:

     Start = 8.5 msec

     Stop = 8.5 msec + 1.0e6 × 1 msec

     ControlSimulation = YES

     SymbolTime = 1 msec

     EstRelVariance = 0.01

     Parameters of the second sink must be set as follows:

     Start = 8.5 msec

     Stop = 8.5 msec + 1.0e6 × 1 msec

     (not used when ControlSimulation= NO)

     ControlSimulation = NO

     SymbolTime = 1 msec

     EstRelVariance = 0.01

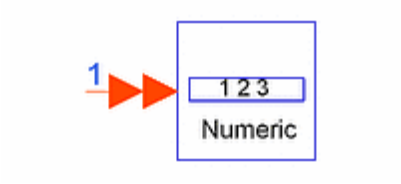
     (not used when ControlSimulation=NO)

     Depending on the order the scheduler decides to fire the two sinks, the sink that controls the simulation can end up processing one symbol more than the other one. This is insignificant and does not bias the results obtained with the equations given in note 6.
8. The StatusUpdatePeriod parameter can be used to control how often estimation

relative variance status messages are reported to the simulation status window.

9. Lengthy BER simulation times can be significantly shortened by using the *Parallel BER* option in the *Simulate* > *Simulation Setup* window > *Distributed* tab. (To use this option, your computer must have LSF client installed and it must be connected to an LSF cluster.) In the Simulation Setup dialog, set *Simulation Mode* to *Distributed* and on the *Distributed* tab, select the option *Parallel BER*.
   When the *Parallel BER* option is selected, enter a value for *Number of Partitions*. This value defines the number of parallel simulations that LSF spawns. It is also the expected simulation time speedup factor assuming that there are at least this many computers in the LSF cluster and that the computers are equivalent in terms of processing power.
   The BER results from a simulation that run using the *Parallel BER* option are NOT identical to the results if the same simulation was run on a single computer. However, the two results are statistically equivalent.
   - For a demonstration of this equivalence and some simulation performance speedup results refer to the design in *File* > *Open* > *Example* > Com_Sys/ParallelBER_wrk_ > *ParallelBER_Example1* and the corresponding data display file.
   - For an example of a swept Eb/No BER simulation using the *Parallel BER* option refer to the design in *File* > *Open* > *Example* > Com_Sys/ParallelBER_wrk_ > *ParallelBER_Example2* .
     Note that, although there is no upper limit to the value you can enter in the *Number of Partitions* field, large values for this parameter can lead to wrong results. Consider the following scenario.
     A design is set up for a BER measurement that simulates 1e6 bits. Let's assume that the time period corresponding to 1e6 bits is 1 sec. If this simulation is split over 100 computers using the *Parallel BER* option then each computer simulates a time period of approximately 10 msec.
     If the goal of the simulation was to observe the effect on BER of fading with fade rate 100 Hz, then the results from the simulation using the *Parallel BER* option most likely are wrong. The reason is that in a fading environment with a fade rate of 100 Hz, a fade in the signal power occurs every 10 msec on the average. In the single simulation setup, the computer simulates a time period of 1 sec over which approximately 100 fades occur. However, in the Parallel BER simulation a fade can not occur at all since each computer simulates a time period of approximately 10 msec. A value of 10 for *Number of Partitions* is a more appropriate choice for this scenario.
     The conclusion is that when selecting a value for *Number of Partitions* you must ensure that the phenomena you are trying to observe occurs in the reduced time period simulated by each of the parallel simulations LSF spawns.
10. For general information regarding sinks, refer to *About Sinks* (sinks).

NumericSink



Description: Numeric Data Sink
Library: Sinks
Class: SDFNumericSink
Derived From: NumericSinkGated

Parameters

Name	Description	Default	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None	enum	
Start	Sample number to start collecting data. DefaultNumericStart will inherit from the DF controller.	DefaultNumericStart	int	[0, ∞)
Stop	Sample number to stop collecting data. DefaultNumericStop will inherit from the DF controller.	DefaultNumericStop	int	[Start, ∞)
ControlSimulation	If set to YES, 'Stop' sample number determines how long the simulation will run: NO, YES	YES	enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	multiple anytype

Notes/Equations

1. NumericSink collects data from the output of the connected component and saves it to the simulation dataset.
   When the connected component is a numeric component, the collected data is in the form of integer, real, or complex data values versus an integer index value.
   When the connected component is a timed component, the collected data is in the form of complex data values (for baseband timed signals the imaginary part is set to 0) versus an integer index value. The collected data does not have information on the characterization frequency of the timed signal and it does not preserve the timestamps.
2. The NumericSink component can record data of all the following data types: integer scalar, floating-point (real) scalar, fix scalar, complex scalar, timed scalar, integer matrices, floating-point (real) matrices, fix matrices, complex matrices, integer busses, floating-point (real) busses, fix busses, and complex busses. All types of fix data are converted to the corresponding floating-point (real) type before being

recorded.

Busses of timed data, busses of different data types, and busses of matrices (integer, floating-point (real), fix, or complex) are not supported. In addition, swept matrix dimensions and/or swept bus sizes are not supported.

3. The name of the variable (holding the simulation data) that is created in the dataset by each NumericSink is the same as the sink's instance name. For bus or matrix data the variable with the sink's instance name represents the whole bus or matrix structure. The individual bus or matrix elements are also available for plotting or use in AEL expressions and meas equations. For example, a matrix of size 2×3 connected to a NumericSink named MyMatrix results in the following variables being listed in the set of variables available for plotting in the Data Display window: MyMatrix, MyMatrix(1,1), MyMatrix(1,2), MyMatrix(1,3), MyMatrix(2,1), MyMatrix(2,2), MyMatrix(2,3).

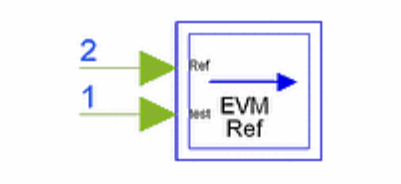
   When iterated NumericSink instances are used (for example, N<1:3>) the names of the instantiated sinks contains the special characters < and > (for example, N<1>, N<2>, and N<3>). These characters have special meaning and these must be quoted when used in AEL expressions, meas equations, or as the goal expression in an optimization simulation. For AEL expressions and meas equations the correct usage is var("N<1>"); for optimization goal expressions the correct usage is var(""N<1>"") (the Expression parameter of the Goal component is a string; two double quotes are needed to represent one double quote inside a string).
4. When a bus is connected to an iterated NumericSink instance then the bus size S must be an integer multiple of the number of the NumericSink instances N, otherwise an error is reported.
   - If S = N, then each NumericSink instance is connected to a single bus element so the data saved by each sink is scalar.
   - If S = k × N, where k is an integer greater than 1, then the first k bus elements are connected to the first NumericSink instance, the second k bus elements are connected to the second NumericSink instance, and so on.

     In this case, each NumericSink instance is connected to a bus of size k so the data saved by each sink has the bus format. A known problem in this case is that the individual bus elements are not listed in the set of variables available for plotting in the Data Display window. For example, assuming that k = 2 and the iterated NumericSink instance is A<1:3>, you would expect to see A<1>, A<1>(1), A<1>(2), A<2>, A<2>(1), A<2>(2), A<3>, A<3>(1), A<3>(2) listed in the set of variables available for plotting in the Data Display window but in fact only A<1>, A<2>, and A<3> are listed. However, you can still plot the individual bus elements by first selecting to plot the whole bus (for example, A<2>) then manually editing the expression that is being plotted, for example, var("A<2>")(2) to plot the second element of the bus A<2>.

5. The amount of data collected by a NumericSink is controlled by the Start, Stop, and ControlSimulation parameters. Data collection always begins at the input sample with an index value equal to the value of the Start parameter (index values start at 0 and increment by 1 for every sample).
   - If ControlSimulation is set to YES, then data collection ends at the input sample with an index value equal to the value of the Stop parameter.
   - If ControlSimulation is set to NO, then data collection ends when the simulation ends; in this case, there must be at least one other source or sink component that controls how long the simulation runs.

6. For general information regarding sinks, refer to *About Sinks* (sinks).

EVM_WithRef



Description: EVM measurement with reference signal input
Library: Sinks
Class: SDFEVM_WithRef

Parameters

Name	Description	Default	Sym	Unit	Type	Range
StartSym	start symbol	20			int	[0, ∞)
SymBurstLen	number of symbols within burst to be measured	100	N		int	[1, 10000]
SampPerSym	number of samples per symbol	16			int	[1, ∞)
SymDelayBound	upper bound of delay detection, in symbols, -1 for no detection	3			int	[-1, 255]
NumBursts	number of bursts to be measured	5			int	[1, ∞)
MeasType	type of measurement: EVM_rms, EVM_peak, EVM_95th_percentile	EVM_rms			enum	
SymbolRate	symbol rate	1e3		Hz	real	(0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	testDataInput	test signal for EVM measurement	complex
2	RefDataInput	reference signal for EVM measurement	complex

Notes/Equations

1. EVM_WithRef performs the error vector magnitude measurement on the signal applied at its testDataInput pin using the signal applied at the RefDataInput pin as a reference. The fact that a signal is used as a reference makes this component modulation independent, that is EVM_WithRef can be used to measure EVM for any modulation format.
   The signal at the RefDataInput input can be a non ideal signal. EVM is evaluated at all downsampling phases in a symbol period (if the input signals have more that one sample per symbol) in order to find the minimum EVM. This may result in the input signals not being sampled at the optimal instant inside the symbol period.
   The intended use of EVM_WithRef is for signals whose reference is not one of the standard formats handled by the EVM component. Use of EVM_WithRef for one of the signal formats handled by the EVM component might not give the same EVM value as the one obtained with the EVM component. This is expected. The EVM_WithRef can

provide EVM measurement capability for user custom signals. That custom EVM value might not occur at the optimal sampling instant for the users custom signal and its EVM value may differ from that obtain at an optimal sampling instant.
For EVM measurements for signal formats supported by the EVM component it is recommended that you use the EVM component instead of EVM_WithRef. The EVM component does not take a reference signal; it generates the ideal reference signal internally. However, the EVM component cannot handle test signals with high frequency error.

2. EVM measurements are used to evaluate the modulation accuracy of modulators. For example, in the IS-54 TDMA digital cellular, these are used to set the minimum specifications for the accuracy of π/4-DQPSK modulators.
Let Z(k) denote the actual complex vectors (I and Q) produced by observing the real transmitter through an ideal receiver filter at instants k, one symbol period apart. Let S(k) denote the ideal reference symbol. Then, Z(k) is modeled as:

$$Z(k) = \{C0 + C1 \times [S(k) + E(k)]\} \times W^k, 0 \le k \le N-1$$

where

W = e $^{Dr + jDa}$ accounts for both a frequency offset (*Da* radians per symbol phase rotation) and an amplitude change rate (*Dr* nepers per symbol)

C0 is a complex constant origin offset (in Volts)

C1 is a unitless complex constant representing the arbitrary phase and output power of the transmitter

E(k) is the residual vector error on sample S(k) (in Volts)

The sum square error vector is

$$\sum_{k=0}^{N-1} |E(k)|^2 = \sum_{k=0}^{N-1} \left| \frac{[Z(k)W^{-k} - C_0]}{C_1} - S(k) \right|^2$$

where C $_0$, C $_1$, W are chosen such as to minimize the above expression.

EVM (rms) is defined to be the rms value of | E(k) | normalized by the rms value of | S(k) |. Therefore,

$$EVM(rms) = \frac{\sqrt{\frac{1}{N} \times \sum_{k=0}^{N-1} |E(k)|^2}}{\sqrt{\frac{1}{N} \times \sum_{k=0}^{N-1} |S(k)|^2}} = \sqrt{\frac{\sum_{k=0}^{N-1} |E(k)|^2}{\sum_{k=0}^{N-1} |S(k)|^2}}$$

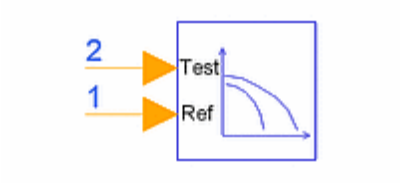
The symbol EVM at symbol k is defined as

$$EVM(k) = \frac{|E(k)|}{\sqrt{\frac{1}{N} \times \sum_{k=0}^{N-1} |S(k)|^2}}$$

which is the vector error magnitude at symbol k normalized by the rms value of | S(k) |.

3. The MeasType parameter determines the output of the component.
   - If MeasType = EVM_rms, the output is the EVM(rms) value as defined in the equations of note 2. This value is not in percent.To get EVM RMS in percent, multiply by 100.
   - If MeasType = EVM_peak, the output is max{ EVM(k) }, where the maximum is evaluated over k = 0, 1, ... , N-1. This value is not in percent.To get EVM Peak in percent, multiply by 100.
   - If MeasType = EVM_95th_percentile, the output is the 95th percentile of the EVM(k) values. The 95th percentile of the EVM(k) values is the value that is greater than 95% of the EVM(k) values and less than 5% of the EVM(k) values. This value is not in percent.To get EVM RMS in percent, multiply by 100.
     In addition the following quantities are displayed on the status window
   - $20 \times \log 10\left(\frac{|C0|}{RMS(|S(k)|)}\right)$, origin offset expressed in dB with respect to the rms value of | S(k) |.
   - $(Da \times SymbolRate)/(2 \times \pi)$, frequency error in Hz.
   - $-20 \times \log 10(e^{Dr})$, droop expressed in dB with respect to 1.
4. The StartSym parameter can be used to set the symbol at which data collection for the EVM measurement starts. The primary use of this parameter is to exclude antisocial symbols (during filter transients) so that these do not affect the measurement results.
5. The SymBurstLen parameter defines the burst size on which the measurement is performed.
6. The SymDelayBound parameter is used for synchronizing the test and reference signals. Since the two signals go through different paths the test signal most likely is delayed with respect to the reference signal. If SymDelayBound is set to a value greater than 0, then the component cross-correlates the two signals trying to detect the delay between them. However, only delays between 0 and SymDelayBound are considered. Note that SymDelayBound is in number of symbols, that is if an upper bound of the delay is 3 symbols, SymDelayBound should be set to 3 and not to 3 × SampPerSym. If SymDelayBound is set to −1 then no synchronization is performed.
7. Most standards specify that the EVM measurement should be performed over multiple bursts and the results should be averaged. This functionality is provided by the NumBursts parameter. If NumBursts is set to a value greater than 1 then the EVM measurement is performed over NumBursts consecutive bursts and the results of the individual measurements are averaged. Averaging is applied to both the values sent to the dataset and the ones displayed on the status window.
8. The SymbolRate parameter is used only to calculate the frequency error based on the formula given in note 3. It does not affect the EVM measurement in any other way.
9. For general information regarding sinks, refer to *About Sinks* (sinks).

BER_FER



Description: Bit Error Rate and Frame Error Rate estimation
Library: Sinks
Class: SDFBER_FER

Parameters

Name	Description	Default	Type	Range
Plot	Plot data when set to 'Rectangular' and Simulation Setup set to 'Open Data Display when simulation completes': None, Rectangular	None	enum	
Start	Data collection start index	DefaultNumericStart	int	[0, ∞)
Stop	Data collection stop index when EstRelVariance is not met	DefaultNumericStop	int	(Start, ∞)
ControlSimulation	Let sink control how long the simulation will run? NO, YES	YES	enum	
BitsPerFrame	Bits per frame	100	int	[1, ∞)
EstRelVariance	BER estimation relative variance	0.01	real	[0, 1)
OutputBER	BER output: BER vs index, BER vs index every 10 bits, BER vs index every 100 bits, BER vs index every 1000 bits, BER vs index every BitsPerFrame bits, Final BER	Final BER	enum	
OutputFER	FER output: FER vs frame, FER vs frame every 10 frames, Final FER, No FER	Final FER	enum	
StatusUpdatePeriod	Status update period in number of bits	1000	int	[1, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	ref	reference bit stream	int
2	test	test bit stream	int

Notes/Equations

1. BER_FER can be used to measure the BER (bit error rate) and FER (frame error rate) of a system. In some systems, FER is referred to as PER (packet error rate) or BLER (block error rate). The input signals to the reference (ref) and test (test) inputs must be bit streams. The bit streams must be synchronized, otherwise the BER/FER estimates are wrong.
2. The Start parameter defines when data processing starts. The end of data processing depends on the settings of the parameters ControlSimulation, Stop, and

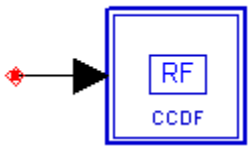
EstRelVariance:

- If ControlSimulation is NO, then Stop and EstRelVariance are ignored. Data processing ends when the simulation ends. In this case, the end of the simulation is determined by other sink or source components that control the simulation.
- If ControlSimulation is YES and EstRelVariance is 0.0, then data processing ends when Stop is reached.
- If ControlSimulation is YES and EstRelVariance is greater than 0.0, then data processing ends when EstRelvariance is met or when Stop is reached. In this case, Stop acts as an upper bound on how long the simulation runs just in case the simulation takes too long for EstRelVariance to be met. In this mode of operation, messages are printed in the simulation status window showing the value of estimation relative variance as the simulation progresses. The EstRelVariance parameter can be used to control the quality of the BER estimate BER_FER generates. The lower the value of EstRelVariance the more accurate the estimate is.
  For more details, refer to *PE Measurement Concepts* (sinks). Note that the equation for the estimation relative variance described above assumes that the errors happen randomly (as in the case of an AWGN channel) and not in bursts (as in the case of a fading channel).

3. The BitsPerFrame parameter sets the number of bits in each frame. A frame is considered to be in error if at least one of the bits in the frame is detected incorrectly. If the bit errors are independent identically distributed events then BER and FER are related through the equation $FER = 1 - (1 - BER)^{BitsPerFrame}$.
   To estimate BER/FER over an exact number of frames set ControlSimulation to YES, EstRelVariance to 0.0, and Stop to Start + N × BitsPerFrame − 1, where Start is the value of the Start parameter, BitsPerFrame is the value of the BitsPerFrame parameter and N is the number of frames to be simulated.
4. The OutputBER parameter determines how often BER values are written in the dataset.
   - Final BER - writes the final BER value.
   - BER vs index - writes BER values as a function of index. You can see how BER changes throughout the simulation.
   - BER vs index every 10 bits, BER vs index every 100 bits, BER vs index every 1000 bits, and BER vs index every BitsPerFrame bits - behave similar to BER vs index but BER values are written every 10, 100, 1000, and BitsPerFrame bits, respectively. You can see how BER changes throughout the simulation while keeping the dataset at a reasonable size.
5. The OutputFER parameter determines how often FER values are written in the dataset.
   - No FER - does not write any FER values.
   - Final FER - writes the final FER value.
   - FER vs frame - writes FER values as a function of frame. You can see how FER changes throughout the simulation.
   - FER vs frame every 10 frames - behaves similar to FER vs frame but FER values are written every 10 frames. You can see how FER changes throughout the simulation while keeping the dataset at a reasonable size.
6. The StatusUpdatePeriod parameter can be used to control how often estimation relative variance status messages are reported to the simulation status window.
7. Lengthy BER simulation times can be shortened significantly by using the *Parallel BER*

option in the *Simulate* > *Simulation Setup* window > *Distributed* tab. (To use this option, your computer must have LSF client installed and it must be connected to an LSF cluster.) In the Simulation Setup dialog, set *Simulation Mode* to *Distributed* and on the *Distributed* tab, select the option *Parallel BER*. For more information, see the description of the *Parallel BER option* (sinks) in the *berMC* component documentation. An example that demonstrates the *Parallel BER* feature using the BER_FER component is provided; to access this example, from the ADS Main window choose **File > Open > Example > WLAN > WLAN_80211a_PER_wrk > WLAN_80211a_36Mbps_AWGN_System**.

RF_CCDF



Description: Complementary, cumulative density function (CCDF) for RF Signals
Library: Sinks
Class: TSDF_RF_CCDF
Derived From: baseAnalysis

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
NumBins	Number of points in the CCDF curve	100		int	[3, 65535]
OutputPeakMean	Output signal peak and mean values: NO, YES	NO		enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

1. RF_CCDF provides a complementary cumulative distribution function (CCDF) measurement for an RF time-domain signal. As its name suggests, CCDF is the complement of cumulative distribution function (CDF), and is defined as follows:

   CCDF = 1 - CDF

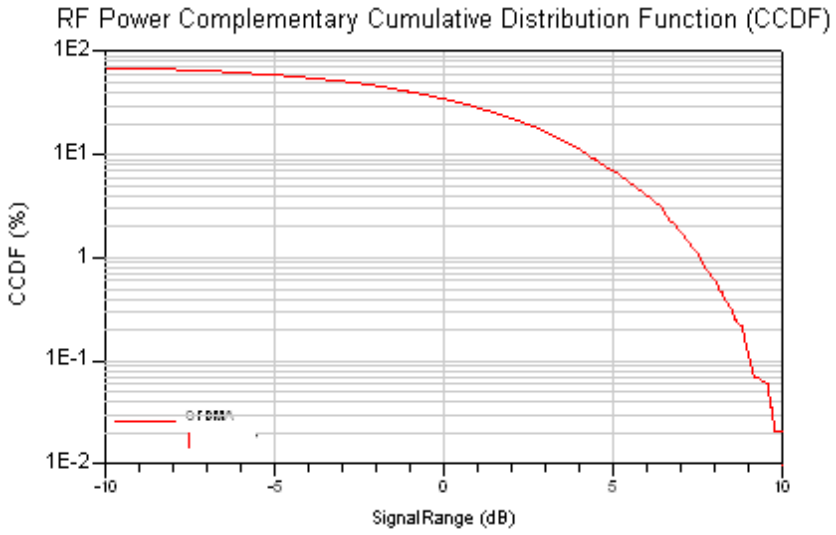
   To calculate CCDF, the following steps need to be taken:

   - Calculate the RMS value for all measured samples; this becomes the 0 dB point

on the x-axis.
- Normalize all samples to the RMS value in units of dB.
- Split the x-axis in equal width NumBin bins starting from minimum measured power to maximum measured power.
- Determine which x-axis bin each sample belongs to.
- Calculate the total number of samples that are greater than or equal to each x-axis bin and output it as a percent of the number of samples measured.

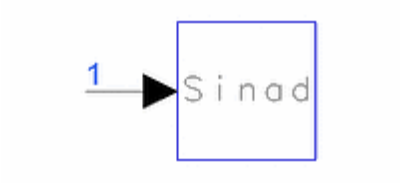
2. Besides CCDF measurement, this component can provide PeakPower (the peak power for the input signal) and MeanPower (the mean or average power of the input signal power). Note that the units of PeakPower and MeanPower are dBm. To compute and output MeanPower and Peak Power, set the parameter OutputPeakMean to YES.
3. The CCDF measurement is a very common measurement performed on 2G, 3G and 4G wireless signals. The CCDF curve shows the probability that the instantaneous signal power will be higher than the average signal power by a certain amount of dB. The independent axis of the CCDF curve shows power levels in dB with respect to the signal average power level (0 dB corresponds to the signal average power level). The dependent axis of the CCDF curve shows the probability that the instantaneous signal power exceeds the corresponding power level on the independent axis. The following figure shows the CCDF curve for a WiMax 802.16e Downlink signal. In the figure, you can see that the instantaneous signal power exceeds the average signal power (0 dB) for 35% of the time. Also, you can see that the instantaneous signal power exceeds the average signal power by 5 dB for only 7% of the time.

RF CCDF for a WiMax 802.16e Downlink Signal



4. RF_CCDF is a single rate component collecting input timed data point by point. The amount of data collected is controlled by parameters Start and Stop. Data collection always begins at the time instant specified by Start and ends at the time instant specified by Stop.
5. In most wireless communication systems with framed data, the CCDF measurement is defined in the active signal power region. In this case, to achieve the correct CCDF measurement, you should de-frame the data, then use the RF_CCDF Measurement to measure the signal CCDF in the active signal power region.

Sinad



Description: SINAD measurement
Library: Sinks
Class: TSDF_Sinad
Derived From: baseAnalysis

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
SignalFrequency	signal Frequency of interest.	1.0e6	Hz	real	(0, 1/(2 × TStep))†
BandRejectSpan	band reject filter bandwidth.	0.0	Hz	real	[0, 1/(2 × TStep))††
HanningWindow	windowing of signal with a Hanning window.: OFF, ON	ON		enum	

† TStep is the simulation time step for the component input signal.
†† Refer to Note 3 for details regarding use of this parameter.

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

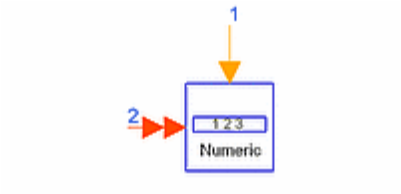
Notes/Equations

1. The SINAD measurement calculates the ratio *(S+N+D)/(N+D)* in dB, where *S* denotes signal power (that is, the signal power centered at SignalFrequency), *N* denotes the noise power and *D* denotes distortion power (that is, the residual power not accounted for in *S* or *N*).
   The *(S+N+D)* power term is estimated from the signal over the time interval from Start to Stop.
   The *(N+D)* power term is estimated by first taking an FFT (fast Fourier transform) of the signal over the time interval from Start to Stop, applying an ideal band-reject filter (with a reject span of BandRejectSpan) centered at SignalFrequency on the FFT'd signal (in the frequency domain) and finally summing up the power of the filtered signal.
   In the SINAD measurement, it is assumed that the signal is a real signal (the Q-channel information is not used).
2. SignalFrequency is the center frequency of the desired signal. For example, in a 1-tone SINAD measurement of an FM demodulator, SignalFrequency should be set equal to the frequency of the sinusoidal tone.
3. BandRejectSpan is the bandwidth of the band reject filter centered at SignalFrequency which is to be applied on the signal in order to estimate the *(N+D)* noise plus distortion power term.
   If BandRejectSpan is set equal to zero or less than 1/(Stop − Start), (that is, BandRejectSpan < 1/(Stop − Start)), the actual band reject span used internally in the SINAD measurement is set equal to 5/(Stop − Start).
4. HanningWindow is used to enable you the ability to window the signal before the estimation of the SINAD (*(S+N+D)* / *(N+D)*) is performed. If HanningWindow = ON, then a Hanning window is applied to the signal before the SINAD measurement is performed.
   Generally, set HanningWindow = ON to avoid excessive spectral leakage problems in the computation of the FFT of the signal.

   Note
   Due to a migration problem, when an ADS 1.5 (or earlier) design using this component is opened in a later ADS release, the HanningWindow parameter is set to OFF, no matter what the value is in the original design.

5. For general information regarding sinks, refer to *About Sinks* (sinks).

NumericSinkGated



Description: Numeric Data Sink with Control Input
Library: Sinks
Class: SDFNumericSinkGated

Parameters

Name	Description	Default	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None	enum	
Start	Sample number to start collecting data. DefaultNumericStart will inherit from the DF controller.	DefaultNumericStart	int	[0, ∞)
Stop	Sample number to stop collecting data. DefaultNumericStop will inherit from the DF controller.	DefaultNumericStop	int	[Start, ∞)
ControlSimulation	If set to YES, 'Stop' sample number determines how long the simulation will run: NO, YES	YES	enum	

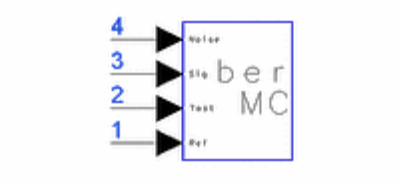
Pin Inputs

Pin	Name	Description	Signal Type
1	control	control signal	int
2	input	input signal	multiple anytype

Notes/Equations

1. The NumericSinkGated component is similar to the NumericSink component; it gives you more flexibility in controlling when data is collected. Refer to documentation for *NumericSink* (sinks) for information about the basic functionality.
2. When the control input is unconnected then this component behaves exactly the same as NumericSink.
   When the control input is connected then data is collected as described in *note 5* (sinks) of NumericSink documentation with the additional requirement that the control signal is not 0. To control the data collection only through the control pin set Start to 0 and ControlSimulation to NO. In this case, there must be at least one other source or sink component that controls how long the simulation runs.
   The use of a control signal enables you to capture one or more signal segments whose start and end is not known before the simulation starts but for which a binary signal can be generated that goes high when the segment starts and low when the segment ends. It can also significantly reduce the amount of collected data if the desired segments are spaced far apart from each other in a long simulation.

berMC4



Description: Error Probability measurement using Monte Carlo Method, 4 inputs
Library: Sinks
Class: TSDF_berMC4
Derived From: _berMC

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
ControlSimulation	if set to YES, 'Stop' time determines how long the simulation will run: NO, YES	YES		enum	
SymbolTime	symbol time	1.0	sec	real	[TStep, ∞)†
EstRelVariance	estimation relative variance (if set to 0.0 simulation will run until Stop is reached)	0.0		real	[0, 1\]]
NumThreshold	number of threshold	1		int	[1, ∞)
ThresholdSetting	type of threshold settings: automatically, manually	automatically		enum	
Threshold	threshold values when user selects manually setting	0		real array	
DelayBound	upper bound of delay for Synchronizing inputs; if DelayBound = -1, the Synchronizer is turned off	-1.0	sec	real	{-1} or (0, ∞)††
berOutput	Type of ber output: ber vs time, ber vs time every 10 symbols, ber vs time every 100 symbols, ber vs time every 1000 symbols, ber only	ber only		enum	
StatusUpdatePeriod	Status update period in number of bits	1000		int	[1, ∞)

† TStep is the simulation time step for the component input signals.
†† If DelayBound is set to -1, berMC will assume test and reference signals are already synchronized.

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed
2	input2	test signal input	timed
3	input3	receiving signal input	timed
4	input4	receiving noise input	timed

Notes/Equations

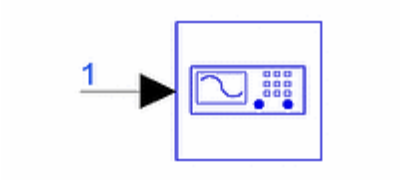
1. berMC4 works the same way as berMC and provides all features that berMC provides.

Also, berMC4 provides the feature of measuring the Es/No (energy per symbol to noise power spectral density ratio).

To measure Es/No, connect the pure signal (before noise is added) to berMC4 input pin 3 and the noise signal to input pin 4. This is only possible if the noise is generated by a single Noise source on the schematic and SummerRF is used to combine the signal and the noise. If the noise is generated in more than one place, for example, by setting resistor temperatures to values > -273.15°C, by setting NoiseFigure parameters of GainRF and/or MixerRF components, by setting the Loss parameter of filters to non-zero values, then it is more complicated, if not impossible, to separate the signal and the noise.

2. The Es term is calculated by measuring the power of the signal at input 3 and multiplying it by SymbolTime. The No term is calculated by measuring the power of the signal at input 4 and dividing it by the simulation bandwidth (1/TStep for RF signals, 1/(2 × TStep) for baseband signal; TStep is the simulation time step).
3. For general information regarding sinks, refer to *About Sinks* (sinks).

SpectrumAnalyzer



Description: Spectrum analyzer
Library: Sinks
Class: TSDF_SpectrumAnalyzer
Derived From: baseAnalysis

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start+16×TStep, ∞)†
Window	Window with default constant applied to collected data (default constant is used when WindowConstant is 0.0): none, Hamming 0.54, Hanning 0.50, Gaussian 0.75, Kaiser 7.865, _8510 6.0, Blackman, Blackman-Harris	none		enum	
WindowConstant	Window constant used for windows of type Hamming, Hanning, Gaussian, Kaiser, 8510	0.0		real	[0, ∞)
Bias	Spectrum normalization bias type to correct for effects of windowing and/or averaging: no bias, power	power		enum	
FStart	Start frequency for spectrum calculation	0.0	Hz	real	[0, ∞)
FStop	Stop frequency for spectrum calculation	100e9	Hz	real	(FStop, ∞)
NumFreqs	Number of frequencies uniformly spaced from FStart to FSTop; NumFreqs=0 results in default frequency resolution	0		int	[0, ∞)
NPoints	Number of points to be used in a segment; NPoints=0 results in one segment of (Stop-Start)/TStep+1 points and Overlap is ignored	0		int	[0, (Stop-Start)/TStep+1)†
Overlap	Number of overlapping points between two adjacent segments	0		int	[0, NPoints - 1\]]
† TStep is the simulation time step for the component input signal.					

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

1. The SpectrumAnalyzer component can be used to measure the spectrum of a baseband or an RF signal. In the following notes TStep is used to denote the simulation time step and fc is used to denote the signal characterization frequency (fc = 0 for a baseband signal and fc>0 for an RF signal).

   Note
   Information regarding time domain signal differences between ADS Ptolemy simulations and Circuit Envelope and Transient simulations is given in the *Timed Synchronous Dataflow* (ptolemy) section of the *ADS Ptolemy Simulation* (ptolemy) documentation.

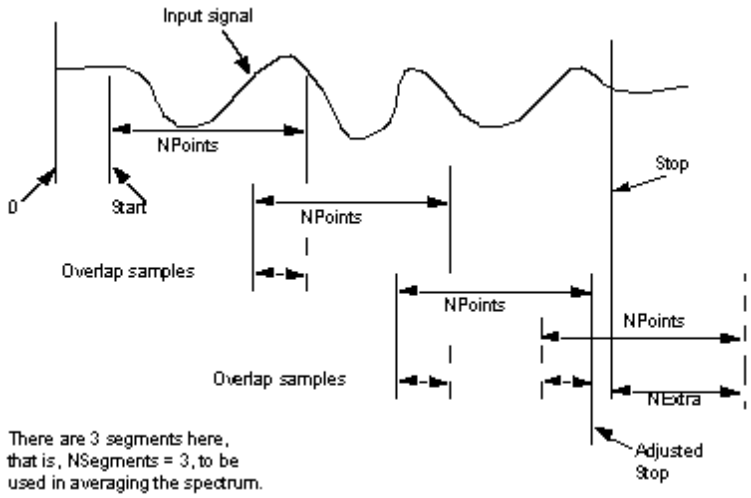
2. The component outputs the complex amplitude voltage values at the frequencies of the spectral tones. The component does not output the power at the frequencies of the spectral tones. However, you can calculate and display the power spectrum as well as the magnitude and phase spectrum by using the dBm, mag, and phase functions of the Data Display window. Note that the dBm function assumes a 50-ohm resistive load. If a different load was used in the simulation, its value can be specified as a second argument to the dBm function, for example, dBm(VRF, 75) where VRF is the instance name of the SpectrumAnalyzer component and 75 ohms is the resistive load used. To integrate the power spectrum over a frequency range see note 7.
   Note that, for baseband signals and for the frequency of 0 Hz the dBm function returns a power value that is 3 dB less than the actual power. This is because the primary use of the dBm function is with RF signals from an Analog/RF schematic simulation, where the 0 Hz frequency corresponds to the carrier frequency and not really 0 Hz signal frequency.
   If the baseband signal has no significant power at dc, this 3 dB error is insignificant and can be ignored-otherwise, it must be considered. For a baseband signal, the energy at 0 Hz is more typically measured as the average voltage value of the signal and not by power.

3. By default, the displayed spectrum extends from 0 Hz to $1/(2 \times TStep)$ Hz for a baseband signal and from $fc - 1/(2 \times TStep)$ Hz to $fc + 1/(2 \times TStep)$ Hz for an RF signal. When $fc < 1/(2 \times TStep)$, the default spectrum extends to negative frequencies. The spectral content at these negative frequencies is conjugated, mirrored, and added to the spectral content of the closest positive frequency. This way, the negative frequency tones are displayed on the positive frequency axis as would happen in an RF spectrum analyzer measurement instrument. This process can introduce an error in the displayed frequency for the mirrored tones. The absolute error introduced is less than deltaf / 2 (see note 5 for the definition of deltaf).
4. The basis of the algorithm used by SpectrumAnalyzer is the fs() function (the chirp-Z transform option of fs() is used). The algorithm enables fs() to be called on multiple signal segments and averages the results to achieve video averaging (see note 5). Also, the algorithm can scale the output spectrum to correct errors in power introduced by windowing.
   For details regarding the *fs()* function, refer to *Measurement Expressions* (expmeas).

5. The Start and Stop parameters define the time interval over which the measurement is performed. The number of data points collected is
   $$\frac{Stop - Start}{TStep} + 1$$
   If NPoints = 0, all collected data is processed as one segment. The assumptions made by fs() lead to better results if the segment size is odd. Therefore, if the total

number of data points collected is even, Stop is increased by TStep. The frequency resolution achieved in this case is *deltaf* = 1/(*Stop* - *Start*).
If NPoints > 0, the collected data is broken down in segments of size NPoints, the spectrum of each segment is calculated and at the end all the spectra are averaged (video averaging). For the same reason explained above, if NPoints is even, it is increased by 1 to make it odd. The frequency resolution achieved in this case is *deltaf* = 1/(*TStep* × *NPoints*). If Overlap = 0, the Npoints-long segments are non-overlapping, otherwise these overlap by Overlap points.
Since it is very unlikely that an integer number of Npoints-long segments that overlap by Overlap points fits exactly in the interval from Start to Stop, Stop can be increased/decreased so that the resulting number of segments is an integer. Let NExtra be the number of extra points that need to be collected in order to have an integer number of segments. If NExtra < Npoints / 2, Stop is increased by *NExtra* × *TStep* , otherwise it is decreased by *(NPoints - NExtra - Overlap*) × *TStep* .
The following figure shows the multiple segments and video averaging concepts discussed above.

Input Signal and SpectrumAnalyzer Parameters Start, NPoints, and Overlap



6. The Window and WindowConstant parameters are used to define the window that is applied to each segment before its spectrum is calculated. Windowing is often necessary in transform-based (chirp-Z, FFT) spectrum estimation. Without windowing, the estimated spectrum can suffer from spectral leakage that can cause misleading measurements or masking of weak signal spectral detail by spurious artifacts.
Window Type Definitions
Hanning Window:

$$w(kT_s) = \begin{cases} W-(1-W)\cos\left(\frac{2\pi k}{NPoints}\right) & 0 \le k \le NPoints \\ 0.0 & otherwise \end{cases}$$

where W = WindowConstant
Hamming Window:

$$w(kT_s) = \begin{cases} W-(1-W)\cos\left(\frac{2\pi k}{NPoints}\right) & 0 \le k \le NPoints \\ 0.0 & otherwise \end{cases}$$

where W = WindowConstant
Blackman Window:

$$w(kT_s) = \begin{cases} 0.42-0.5\cos\left(\frac{2\pi k}{NPoints}\right)+0.08\cos\left(\frac{4\pi k}{NPoints}\right) & 0 \le k \le NPoints \\ 0.0 & otherwise \end{cases}$$

Blackman-Harris Window

$$w(kT_s) = \begin{cases} 0.36-0.49\cos\left(\frac{2\pi k}{NPoints}\right)+0.14\cos\left(\frac{4\pi k}{NPoints}\right)-0.0117\cos\left(\frac{6\pi k}{NPoints}\right) & 0 \le k \le NPoints \\ 0.0 & \text{otherwise} \end{cases}$$

Kaiser (Kaiser-Bessel) Window:

$$w(kT_s) = \begin{cases} \frac{I_0\left(\beta\left[1-\left(\frac{k-\alpha}{\alpha}\right)^2\right]^{1/2}\right)}{I_0(\beta)} & 0 \le k \le NPoints \\ 0.0 & otherwise \end{cases}$$

Here α = NPoints / 2, β is the shape parameter (set by WindowConstant), and I_0 (.) is the 0th order modified Bessel function of the first kind.
8510 6.0 Window
This is a Kaiser window with β = 6.0
Gaussian (Weierstrass) Window:

$$w(kT_s) = \begin{cases} exp\left(-\left(\alpha\pi\left(\frac{2k-N}{N}\right)\right)^2\right) & 0 \le k \le N \\ 0 & otherwise \end{cases}$$

where α = WindowConstant.

Window Options

Available window options (some with default constants) are:

Hamming 0.54

anning 0.50

Gaussian 0.75

Kaiser 7.865

8510 6.0

Blackman

Blackman-Harris

The default window constants for Hamming, Hanning, Gaussian, Kaiser, and 8510 can be changed by using the WindowConstant parameter.
As an example, if you are using the SpectrumAnalyzer component with the parameter *Window = Gaussian 0.75* , you may wonder if you need to explicitly set the parameter *WindowConstant = 0.75* , or if you can just accept the default *WindowConstant = 0.0* . If you want to use the default constant value (for example, 0.75 for the *Window = Gaussian 0.75*), then you do not need to modify the WindowConstant parameter at all. In this example, WindowConstant = 0.75 is equivalent to WindowConstant = 0.0.
The point here is that if you leave *WindowConstant = 0.0* , then the default value of the Window parameter (e.g. Hamming 0.54, Hanning 0.50, Gaussian 0.75, etc.) will be used.

7. The windowing of a signal in time has the effect of changing its power. The Bias parameter can be used to compensate for this. If Bias is set to no bias, then no normalization of the spectrum is done. If Bias is set to power, then the spectrum is normalized so that the power contained in it is the same as the power of the input signal.
   To calculate the power contained in a spectrum, the spec_power() function can be used in the Data Display window. The *spec_power()* function expects its input to be in dBm and returns a value in dBm. Therefore, if spectral data generated using the SpectrumAnalyzer is passed to the spec_power() function, the dBm function must be applied to the data before spec_power() is called. If frequency limits needs to be passed to spec_power(), specify these in Hz.
   For details regarding the *spec_power()* function, refer to refer to *Measurement Expressions* (expmeas).
8. How to choose the right window.
   Every time a window is applied to a signal (Window = none effectively applies a rectangular window to the signal), leakage occurs, that is, power from one spectral component leaks into the adjacent ones. Leakage from strong spectral components can result in hiding/masking of nearby weaker spectral components. Even strong spectral components can be affected by leakage. For example, two strong spectral components close to each other can appear as one due to leakage.
   Choosing the right window for a spectral measurement is very important. The choice of window depends on what is being measured and what the trade-offs between frequency resolution (ability to distinguish spectral components of comparable strength that are close to each other) and dynamic range (ability to measure signals with spectral components of widely varying strengths and distributed over a wide range) are.
   Windows can be characterized by their Normalized Equivalent Noise Bandwidth (NENBW). Equivalent noise bandwidth (ENBW) compares the window to an ideal, rectangular filter. It is the equivalent width of a rectangular filter that passes the same amount of white noise as the window. The normalized ENBW (NENBW) is the ENBW multiplied by the time duration of the signal being windowed (see *SpectrumAnalyzerResBW* (sinks) documentation for a table of NENBW values for the supported windows). In general, for the same length of signal processed, the higher the NENBW of a window the higher its dynamic range (less leakage) and the poorer its frequency resolution.
   Some general guidelines for choosing a window are given below:
   - Do not use a window (Window = none) when analyzing transients.

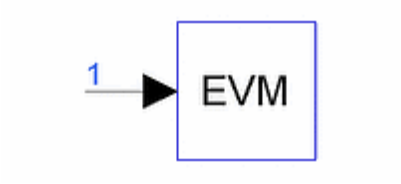
- For periodic signals whose spectral components have comparable strengths and when the signal segment processed includes an exact integer multiple of periods, the best results are obtained if no window is used (Window = none, which is equivalent to using a rectangular window). Any start up transients should be excluded.
- For periodic signals whose spectral components have significantly different strengths and/or when the signal segment processed does not include an exact integer multiple of periods, the use of a window can improve the detection of the weaker spectral components. The higher the NENBW the more likely the weaker spectral components will be detected. However, this trades-off frequency resolution and so if the spectral components are very close to each other the weaker one might remain unresolved. To improve frequency resolution while still maintaining a good dynamic range use a window but process a longer signal segment.
- For aperiodic signals such as modulated signals (QPSK, QAM, GSM, EDGE, CDMA, OFDM) the use of a window is highly recommended. The window will attenuate the signal at both ends of the signal segment processed to zero. This makes the signal apear periodic and reduces leakage.

9. The FStart and FStop parameters control the frequency range over which the spectrum is calculated overriding the default values (see note 3). If the values of these parameters are outside the valid range
   - [0, 1/(2 × *TStep*)] for a baseband signal
   - [*fc* - 1/(2 × *TStep*) , *fc* + 1/(2 × *TStep*)] for an RF signal
   FStart is forced to the lower limit of the valid range and FStop is forced to the upper limit of the valid range.
10. The NumFreqs parameter controls how many spectral tones is calculated between FStart and FStop. The default value of 0 results in (*FStop* - *FStart*)/(*deltaf*) +1 spectral tones being calculated (see note 5 for the definition of deltaf). If *(FStop* - *FStart*)/(*NumFreqs*) < *deltaf* then NumFreqs is forced to *(FStop* - *FStart*)/(*deltaf*) + 1.
11. For general information regarding sinks, refer to *About Sinks* (sinks).

References

1. A. V. Oppenheim, R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 1989, Chapter 9, section 9.7.

EVM



Description: Error Vector Magnitude Measurement
Library: Sinks
Class: TSDF_EVM
Derived From: baseAnalysis

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
SymTime	symbol duration time	1e-3	sec	real	[TStep, ∞)†
SymBurstLen	burst length in number of symbols	100		int	[1, ∞)
MeasType	options for measurement type: EVM RMS, EVM Peak, C0, Frequency error, Droop, Error sequence, Magnitude error sequence, Phase error sequence, Sampled data	EVM RMS		enum	
ModType	options for modulation type of input signal: BPSK, QPSK, HPSK, PI/4 DQPSK, PSK8, PSK16, QAM4, QAM16, QAM32, QAM64, QAM128, QAM256, PAM4, PAM8, User defined	QPSK		enum	
Constellation	used if ModType="User defined" to define constellation as an array of complex values (re, im)	(1,1) (1,-1) (-1,-1) (-1,1)		complex array	
OptimizeSamplingInstant	if YES, optimal sampling instant will be automatically found: NO, YES	YES		enum	
† TStep is the simulation time step for the component input signal.					

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

1. EVM (Error Vector Magnitude) measurements are used to evaluate the modulation accuracy of modulators/transmitters (to measure EVM for a demodulator/receiver see Note 5). EVM is used, for example, in the IS-54 TDMA digital cellular to set the minimum specifications for modulation accuracy of π/4-DQPSK modulators.
   The defining equations for the EVM measurement follow the definition in the *EIA/TIA IS-54-B TDMA Cellular System Dual-Mode Mobile Station-Base Station Compatibility Standard, section 2.1.3.3.1.3.3 (Error Vector Magnitude Requirement).* Let Z(k) denote the actual complex vectors (I and Q) produced by observing the real

transmitter through an ideal receiver filter at instants k, one symbol period apart. S(k) is defined as the ideal reference symbol (normalized such that its maximum energy symbol falls on the unit circle). Then, Z(k) is modeled as:

$$Z(k) = [C_0 + C_1(S(k) + E(k))]W^k$$

where

W = e $^{Dr+jDa}$, accounts for both a frequency offset (Da radians/symbol phase rotation) and an amplitude change rate (of Dr nepers/symbol)

C $_0$ is a complex constant origin offset (in Volts)

C $_1$ is a unitless complex constant representing the arbitrary phase and output power of the transmitter, and

E(k) is the residual vector error on sample S(k) (in Volts)

The sum square error vector is

$$\sum_{k=0}^{N-1} |E(k)|^2 = \sum_{k=0}^{N-1} \left| \frac{[Z(k)W^{-k} - C_0]}{C_1} - S(k) \right|^2$$

where N is equal to SymBurstLen and C $_0$, C $_1$, W are chosen such as to minimize the above expression.

EVM (rms) is defined to be the rms value of | E(k) | normalized by the rms value of | S(k) |. Therefore,

$$EVM(rms) = \frac{\sqrt{\frac{1}{N} \cdot \sum_{k=0}^{N-1} |E(k)|^2}}{\sqrt{\frac{1}{N} \cdot \sum_{k=0}^{N-1} |S(k)|^2}} = \sqrt{\frac{\sum_{k=0}^{N-1} |E(k)|^2}{\sum_{k=0}^{N-1} |S(k)|^2}}$$

The symbol EVM at symbol k is defined as

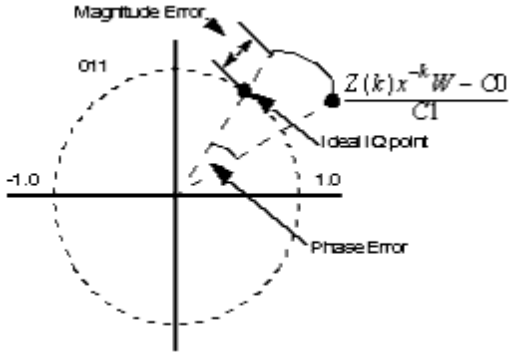
$$EVM(k) = \frac{|E(k)|}{\sqrt{\frac{1}{N} \cdot \sum_{k=0}^{N-1} |S(k)|^2}}$$

which is the vector error magnitude at symbol k normalized by the rms value of | S(k) |.

2. The MeasType parameter determines the output of the component.
   - If MeasType = EVM RMS, the output is the EVM(rms) value as defined in the equations of note 1. This value is not in percent.To get EVM RMS in percent, multiply by 100.
   - If MeasType = EVM Peak, the output is max{ EVM(k) }, where the maximum is evaluated over k = 0, 1, ... , N-1. This value is not in percent.To get EVM Peak in percent, multiply by 100.
   - If MeasType = C0, the output is

$20 \times \log 10\left(\frac{|C0|}{RMS(|S(k)|)}\right)$ dB

- If MeasType = Frequency error, the output is $(Da)/(2 \times \pi \times SymTime)$ Hz
- If MeasType = Droop, the output is $(-20 \times \log 10(e^{Dr}))$ dB
- If MeasType = Error sequence, the output is the sequence $\frac{Re\{E(k)\} + j \times Im\{E(k)\}}{RMS(|S(k)|)}$
- If MeasType = Magnitude Error sequence, the output is the magnitude error sequence (in Volts). For the definition of magnitude error, see the following figure.
- If MeasType = Phase Error sequence, the output is the phase error sequence (in degrees). For the definition of phase error, see the following figure.
- If MeasType = Sampled data, the output is the downsampled (1 sample per symbol period) input signal.



Magnitude and Phase Error

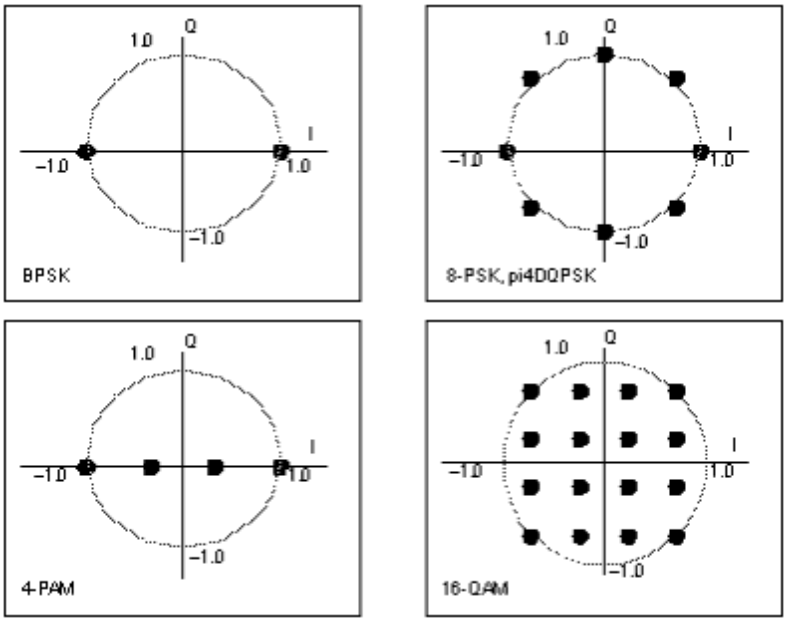
3. The ModType parameter is used to select one of the pre-defined or a user-defined signal constellation. All signal constellations (including user- defined) are normalized so that the maximum magnitude constellation point lies on the unit circle. For example, for a QPSK constellation, all four IQ points lie on the unit circle.
   For a PAM-type signal (includes BPSK, 4-PAM, 8-PAM) the Q-channel of the signal is set to 0.0.
   The following figure shows a few examples of the IQ constellation sets used in the EVM measurement.
   The HPSK option is for an HPSK signal that has equal gains applied to the I and Q channels resulting in a 4-point constellation. In effect, this is the same as a QPSK constellation. If the gains applied to the I and Q channels are different, the resulting constellation has 8 points and using the HPSK option gives the wrong results; in this case, the *User defined* option should be used, which enables you to specify an arbitrary constellation in the Constellation parameter.
4. The Start and OptimizeSamplingInstant parameters define how the input signal is downsampled. Note that the EVM algorithm operates on the downsampled input signal.
   If OptimizeSamplingInstant is set to NO, the input signal is downsampled at one sample per symbol period starting at Start. If Start is not an exact multiple of the

simulation time step, interpolation is used to find the input signal values in between the available samples.

If OptimizeSamplingInstant is set to YES, the input signal is downsampled at one sample per symbol period starting at StartDownSampling, where StartDownSampling is swept from Start to (Start + SymTime) with a step of (simulation time step) / 10. This way the optimal sampling instant can be found. The optimal value for StartDownSampling is displayed on the status window and can be used in subsequent simulations as the value of Start with OptimizeSamplingInstant set to NO. This setting results in faster simulations and gives the correct results as long as no parameter that can affect the optimal sampling instant (for example the delay of a filter) is changed. If such a parameter changes, then OptimizeSamplingInstant should be set to YES.

Another way to find when the optimal sampling instant is, without setting the OptimizeSamplingInstant parameter to YES, is to observe the eye diagram and find where its maximum opening occurs.

Start must be set to a value large enough so that any initial transients are excluded. If any part of the transient is included in the EVM measurement the EVM result is going to be higher than expected.



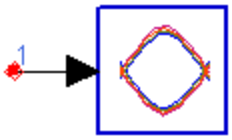
Examples of IQ-Constellations as Used in EVM Measurement

5. EVM is a measurement typically performed for a modulator/transmitter. Therefore, the input for this component is a complex envelope signal that is typically generated at the output of a modulator, e.g. *QAM_Mod* (timed), *QPSK_Mod* (timed). In recent years, EVM measurements are used more and more often for receivers to get a rough estimate of their BER performance. To measure EVM at the output of a demodulator, the I/Q signals at the output of the demodulator should be combined to form a complex envelope signal that is required at the input of the EVM component. To do this, just connect the I/Q signals at the output of the demodulator to the I/Q inputs of an ideal QAM modulator (*QAM_Mod* (timed); default parameter values will work) and then connect the output of the QAM modulator to the input of the EVM component. If the demodulator I/Q outputs are connected to other parts of your design you may need to use *SplitterRF* (timed) components to make sure the

QAM_Mod (timed) input resistances do not undesirably modify the load for the rest of your design.

6. For general information regarding sinks, refer to *About Sinks* (sinks).

StatEye



Description: Statistical Eye Diagram
Library: Sinks
Class: TSDF_StatEye
C++ Code: See *doc/sp_items/TSDF_StatEye.html* under your installation directory.

Parameters

Name	Description	Default	Unit	Type	Range
RIn	Input resistance	DefaultRIn	Ohm	int	(0, ∞)
RTemp	Temperature, in degrees C	DefaultRTemp	Celsius	real	[-273.15, ∞]
BaudRate	Baud rate in the unit of GBps (giga baud per second)	11.1		real	
SampleNum	Number of input timed samples for calculating StatEye	256		int	
DJ	Deterministic jitter peak to peak value	0.15		real	
RJ	Random jitter rms value	0.15/(2*7.94)		real	
PreCursors	Number of precursors	4		int	
PostCursors	Number of postcursors	90		int	
BinNum	Number of bins	4000		int	
RequiredDJ	Required deterministic jitter for jitter compliance	0.45		real	
RequiredQ	Required Q factor for jitter compliance	2*7.94		real	
RequiredTJ	Required total jitter for jitter compliance	0.6		real	
RequiredEye	Required minimum vertical eye opening for standards compliance	0.4		real	
TargetBER	Target bit error rate with which the eye opening is calculated	1e-12		real	
BERDisplay	BER (= 10^BERDisplay) to be displayed	{-15, -6}		real array	
CDREnabled	CDR enabled or not: NO, YES	NO		enum	
DFEEnabled	DFE enabled or not: NO, YES	NO		enum	
DFEtaps	Number of DFE taps	5		int	
SaveTapsFile	Filename in which to save final tap values			string	
FFEEnabled	FFE enabled or not: NO, YES	NO		enum	
FFEtaps	Number of FFE taps	3		int	
FFEtapsOpt	Number of FFE taps automatically optimized or not: NO, YES	NO		enum	
SaveFFETapsFile	Filename in which to save final FFE tap values			string	

Pin Inputs

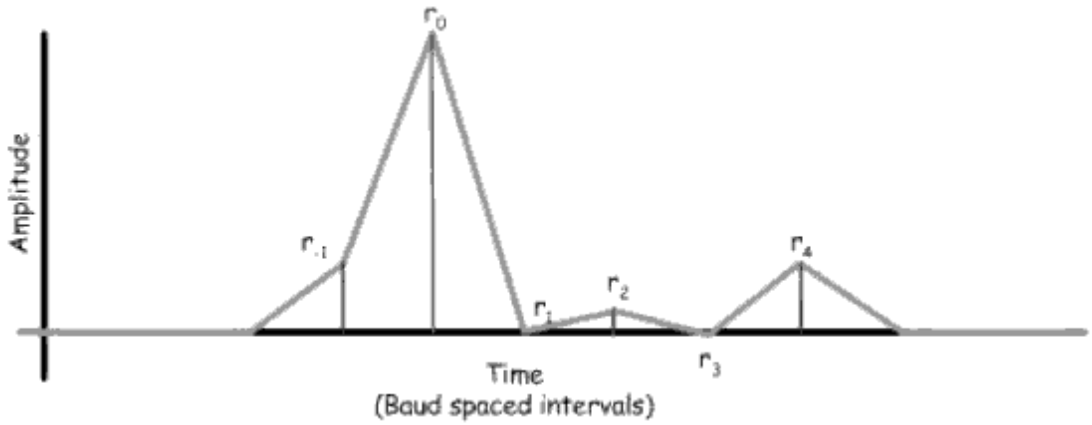
Pin	Name	Descriptions	Signal Type
1	in	input Rx signal	timed

Notes/Equations

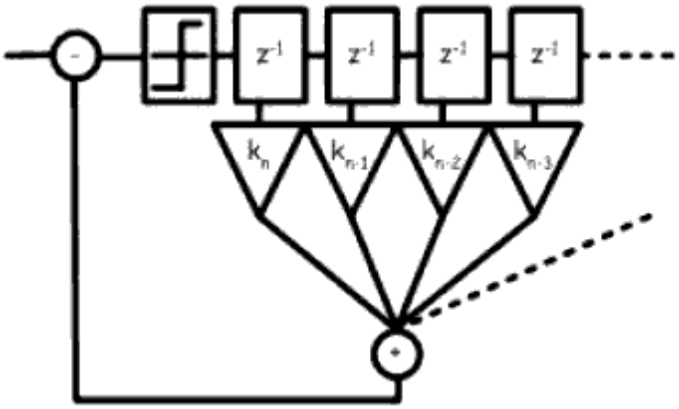
1. This model is the statistical eye sink. The statistical eye simulation of a high-speed digital channel is a much faster way to predict the eye diagram and BER performance with reasonable accuracy and without having to simulate large numbers of bits. The statistical eye provides contour plots for a specific BER level. Bathtub plots clearly show the eye width or eye height at different BER levels by intercepting the statistical eye along the timing or amplitude axis.
2. Jitter measurement is performed. In a digital communications channel, jitter is the time deviation from ideal timing of a signal transition through a decision threshold (essentially variation in the zero crossing times of the data eye). Two general types of jitter are characterized: deterministic jitter (DJ) and random jitter (RJ). Because each type accumulates differently in the channel, they are characterized independently.
    - Deterministic jitter (DJ) is generally bounded in amplitude, non-Gaussian, and expressed in units of time (Unit-Interval), peak to peak. These are examples of deterministic jitter:
        - Duty-cycle distortion - e.g., from asymmetric rise/fall times.
        - Intersymbol interference (ISI) - e.g., from channel dispersion or filtering.
        - Sinusoidal - e.g., from power-supply feedthrough.
        - Uncorrelated - e.g., from crosstalk by other signals.
    - Random jitter (RJ) is assumed to be Gaussian in nature and accumulates from thermal noise sources. Because peak-to-peak measurements take a long time to achieve statistical significance, random jitter is measured as an RMS (root mean square) value. Multiple random-jitter sources add in an RMS fashion, but a peak-to-peak value is needed when adding random jitter to deterministic jitter to get total jitter, peak to peak. Although Gaussian statistics imply an "infinite" peak-to-peak amplitude, a useful peak-to-peak value can be calculated from the RMS value after a probability of exceeding the peak-to-peak value is chosen. The standard deviation (RMS value) is chosen. For example, the peak-to-peak random jitter having less than 10^{-15} probability of being exceeded is 15.883 times the RMS value.
    - DJ and RJ are used at the function *genJitGauss* shown in StatEye Code Structure figure.
3. Currently only an ideal pulse source is supported as the input to the channel. A pulse source timed transmission signal passes through a channel and then the receiver signal is fed to the sink. The receiver signal should be scaled as 1 volt before entering the sink by adjusting *VPlateau*. With the default settings in the demo example, you do not need to set the pulse source. The *Delay* can be set as 0 or easily adjusted with *N*TimeStep*. The *RepetitionInterval* should be larger than *TimeStop*. For more information, refer to the *Pulse* (timed) model documentation.
4. Decision Feedback Equalizer (DFE) is supported. A DFE is a nonlinear equalizer insofar that it cannot be represented in the frequency domain. The concept of a DFE lies in its ability to cancel the post-cursors of the channel pulse response. If a pulse response is defined in terms of its amplitude at baud spaced samples (see the first of the following two figures), where r_n with $n<0$ are called pre-cursors and r_n with $n>0$ are called post-cursors, the DFE can cancel the inter-symbol interference (ISI)

caused by post-cursors. Given that the channel response is known and an equalized signal has been correctly received as a 1 or 0, then referring to the second of the following two figures, the influence of the post-cursors can be removed by setting $k_n = r_n$. Since the DFE only feeds back a decision, any noise present on the signal is not amplified. However, in the case where the noise causes a decision error, this error is then fed back to the receiver and can cause error propagation.

Pulse Response Definition

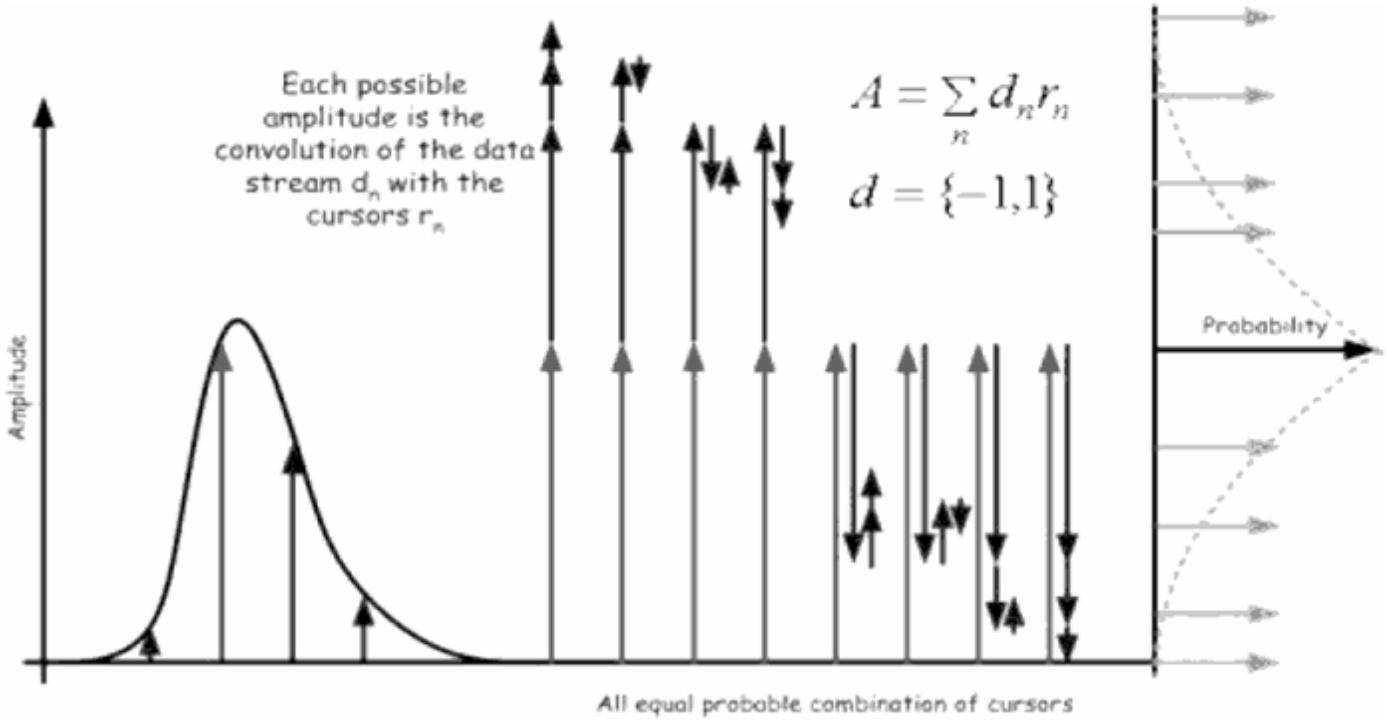


DFE Architecture



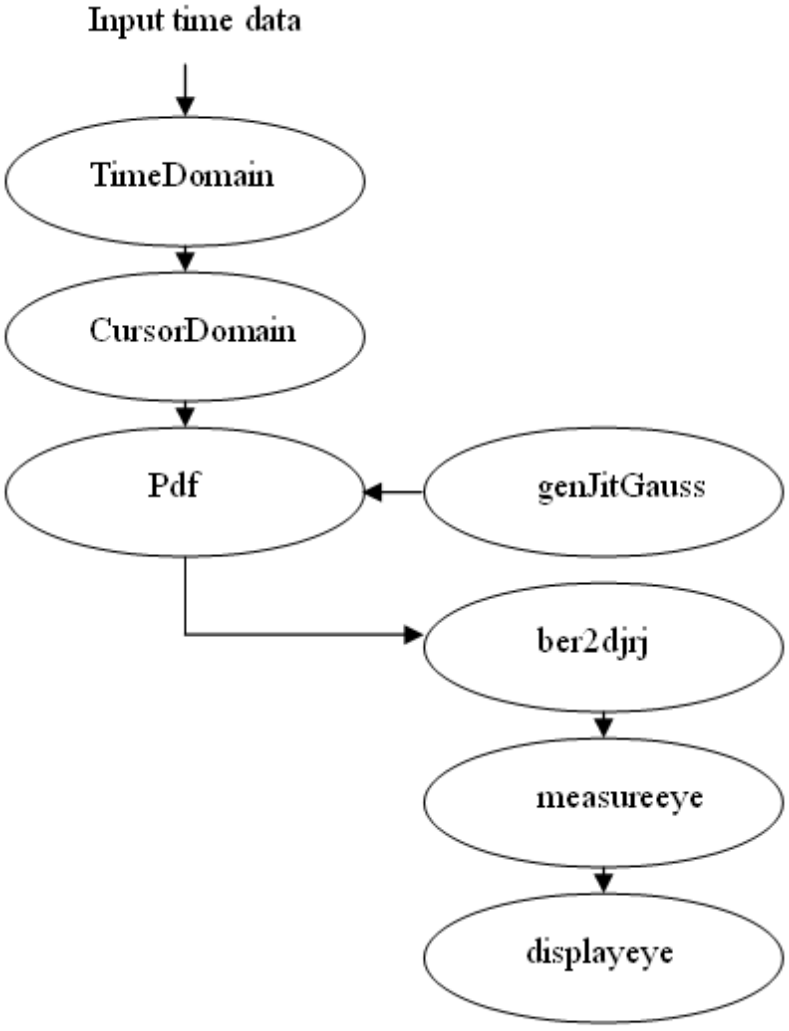
5. The superposition of post-cursors and pre-cursors to form ISI is statistical in nature. Given a full random binary data stream and a finite number of cursors, each possible combination of cursors can superimpose each with equal probability. The following figure shows a simple example with an arbitrary sample time that has one pre-cursor and two post-cursors. The example shows how the eight possible combinations of the cursors can combine to cause ISI. In this example, the ISI is as large as the signal itself and closes the eye. The probability of each amplitude can be represented by a Conditional Probability Distribution function which; for this example, is simply eight diracs or deltas of equal probability; that is, 1/8. As more cursors are taken into account, the number of possible combinations increases and the Probability Distribution becomes more detailed. This can be represented mathematically taking into account the effect of the DFE cancellation of the post-cursors.

ISI PDF



6. Clock data recovery (CDR) is supported to calculate the position of the center of the eye. When CDR is enabled, the center of the eye is determined by finding the mean of the edge distributions; that is, the bathtub. This is somewhat like how a real CDR works. When disabled, the eye center is found through the maximum eye of the eye, typical to DFE implementations. In either case, the displayed eye is centered on the peak of the pulse response.
7. Feed Forward Equalizer (FFE) is supported. The FFE is used to compensate the signal distortion caused by the channel. Two choices are provided when FFE is enabled. One choice is to set the number of FFE taps yourself; however, sometimes the FFE may not work or this may cause a bad signal. Another choice is to use the automatic optimization feature, as set by parameter *FFEtapsOpt*. When *FFEtapsOpt* is set to *YES*, the optimal number of FFE taps are calculated and output in the hpeesofsim status and the dds. Note the CDR must be enabled if *FFEtapsOpt* is set to *YES*.
8. The original model was developed with C++ and MATLAB ™ code in http://www.oiforum.com/public/Open_Software.html . This ptolemy model utilizes revised C++ code and rewritten MATLAB ™ code in ADS. The figure below shows the code structure in ADS.

StatEye Code Structure



9. The StatEye model source code can be found in the directory doc/sp_items under your ADS installation directory. The name of the source file is *TSDF_StatEye.pl*. The revised .cc and .h files from the C++ code can be found in the directory *adsptolemy/src/serdes/stateye* under your ADS installation directory.
10. Parameter Details:
    - *RIn* - specifies the input resistance.
    - *RTemp* - specifies the temperature, in degrees C.
    - *Baudrate* - specifies the baud rate in the unit of GBps (giga baud per second). It should match the channel. For different *Baudrate*, the channel pulse responses are different which affect the sink results.
    - *SampleNum* - specifies the number of timed samples for calculating StatEye. It is the number of input time samples. The effective pulse response should be included. When the effective pulse response information are contained, it does not affect the results. The default value is 8192.
    - *DJ* - specifies the deterministic jitter peak to peak value. The DJ and RJ parameters define the dual Dirac jitter distribution used to convolve with the resulting channel ISI to create the final eye.
    - *RJ* - specifies the random jitter rms value. The default value is 0.1/(2*7.94).

- *PreCursors* - specifies the number of precursors.
- *PostCursors* - specifies the number of postcursors. PreCursors and PostCursors affect the results and are set by experience. Mainly, there are small pre-cursors in electrical systems, and the reflections have died away beyond 90 UI.
- *BinNum* - specifies the number of bins for PDF calculation. It describes the number of segments used in the histogram or probability density function during the calculation of the channel ISI. Clearly this value must be high enough so as to encompass the smallest crosstalk aggressor, and a warning is generated if this is not the case. Typical value: 1000.
- *RequiredDJ* - specifies the required deterministic jitter for jitter compliance. *RequiredDJ*, *RequiredQ*, *RequiredTJ* and *RequiredEye* do not influence any receiver characteristics/properties, as far as analysis is concerned - they are only used for compliance check at the end of an analysis run.
- *RequiredQ* - specifies the required Q factor for jitter compliance.
- *RequiredTJ* - specifies the required total jitter for jitter compliance.
- *RequiredEye* - specifies the required minimum vertical eye opening for standards compliance.
- *TargetBER* - specifies the target bit error rate for which the eye opening is calculated.
- *BERDisplay* - specifies which BER ($=10^{\text{BERDisplay}}$) to be displayed.
- *CDREnabled* - specifies CDR enabled or not.
- *DFEEnabled* - specifies DFE enabled or not.
- *DFEtaps* - specifies Number of DFE taps.
- *SaveDFETapsFile* - specifies the filename in which to save final DFE tap values. If the *SaveDFETapsFile* string is non-null, a file will be created with the name given by that string, and the final tap values will be stored there after the run has completed.
- *FFEEnabled* - specifies FFE enabled or not.
- *FFEtaps* - specifies Number of FFE taps.
- *FFEtapsOpt* - specifies Number of FFE taps automatically optimized or not.
- *SaveFFETapsFile* - specifies the filename in which to save final FFE tap values. If the *SaveFFETapsFile* string is non-null, a file will be created with the name given by that string, and the final tap values will be stored there after the run has completed.

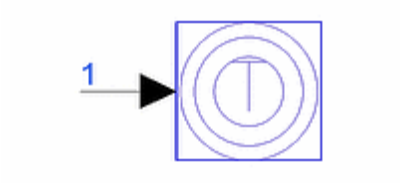
11. The StatEye demo is documented under the *Signal Integrity* section of the ADS Examples Documentation. You can refer to the example for the output of the component. The opening of the eye (calculated using Statistical Eye Analysis methods) will be confirmed within the requirements at the required BER of the Implementation Agreement; usually,
    - Amplitude at the zero time offset sampling point.
    - Time jitter measured at the zero amplitude sampling point.
12. Output Details
    - *AftEqu* is the signal after equalization.
    - *Amp* is the signal amplitude.
    - *BER* is the measured Bit Error Rate (of which the bathtub curve consists) for various sampling offsets.
    - *BER1*, *BER2*, etc. specifies the Bit Error Rate (= 10^{BER1} or 10^{BER2}, etc.) to be displayed in eye plot.
    - *CDF* is the cumulative distribution function of jitter signal.
    - *CDR* is the offset between the eye center and zero.

- *Contour* is a Complex value with the real part (x-axis) as time interval and the imaginary part (y-axis) as signal amplitude. So function real() and imag() are used when plot. In the dds equations, *Num1* is the number of contour data for *BER1* and *Data1* are the data used for *BER1* contour plot, and so on. To show the contour for *Data2*, just change *Data1* in the Eye contour to *Data2*. Thus the eye contour for specified *BER1*, *BER2* and etc. can be drawn.
- *DJ* is the measured deterministic jitter value.
- *eyeopen* is the measured eye opening value, which should be confirmed to be within the requirements at TargetBER.
- *mask_x* specifies the x-axis coordinates that draw the eye mask. They are +/- (1 - RequiredTJ) / 2.
- *mask_y* specifies the y-axis coordinates that draw the eye mask. They are +/- RequiredEye.
- *Q* is the Q factor corresponding to the measured BER.
- *RequiredDJ* is the output of parameter RequiredDJ to show the compliance.
- *RequiredEye* is the output of parameter RequiredEye to show the compliance.
- *RequiredTJ* is the output of parameter RequiredTJ to show the compliance.
- *RJ* is the measured random jitter value.
- *RxSignal* is the input signal to the StatEye sink.
- *Time_UI* is time interval.
- *TxSignal* is the ideal pulse signal which is input to the channel.

References

1. *Common Electrical I/O (CEI) - Electrical and Jitter Interoperability agreements for 6G+ bps and 11G+ bps I/O IA*, OIF-CEI-02.0, 28th February 2005.
2. Anthony Sanders, *Channel Compliance Testing Utilizing Novel Statistical Eye Methodology*, DesignCon 2004.
3. http://www.oiforum.com/public/Open_Software.html

TimedSink



Description: Timed Data Collector
Library: Sinks
Class: TSDF_TimedSink
Derived From: baseSink

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
ControlSimulation	if set to YES, 'Stop' time determines how long the simulation will run: NO, YES	YES		enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	timed

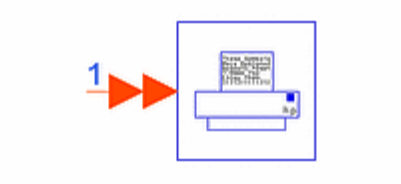
Notes/Equations

1. TimedSink collects timed (baseband or complex envelope) data from the output of the connected component and saves it to the simulation dataset. Timed baseband data is in the form of real voltage values versus time. Timed complex envelope data is in the form of complex voltage values versus time.
2. The name of the variable (holding the simulation data) that is created in the dataset by each TimedSink is the same as the sink's instance name.
3. In addition to the voltage data, the TimedSink component also saves the input signal characterization frequency as an attribute named *fc*. The ael expression *get_attr*( ) can be used to retrieve the value of the characterization frequency. For example, if the sink instance name is T1, then fc1 = *get_attr*(*T1*, "*fc*") returns the

characterization frequency of the signal T1.

4. The amount of data collected by a TimedSink is controlled by the Start, Stop, and ControlSimulation parameters. Data collection always begins at the time instant specified by Start.
    - If ControlSimulation is set to Yes, then data collection ends at the time instant specified by Stop.
    - If ControlSimulation is set to No, then data collection ends when the simulation ends; in this case, there must be at least one other source or sink component that controls how long the simulation runs.
5. For general information regarding sinks, refer to *About Sinks* (sinks).

Printer



Description: Plain output data file writer
Library: Sinks
Class: SDFPrinter
C++ Code: See *doc/sp_items/SDFPrinter.html* under your installation directory.

Parameters

Name	Description	Default	Type	Range
Start	sample Index to start recording data	DefaultNumericStart	int	[0, ∞)
Stop	sample Index to stop recording data	DefaultNumericStop	int	[Start, ∞)
ControlSimulation	control simulation: NO, YES	YES	enum	
FileName	output file name	print.txt	filename	

Pin Inputs

Pin	Name	Description	Signal Type
1	input		multiple anytype

Notes/Equations

1. Printer prints out one sample from each input port per line.
2. The output file is a text file that contains data in ADS Ptolemy format specific to the input data type: real array (for floating-point (real), fixed, and integer scalar input data), complex array, string array, real matrix, integer matrix, fixed-point matrix, or complex matrix. For format information, refer to *Understanding File Formats* (ptolemy).
3. For general information regarding sinks, refer to *About Sinks* (sinks).

About Sinks

The Sinks library contains sinks that accept input from numeric or timed components. Sinks produce unprocessed data or perform measurement algorithm specific processing and produce processed data.

Data received by a sink can be numeric (scalar, fixed, or matrix) or timed (baseband or RF). All data is stored in double-precision floating-point (real) or complex numerical format. Some sinks accept numeric and timed data; other sinks accept specifically either numeric or fixed or timed data; refer to the following table for details.

Note
Information regarding time domain signal differences between ADS Ptolemy simulations and Circuit Envelope and Transient simulations is given in the *Timed Synchronous Dataflow* (ptolemy) section in the *ADS Ptolemy Simulation* (ptolemy) documentation.

Sinks Summary

Sink Component	Possible Input Data Type	Storage Medium
berMC	timed	binary dataset
berMC4	timed	binary dataset
berIS	timed	binary dataset
EVM	timed	binary dataset
NumericSink	numeric or timed (partial data stored)	binary dataset
Printer	numeric or timed	ASCII data file
Sinad	timed	binary dataset
SpectrumAnalyzer	timed	binary dataset
SpectrumAnalyzerResBW	timed	binary dataset
TimedSink	timed	binary dataset
Obsolete Sink Component	Equivalent New Component	
FFTAnalyzer	SpectrumAnalyzer	
SpecAnalyzer	SpectrumAnalyzer	
StatEye	timed	binary dataset
OutFile	TimedDataWrite (for .tim, .bintim, .ascsig, .sig file types);SDFWrite (for .sdf file types) in the Instruments Library	
ErrVecMeas	EVM	
WriteVar	not available	

Connection of a Timed Component Output to a Sink

Timed components have the RLoad and RTemp parameters.

- By default, RLoad specifies the resistive load DefaultRLoad, which is a reference to a VAR expression of that name, or a reference to the DF controller parameter of that name that is set to an infinite value. If you expect to collect timed data into a matched resistive load, you must specify the resistive value to the RLoad parameter. Changes on the DF controller affect any other timed sink that has RLoad = DefaultRLoad. Changes done on a particular sink only affect that sink.
- RTemp has a usage similar to the RLoad parameter.

Sink Data in Binary Dataset

The abbreviated dataset name is visible in the Data Display dialog boxes for the various plot types (Linear, Polar, Smith, Stack, List).

When accessing data stored in a binary dataset within the Data Display window for plotting, the abbreviated name of each set of this data follows one of these naming conventions. The dot separation mark is used to concatenate a partial name into a unique data name.

The abbreviated name of data collected by a sink component that is not part of any hierarchical component in the schematics is:

<sink instance name>

or

<top level schematic design name>..<sink instance name>

The double dot separation mark is used to abbreviate the entire hidden or implicit partial names between the two explicit partial names.

The abbreviated name of data collected by a sink component that is nested one design level below the top level design of a hierarchical schematic design is:

<nested hierarchical design instance name>.<sink instance name>

The naming convention for the full name of the data collected by a sink component that is part of an N-level hierarchical design in the schematics is:

<1st nested design instance name>.<2nd nested design instance name>.....
<2nd from last nested instance name>.<last nested design instance name>.
<sink instance name>

Automatically Opening Data Display after Simulation

All sinks (except *Printer* (sinks), *TimedDataWrite* (sinks), and OutFile) that generate a binary dataset have a Plot parameter with Rectangular and None options. This variable, in conjunction with the Open Data Display when simulation completes setting in the Simulate

> Simulation Setup dialog box, determines whether the data collected by the sink is plotted automatically in a Data Display window.

To automatically plot sink data at the end of simulation, before the start of simulation, from the Schematic window go to *Menu > Simulate > Simulation Setup* and set *Open Data Display when simulation completes*. Then, for each sink that you want data to automatically plot, set Plot = Rectangular. At completion of the simulation, a Data Display window automatically displays your data with rectangular plots. In this window you can set the plotting format, and the plot is updated in the same format after the next simulation.

Sinks and Optimization

When optimizing a signal processing schematic design, place a Goal component in conjunction with the Optim controller.

The RangeVar parameter in the Goal component can be used to specify the range variable to be used during optimization; refer to the following table when it is necessary to specify RangeVar. Note that optimization proceeds successfully if RangeVar is not specified.

Sinks and Their Independent Variables

Sink	Conditions	Independent Variable
berMC, berMC4	berOutput = ber vs time	Index
	berOutput = ber vs time	time
	berOutput = ber only	Index
	berOutput = ber only	currentTime
berIS		
EVM		Index
NumericSink		Index
Sinad		xIndex
SpectrumAnalyzer		freq
SpectrumAnalyzerResBW		freq
StatEye		time(UI)
TimedSink		time
FFTAnalyzer, SpecAnalyzer	DisplayFreqUnit = Hz	Hertz
	DisplayFreqUnit = kHz	Kilo_Hertz
	DisplayFreqUnit = MHz	Mega_Hertz
	DisplayFreqUnit = GHz	Giga_Hertz
ErrVecMeas		xIndex

Access to Timed Signal Carrier Frequency in a Dataset

From the Data Display window, the value for carrier frequency *fc* for a timed signal is available by use of an expression. To use this dataset value, place an equation (*Insert* > *Equation*). In the dialog box, write an equation with the name on the left side (*MyFc*), and on the right side using the *get_attr( )* function. For example,

MyEqn = get_attr(<*sink data name*>, "fc")

where <*sink data name*> is the name of the sink data selected from the list shown in the equation dialog box. This *MyFc* equation can then be used in a plot, table, or other equations. For details on naming a dataset, refer to the section on Sink Data in Binary Dataset.

PE Estimator Usage

This section explains the use of the probability of error (PE) measurement components berMC, berMC4, and berIS in the Signal Processing schematic. With these components, you can measure not only the PE performance of the system but the signal-to-noise ratio of the system as well.

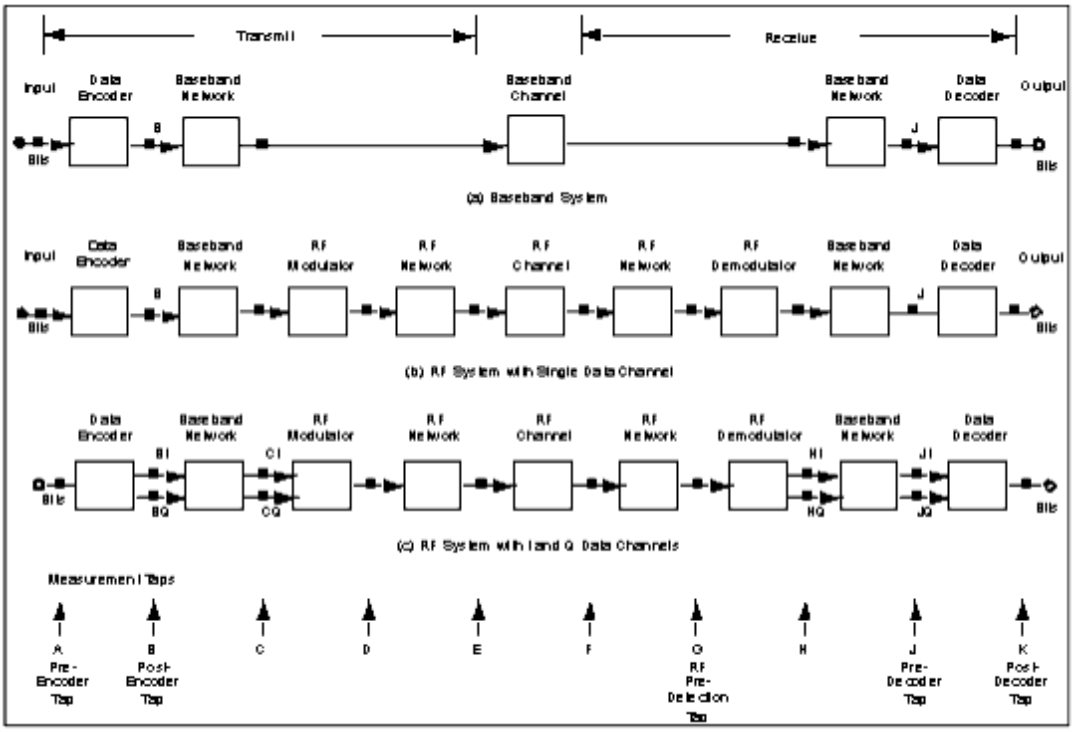
The basic system models and the PE simulation concepts and methodology are explained.

Typical Baseband and RF System Models

Communication systems can be classified broadly as baseband or RF systems. Typically, baseband systems use the PAM data format although other formats, such as pulse width modulation and pulse position modulation, can be used also. RF systems use a wide variety of modulation schemes, such as QAM, PSK schemes (QPSK, DQPSK, PI4DQPSK), and others, such as MSK and FSK. The primary interest is to measure the probability-of-symbol error (Ps) or bit-error rate (BER) of the system.

In the following figure, a general model of a baseband system link is represented in part (a) and general RF system models are represented in parts (b) and (c).

Typical Baseband and RF Systems



An RF system consists of the input bit stream, Transmit Data Encoder, Transmit Baseband

Network, RF Modulator, Transmit RF Network, RF Channel, Receive RF Network, RF Demodulator, Receive Baseband Network, Receive Data Decoder, and the output bit stream. The RF system can have a single data channel, or can have I and Q data channels. The Baseband system omits the RF sections. However, the following discussion applies to both systems.

Baseband System Model

The bit stream is often in the NRZ (non-return-to-zero) data format. The Transmit Data Encoder is used for different purposes such as to convert the NRZ data format to another format: for example, multi-level PAM (pulse amplitude modulation) symbol format, or error control coding.

The Baseband Channel has a transmit and receive side and is typically a bandwidth limited environment with intersymbol interference and noise introduced in the channel. The Receive Data Decoder is used to convert the received symbols to the desired output binary data stream.

RF System Model

Often, the bit stream for this system is also in the NRZ data format. The Transmit Data Encoder is used again to convert the NRZ data format to another format or for error control coding. Typically, the Transmit Baseband Network is used to band limit the frequency spectra of the data symbols, and often introduces intersymbol interference to the symbol stream.

The RF modulator can be of many types. Some formats (such as QAM and QPSK) use I and Q data channels while others (such as BPSK) contain a single data channel.

The RF networks and channel include items such as transmit IF and RF filtering, upconverters, high-power amplifiers, coaxial cable, antennas, line-of-sight link, receive RF and IF filters, and receive low-noise amplifier.

The RF Demodulator converts the RF energy back to a baseband symbol stream that includes symbol distortions, interference, and noise. Typically, the Receive Baseband Network is used to filter the received symbol stream to reduce symbol distortion and noise. The Receive Data Decoder is used to convert the received symbols to the desired output bit stream.

PE Measurement Concepts

The probability of error of a system is measured by comparing the output data stream to the input data stream. The BER gives the average number of output bit-errors per input

bit; for example, a 10 BER means that, on the average, 1 output bit-error occurs with 10^6 input bits.

When the data streams being compared are not simple 2-level (binary) data streams, the measurement is with respect to the data symbols and is called the probability of symbol error (Ps) measurement.

Typical hardware Ps/BER measurements are based on the exact number of symbol/bit-errors and the number of symbols/bits transmitted. This is called a Monte Carlo measurement.

For the PE measurement to be statistically significant, the number of bits transmitted should be much greater than 1/PE (as a rule of thumb).

The relative variance of the PE for N transmitted bits is:

$$VAR = (1 - PE) / (PE \times N)$$

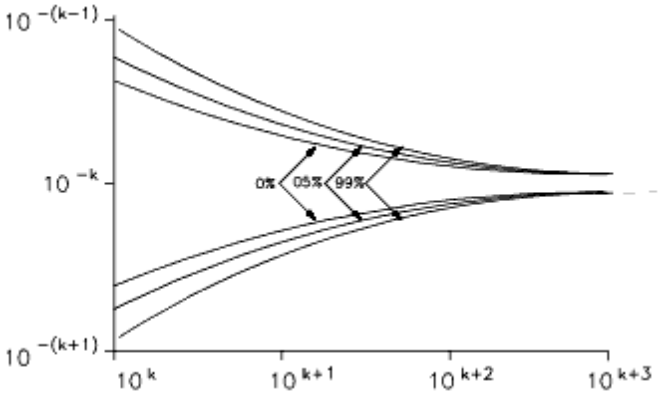
This implies that, for a PE of 10^{-6}, with a relative variance of 0.01, a sample size N of approximately 10^8 bits is required.

The following figure shows the confidence bands on the PE measurement versus total number of bits observed.

For the program PE measurement, the number of samples required is established by setting the relative variance for the PE measurement. As can be seen, for a PE of 1.0×10^{-k} with 100×10^k samples measured, there is a 99% confidence that the actual PE is between 0.77×10^{-k} and 1.3×10^{-k}.

For a low PE uncertainty, a smaller variance is required. However, a smaller variance requires a larger number of transmitted bits.

PE Confidence Bands



Simulation Concepts

Simulation is performed in discrete time steps. At each time step, the signals generated by all the sources are propagated through the system and the outputs are evaluated. Noise can be introduced in the system by means of the random noise sources available. Another method of introducing noise is to use the electrical models of components and to define their noise properties by means of parameters such as noise figure or noise temperature. This is a useful feature that permits you to build an accurate physical model of an actual system. An equivalent noise source is generated automatically that represents the noise properties of the electrical component. No distinction is made between data sources and noise sources during simulation when evaluating the output of the system and the output measured is the net effect due to all sources.

No assumption is made about the nature of the noise at the system output. The statistics of the noise at the output depend on the transformations the noise undergoes when propagating through the system. Thus, the effects of nonlinearities in the system can be simulated accurately.

Measurement Taps

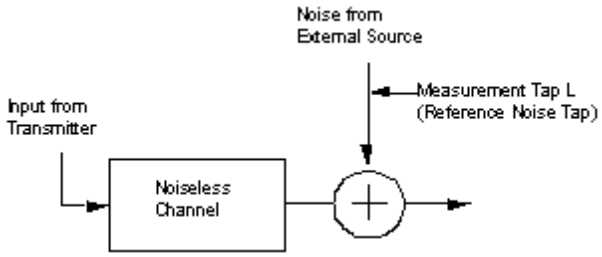
The PE performance of a system is calculated by comparing the transmitted data (also referred to as V_{ref}) with the output data of the receiver (referred to as the V_{test}). V_{ref} can be measured at either the pre-encoder tap or the post encoder tap of the transmitter (see the figure Typical Baseband and RF Systems). Correspondingly, V_{test} should be measured at the post-decoder tap or the pre-decoder tap at the receiver. The choice depends on the system under consideration. For example, if the data encoder consists of an NRZ to 4-level PAM data converter, the post-encoder and pre-decoder taps are convenient measurement taps for the reference and test signals, respectively. However the NRZ data bits can be encoded first with an error correction encoder prior to converting these to a 4-level PAM format. Then the data decoder would consist of a 4-level PAM to binary converter followed by an error correction decoder and the PE performance of the entire system could be measured by comparing the pre-encoder signal to the post-decoder signal.

The signal-to-noise ratio of the system cannot be measured by monitoring the receiver output because the output is the combined effect of the data signals and the noise. Therefore, the signal and noise statistics cannot be measured separately. You must either calculate the SNR by examining the structure of the system and the statistical properties of the noise sources, or make other measurements (depending on the system) to determine the SNR.

However, you can introduce noise in the system by placing an external noise source. Typically, such a noise source would be introduced in the channel. If this external noise source is the predominant source of noise in the system, the SNR can be calculated by measuring the power of the noise source and the power of the data signal separately.

Then an additional measurement tap (labelled L and called the Reference noise tap) is required as shown in the following figure (also see the figure Typical Baseband and RF Systems).

Introduction of Noise from an External Source into a System



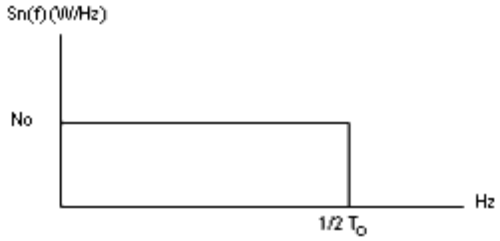
In the preceding figure, the *noiseless channel* block represents effects of the channel such as attenuation of the signal, propagation delay, distortion, and fading. The noise signal measured at the Reference noise tap is called E_{Nref} and the data signal that is used to measure the signal energy is called E_{Sref} . E_{Sref} can be measured at any tap in the system prior to the point where the noise is introduced.

Noise Sources

The nature of the noise introduced in the channel depends on whether the system is a baseband or RF system. Typically, external noise is generated by using the Noise source that is a baseband or RF noise source. The output of the Noise source can be directly added to the signal in the channel.

Another important aspect is the spectral characteristic of the Noise source (see the following figure). Output of the Noise source is a bandlimited white noise signal whose bandwidth is dependent on the time step at which the simulation is carried out. The time interval between two consecutive noise samples T_o , is determined by the parameter TStep in the Noise source. The (baseband) bandwidth of the noise is equal to 1/(2 × TStep) and the power spectral density is constant over this bandwidth.

Spectral Characteristic of the Noise Source



Let

σ = RMS value of the output of the Noise source in volts
T_o = simulation time step in seconds

R_{ref} = default reference resistance is 50 Ohms

S_n (f) = one-sided power spectral density of the bandlimited white noise in W/Hz

Then

S_n (f) = N_o (W/Hz) for 0 ≤ f ≤ $\frac{1}{2T_o}$ (Hz)

where

$N_o = \frac{2\sigma^2 T_o}{R_{ref}}$

Therefore, the auto-correlation function of the noise is given by

$R_n(\tau) = \frac{N_o}{2T_o}\mathrm{sinc}\left(\frac{\tau}{T_o}\right)$

Because the noise is sampled every T_o seconds, the correlation between the noise samples is given by R_n (k T_o), where k is an integer; therefore, the correlation between the noise samples is zero. In the case of a Gaussian noise source, the samples are also independent.

Relationship between SNR, Es/No, and Eb/No

The signal-to-noise ratio of a system can be calculated in different ways. The ratio of the signal power to the noise power is one such measure and is denoted as the SNR of the system. A measure that is used more commonly is the ratio of the energy per symbol to the power spectral density of the noise (Es/No), or the ratio of the energy per bit to the

power spectral density of the noise (Eb/No).

The berMC4 component measures the power in E_{Sref} and E_{Nref} from which the desired signal-to-noise ratio is calculated as follows.

Define the following:

P_{ESref} = power in the E_{Sref} signal in Watts
P_{ENref} = power in the E_{Nref} signal in Watts
T_o = simulation time step in seconds
T_s = symbol time in seconds

Es/No is calculated according to the equations:

$$Es = P_{ESref} \times T_s$$

If the noise signal is in a baseband representation, then

$$No = P_{ENref} \times (2 \times T_o) \text{ and } \frac{Es}{No} = \frac{P_{ESref} \times T_s}{P_{ENref} \times (2 \times T_o)}$$

If the noise signal is in a complex envelope representation, then

$$No = P_{ENref} \times T_o \text{ and } \frac{Es}{No} = \frac{P_{ESref} \times T_s}{P_{ENref} \times T_o}$$

To convert Es/No to Eb/No, let each symbol carry L bits of information. Then Eb/No is simply given by Eb/No = (Es/No)/L.
SNR is simply given by SNR = P_{ESref} / P_{ENref}.

Error Detection

Error detection is performed by sampling V_{ref} and V_{test} every T_s seconds (here T_s is the symbol time) and comparing the samples. If the samples do not lie within the same threshold levels, an error is declared. The ratio of the number of errors counted to the total number of bits transmitted is the estimated PE.

Setting Up BER Simulations

Three important points must be considered when setting up a BER measurement:

- Synchronizing test (V_{test}) and reference (V_{ref}) signals

- Choosing the optimal sampling instant
- Scaling V_{test} and V_{ref} appropriately

Each point is discussed in the following sections. For these discussions, *BER sink* refers to any berMC, berMC4, or berIS component.
BER Simulation Examples provides information on how to access designs that demonstrate the concepts described in the following sections.

Synchronizing Test and Reference Signals

A successful BER simulation requires that Vtest and Vref are synchronized. Otherwise, the BER measurement result are most likely to be close to 0.5 (50%). Vtest and Vref can be synchronized in one of two ways: Manual Synchronization and Automatic Synchronization.

- **Manual Synchronization**
  If the exact delay between V_{test} and V_{ref} is known, synchronization can be achieved easily by introducing the same amount of delay in V_{ref} using the DelayRF component.

  In this case, set the DelayBound parameter of the BER sink to −1 to turn off the auto synchronization feature.
  The exact delay between V_{test} and V_{ref} can be found in several ways:
  - Sometimes the delay introduced by each component in the path between V_{test} and V_{ref} is known. For example, the delay introduced by the LPF_RaisedCosineTimed component is known since delay is one of the parameters set. In this case, adding the delays introduced by each component gives the exact delay between V_{test} and V_{ref}.
  - V_{test} and V_{ref} can be recorded (using TimedSink components) and plotted in the data display. Often, by observing the plots, the delay between the two signals can be determined.
  - The CrossCorr component can be used to measure the delay between Vtest and V_{ref} (see the *MeasuringDelay* design in example workspace: *File* > *Open* > *Example* > *PtolemyDocExamples* > *BER_Validation_wrk*).

  When determining the delay by observing the plots of V_{test} and V_{ref} versus time or by using the CrossCorr component, turn off noise and other impairments in the system (set power level of noise sources very low and make amplifiers linear). When these impairments are turned off, the V_{test} waveform is close to the ideal waveform, which helps determine the delay more easily and accurately.

- **Automatic Synchronization**
  If the exact delay between V_{test} and V_{ref} is unknown, the auto synchronization

feature of the BER sink can be used. In this case, the DelayBound parameter of the BER sink must be set to a value that is an upper bound of the exact delay between the two signals. The BER sink then cross-correlates the two signals and tries to estimate the delay between them. Since the auto synchronization feature relies on the cross correlation to estimate the delay, the BER sink may not be able to always synchronize the two signals, especially at low signal-to-noise ratios.
An upper bound of the delay between V_{test} and V_{ref} can be found using any of the three ways described above used to find the exact delay.

- Knowing the upper bound of the delay introduced by each of the components in the path between V_{test} and V_{ref} and adding these upper bounds.
- Observing the plots of V_{test} and V_{ref} versus time.
- Using the CrossCorr component to get an initial estimate of the delay and adding a few simulation time steps to it.

When the auto synchronization feature is used, no delay needs to be added to the reference signal. However, if the upper bound of the delay is large (greater than 50 symbol periods) and if a lower bound of the delay (DL) between V_{test} and V_{ref} is known, then we recommend that V_{ref} is delayed by this amount (DL) using a DelayRF component and DelayBound is reduced by the same amount (DL). This setting reduces the memory used by the BER sink and speeds up the synchronization process. For example, assume that the delay introduced by the transmitter and receiver filters is 20 µsec and the delay introduced by the rest of the components (e.g. RF channel) has an upper bound of 35 µsec. One way to set up the BER simulation is to not delay the reference signal at all and set the DelayBound parameter of the BER sink to 55 µsec. Another (recommended) way is to delay the reference signal by 20 µsec and set DelayBound to 35 µsec.

When the auto synchronization feature is used, the BER sink saves the value of the estimated delay in the simulation dataset. The name of the dataset variable is *<sink instance name>.Delay*.

Choosing the Optimal Sampling Instant

The BER sink downsamples V_{test} and V_{ref} to one sample per symbol before it compares them in order to detect errors. The first sample of V_{ref} is taken at the time instant specified by the Start parameter. If auto synchronization is turned off (DelayBound = −1), then the first sample of V_{test} is also taken at Start. If auto synchronization is turned on, then the first sample of V_{test} is taken at Start+Delay, where Delay is the delay the BER sink estimated.

The optimal sampling instant is the instant where there is no ISI (intersymbol interference) or where the minimum ISI occurs. For systems, using root raised cosine filters at the transmitter and receiver, the optimal sampling instant is at the center of the symbol period. Therefore, set the Start parameter of the BER sink as follows:

- if no delay is introduced in the reference signal, set Start to
  N × SymbolTime + int((SampPerSym − 1) / 2) × TStep
  where N is a positive integer, SymbolTime is the symbol period, SampPerSym is the number of samples per symbol in V_{test} and V_{ref} , and TStep is the simulation time step for V_{test} and V_{ref}.
- if a delay of D is introduced in the reference signal, set Start to
  N × SymbolTime + int((SampPerSym − 1) / 2) × TStep + D

If your system has differential encoding/decoding, synchronization loops, carrier recovery sub-systems, or other sub-systems that need time to reach their steady state, set N to a value that is large enough to enable the entire system to reach steady state.

When a square root raised cosine filter with pulse equalization is used in the modulator (this filter is used inside the DBPSK_Mod, QPSK_Mod, DQPSK_Mod, DQOPSK_Pi4Mod components) and the input signal to the modulator is an NRZ waveform (piece-wise constant waveform for multi-level PAM and QAM systems) with an even number of samples per symbol, then the signal at the output of the modulator and eventually at the output of the demodulator does not have a sample at the optimal sampling instant. The optimal sampling instant occurs at the midpoints in between the existing signal samples. This is a limitation that results from the need to represent continuous time signals with samples. If there is no signal sample at the optimal sampling instant, then the BER sink samples V_{test} at a point where there is ISI and the BER results are worse than expected.

To overcome this limitation, the Interpolator component can be connected to the output of the modulator. When the TimeRef parameter of the Interpolator is set to TStep / 2 and a fourth order Lagrange interpolation is used, the Interpolator reconstructs the signal samples that occur at the midpoints between the existing samples.

Scaling of Test and Reference Signals

For signals with only two levels, such as 2-level PAM, BPSK, DBPSK, QPSK (2 levels per axis), DQPSK (2 levels per axis), and zero mean value, the threshold for deciding whether an error has occurred or not is 0 (if both V_{test} and V_{ref} are positive or both are negative no error is detected; if one is positive and the other one is negative an error is detected). In this case, the exact level of V_{test} and V_{ref} is not important and does not affect the BER measurement result.

For signals with more than two levels, such as multi-level PAM (4-PAM, 8-PAM, 16-PAM, ...) or QAM (16-QAM, 32-QAM, 64-QAM), the exact level of V_{test} and V_{ref} is important because there is more than one error detection threshold.

If the automatic threshold setting is used (BER sink ThresholdSetting parameter set to *automatically*) then the error detection thresholds are set to (2 × i − N) / N, i = 1, 2, ..., N, where N is the number of thresholds (BER sink NumThreshold parameter). NumThreshold must be set to the number of signal levels minus 1 (3 for 4-PAM, 7 for 8-

PAM, 3 for 16-QAM (3 levels per axis), 7 for 64-QAM (7 levels per axis)). The expected signal levels are located at the midpoints between the thresholds, that is at $(2 \times i - 2 - N) / N$, $i = 1, 2, \ldots, N + 1$. Both V_{test} and V_{ref} must be scaled appropriately so that

when these are sampled at the optimal sampling instant and assuming ideal conditions (no noise or other distortions) these generate samples at the expected levels. Turning off the noise and the rest of the impairments in the system and plotting the eye diagrams for Vtest and Vref can help determine the signal levels at the optimal sampling instant and therefore, the appropriate scale factors.

Note
When the berIS component is used, the V_{ref} and V_{test} signal levels must be the same (including 2-level PAM, BPSK, DBPSK, QPSK, and DQPSK signals). This is because the amount of noise added to V_{test} inside the berIS component is dependent on signal levels at the berIS inputs.

BER Simulation Examples

The *BER_Validation_wrk* example workspace (*File > Open > Example > PtolemyDocExamples > BER_Validation_wrk*) includes several example designs that demonstrate the concepts discussed in the previous sections. These designs show how the theoretical BER performance in an AWGN (additive white gaussian noise) environment can be achieved for various modulation types such as BPSK, DBPSK, QPSK, DQPSK, π/4-DQPSK, 16-QAM, 64-QAM. Each example design provides setup information.

References

1. J. Baprawski, "Generalized Modulation Mathematics Applied to RF System Simulation," *Proceedings of the RF Expo West Conference*, Santa Clara, CA, February 5-7, 1991, pp. 127-147.
2. T. T. Ha, *Digital Satellite Communications*, McGraw-Hill, 1990.
3. N. Kanaglekar, et al. "Wave Analysis of Noise in Interconnected Multiport Networks," *IEEE Transactions on Microwave Theory and Techniques*, Vol. MTT-35, No. 2, February 1987, pp. 112-115.
4. D. Lu and J. Baprawski, "The Quasi-Analytical Method for Estimating Error Probabilities of M-ary Signaling Systems with Intersymbol Interference," *Proceedings of the 13th Symposium on Information Theory and its Applications* , Tateshina, Japan, January 23-25, 1991, pp. 109-114.
5. D. Lu and K. Yao, "Improved Importance Sampling Technique for Efficient Simulation of Digital Communication," *IEEE Journal on Selected Areas in Communication*, J-SAC, Vol. 6, No. 1, January 1988, pp. 67-75.
6. R. W. Lucky, J. Salz and E. J. Weldon, Jr., *Principles of Data Communications*, McGraw-Hill, 1968.
7. P. Z. Peebles, Jr., *Digital Communication Systems*, Prentice-Hall, 1987.
8. G. Proakis, *Digital Communication*, McGraw-Hill, 1989.
9. K. S. Shanmugan, *Digital and Analog Communication Systems*, John Wiley & Sons, 1985.

10. B. Sklar, *Digital Communications*, Prentice-Hall, 1988.

SpectrumAnalyzerResBW

Description: Spectrum analyzer with resolution bandwidth setting
Library: Sinks
Class: TSDF_SpectrumAnalyzerResBW
Derived From: _SpectrumAnalyzer

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
Window	Window with default constant applied to collected data (default constant is used when WindowConstant is 0.0): none, Hamming 0.54, Hanning 0.50, Gaussian 0.75, Kaiser 7.865, _8510 6.0, Blackman, Blackman-Harris	none		enum	
ResBW	Resolution bandwidth	30 kHz	Hz	real	[0, ∞)
NumSegments	Number of segments	0		int	[0, ∞)
SegmentTime	Segment time	1.0 msec	sec	real	(0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

1. The SpectrumAnalyzerResBW component can be used to measure the spectrum of a baseband or an RF signal with a user specified resolution bandwidth. In the following notes, TStep is used to denote the simulation time step and fc is used to denote the signal characterization frequency (fc = 0 for a baseband signal and fc > 0 for an RF signal).

Note
Information regarding time domain signal differences between ADS Ptolemy simulations and Circuit Envelope and Transient simulations is given in the *Timed Synchronous Dataflow* (ptolemy) section of the *ADS Ptolemy Simulation* (ptolemy) documentation.

2. The component outputs the complex amplitude voltage values at the frequencies of the spectral tones. The component does not output the power at the frequencies of the spectral tones. However, you can calculate and display the power spectrum as well as the magnitude and phase spectrum by using the dBm, mag, and phase functions of the Data Display window. Note that the dBm function assumes a 50-ohm resistive load. If a different load was used in the simulation, its value can be specified as a second argument to the dBm function, for example, dBm(VRF, 75) where VRF is the instance name of the SpectrumAnalyzerResBW component and 75 ohms is the resistive load used. To integrate the power spectrum over a frequency range see note 7.
   Note that, for baseband signals and for the frequency of 0 Hz, the dBm function returns a power value that is 3 dB less than the actual power. This is because the primary use of the dBm function is with RF signals from an Analog/RF schematic simulation, where 0 Hz corresponds to the characterization frequency and not 0 Hz signal frequency (DC part of the signal).
   If the baseband signal has no significant power at dc, this 3 dB error is insignificant and can be ignored-otherwise, it must be considered. For a baseband signal, the energy at 0 Hz is more typically measured as the average voltage value of the signal and not by power.
3. The displayed spectrum extends from 0 Hz to $1/(2 \times TStep)$ Hz for a baseband signal and from $fc - 1/(2 \times TStep)$ Hz to $fc + 1/(2 \times TStep)$ Hz for an RF signal. When $fc < 1/(2 \times TStep)$, the default spectrum extends to negative frequencies. The spectral content at these negative frequencies is conjugated, mirrored, and added to the spectral content of the closest positive frequency. This way, the negative frequency tones are displayed on the positive frequency axis as would happen in an RF spectrum analyzer measurement instrument. This process can introduce an error in the displayed frequency for the mirrored tones. The absolute error introduced is less than the frequency step in the displayed spectrum (see note 5 for the more details about frequency step in the displayed spectrum).
4. The basis of the algorithm used by SpectrumAnalyzerResBW is the fs() function (the chirp-Z transform option of fs() is used). Depending on the values of the parameters Start, ResBW, and NumSegments, the algorithm can call fs() on multiple signal segments and average the results to achieve video averaging (see note 5 for more details).
   For details regarding the *fs()* function, refer to to *Measurement Expressions* (expmeas).

5. Depending on the values of the parameters ResBW and NumSegments the spectrum measurement is performed in a different way. The four modes of operation are described:
   - ResBW > 0, NumSegments = 0
     This is the default mode of operation. In this mode, the input signal (in the time interval [Start, Stop]) is broken down in multiple segments of length NENBW / ResBW, where NENBW is the normalized equivalent noise bandwidth for the selected window (see note 8 for the definition of NENBW). The spectra of the individual segments are calculated and averaged (video averaging). If the

interval [Start, Stop] is not long enough, that is Stop − Start < NENBW / ResBW, then Stop is increased so that one segment of length NENBW / ResBW is processed. If Stop − Start > NENBW / ResBW but the interval [Start, Stop] does not contain an integer multiple of NENBW / ResBW long segments, Stop is again adjusted (increased) so that an integer number of NENBW / ResBW long segments is processed.
The resolution bandwidth achieved is ResBW. The frequency step in the displayed spectrum is also ResBW.

- ResBW > 0, NumSegments > 0
  This mode of operation is very similar to the first one. The only difference is that the value of the Stop parameter is ignored and Stop is set to Start + NumSegments × SegmentTime. Stop can then be further adjusted as explained in the first mode of operation.
- ResBW = 0, NumSegments = 0
  In this mode of operation, data is collected from Start to Stop and the spectrum analysis is performed on the whole data treated as one segment. This achieves the highest possible resolution bandwidth for the collected data given by NENBW / (Stop − Start), where NENBW is the normalized equivalent noise bandwidth for the selected window (see note 8 for the definition of NENBW). The frequency step in the displayed spectrum is 1 / (Stop − Start). No video averaging occurs in this mode of operation.
- ResBW = 0, NumSegments > 0
  In this mode of operation, data is collected from Start to Start + NumSegments × SegmentTime. The data is broken down in NumSegments segments of length SegmentTime. The spectra of the individual segments are calculated and averaged (video averaging). The resolution bandwidth achieved is NENBW / SegmentTime, where NENBW is the normalized equivalent noise bandwidth for the selected window (see note 8 for the definition of NENBW). The frequency step in the displayed spectrum is 1 / SegmentTime. Video averaging occurs if NumSegments > 1.

6. The Window parameter is used to define the window that is applied to each segment before its spectrum is calculated. Windowing is often necessary in transform-based (chirp-Z, FFT) spectrum estimation. Without windowing, the estimated spectrum can suffer from spectral leakage that can cause misleading measurements or masking of weak signal spectral detail by spurious artifacts.
   Window Definitions
   - none

     $$w(kT_s) = \begin{cases} 1.0 & 0 \le k \le N \\ 0.0 & otherwise \end{cases}$$

     where N is the window size
   - Hamming 0.54

     $$w(kT_s) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi k}{N}\right) & 0 \le k \le N \\ 0.0 & otherwise \end{cases}$$

     where N is the window size
   - Hanning 0.50

$$w(kT_s) = \begin{cases} 0.5 - 0.5\cos\left(\frac{2\pi k}{N}\right) & 0 \le k \le N \\ 0.0 & otherwise \end{cases}$$

where N is the window size

- Gaussian 0.75

$$w(kT_s) = \begin{cases} exp\left(-\left(0.75\pi\left(\frac{2k-N}{N}\right)\right)^2\right) & 0 \le k \le N \\ 0 & otherwise \end{cases}$$

where N is the window size

- Kaiser 7.865

$$w(kT_s) = \begin{cases} \frac{I_0\left(7.865\left[1-\left(\frac{k-\alpha}{\alpha}\right)^2\right]^{1/2}\right)}{I_0(7.865)} & 0 \le k \le N \\ 0.0 & otherwise \end{cases}$$

where N is the window size, α = N / 2, and I_0 (.) is the 0th order modified Bessel function of the first kind

- 8510 6.0 (Kaiser 6.0)

$$w(kT_s) = \begin{cases} \frac{I_0\left(6.0\left[1-\left(\frac{k-\alpha}{\alpha}\right)^2\right]^{1/2}\right)}{I_0(6.0)} & 0 \le k \le N \\ 0.0 & otherwise \end{cases}$$

where N is the window size, α = N / 2, and I_0 (.) is the 0th order modified Bessel function of the first kind

- Blackman

$$w(kT_s) = \begin{cases} 0.42 - 0.5\cos\left(\frac{2\pi k}{N}\right) + 0.08\cos\left(\frac{4\pi k}{N}\right) & 0 \le k \le N \\ 0.0 & otherwise \end{cases}$$

where N is the window size

- Blackman-Harris

$$w(kT_s) = \begin{cases} 0.35875 - 0.48829\cos\left(\frac{2\pi k}{N}\right) + 0.14128\cos\left(\frac{4\pi k}{N}\right) - 0.01168\cos\left(\frac{6\pi k}{N}\right) & 0 \le k \le N \\ 0.0 & otherwise \end{cases}$$

where N is the window size

7. The windowing of a signal in time has the effect of changing its power. SpectrumAnalyzerResBW normalizes the measured spectrum so that the power contained in it is the same as the power of the input signal.
   To calculate the power contained in a spectrum, the spec_power() function can be used in the Data Display window. The *spec_power()* function expects its input to be

in dBm and returns a value in dBm. Therefore, if spectral data generated using SpectrumAnalyzerResBW is passed to the spec_power() function, the dBm function must be applied to the data before spec_power() is called. If frequency limits need to be passed to spec_power(), these must be specified in Hz.
For details regarding the *spec_power()* function, refer to *Measurement Expressions* (expmeas).

8. The windowing of a signal in time also affects the resolution bandwidth that can be achieved. When calculating the spectrum of a signal segment the resolution bandwidth achieved with a window is always lower (worse) than the resolution bandwidth achieved without a window. The normalized equivalent noise bandwidth (NENBW) of a window is one measure of how much it reduces the resolution bandwidth that can be achieved. To compensate for this a longer signal segment should be processed.
   Equivalent noise bandwidth (ENBW) compares the window to an ideal, rectangular filter. It is the equivalent width of a rectangular filter that passes the same amount of white noise as the window. The normalized ENBW (NENBW) is the ENBW multiplied by the time duration of the of the signal being windowed. NENBW values for windows available in SpectrumAnalyzerResBW are listed in the following table.

Normalized Equivalent Noise Bandwidth of Available Windows

Window	NENBW
none	1
Hamming 0.54	1.363
Hanning 0.50	1.5
Gaussian 0.75	1.883
Kaiser 7.865	1.653
8510 6.0	1.467
Blackman	1.727
Blackman-Harris	2.021

   In an analog swept spectrum analyzer, the resolution bandwidth is determined by the last in a series of analog IF filters. In contrast, SpectrumAnalyzerResBW calculates the spectrum using DSP algorithms and so the resolution bandwidth is determined by the length of the input data segment the algorithm processes and the selected window.
   Without a window (Window=none) the resolution bandwidth achieved is 1/T, where T is the length of the input data segment in seconds.
   With a window the resolution bandwidth achieved is ENBW = NENBW/T, where ENBW is the window equivalent noise bandwidth and NENBW is the window normalized equivalent noise bandwidth.
9. How to choose the right window.
   Every time a window is applied to a signal (Window = none effectively applies a rectangular window to the signal), leakage occurs, that is, power from one spectral component leaks into the adjacent ones. Leakage from strong spectral components can result in hiding/masking of nearby weaker spectral components. Even strong spectral components can be affected by leakage. For example, two strong spectral components close to each other can appear as one due to leakage.

Choosing the right window for a spectral measurement is very important. The choice of window depends on what is being measured and what the trade-offs between frequency resolution (ability to distinguish spectral components of comparable strength that are close to each other) and dynamic range (ability to measure signals with spectral components of widely varying strengths and distributed over a wide range) are.

As described above, windows can be characterized by their Normalized Equivalent Noise Bandwidth (NENBW). In general, for the same length of signal processed, the higher the NENBW of a window the higher its dynamic range (less leakage) and the poorer its frequency resolution.

Some general guidelines for choosing a window are given below:

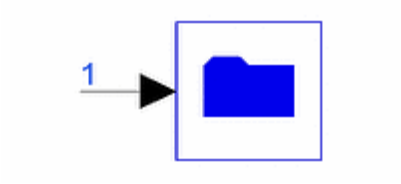
- Do not use a window (Window = none) when analyzing transients.
- For periodic signals whose spectral components have comparable strengths and when the signal segment processed includes an exact integer multiple of periods, the best results are obtained if no window is used (Window = none, which is equivalent to using a rectangular window). Any start up transients should be excluded.
- For periodic signals whose spectral components have significantly different strengths and/or when the signal segment processed does not include an exact integer multiple of periods, the use of a window can improve the detection of the weaker spectral components. The higher the NENBW the more likely the weaker spectral components will be detected. However, this trades-off frequency resolution and so if the spectral components are very close to each other the weaker one might remain unresolved. To improve frequency resolution while still maintaining a good dynamic range use a window but process a longer signal segment.
- For aperiodic signals such as modulated signals (QPSK, QAM, GSM, EDGE, CDMA, OFDM) the use of a window is highly recommended. The window will attenuate the signal at both ends of the signal segment processed to zero. This makes the signal appear periodic and reduces leakage.

10. For general information regarding sinks, refer to *About Sinks* (sinks).

References

1. A. V. Oppenheim, R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 1989, Chapter 9, section 9.7.

TimedDataWrite



Description: Format Specific Output Data File
Library: Sinks
Class: TSDFTimedDataWrite

Parameters

Name	Description	Default	Unit	Type	Range
Start	start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
RLoad	load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
ControlSimulation	if set to YES, 'Stop' time determines how long the simulation will run: NO, YES	YES		enum	
FileName	filename that you assign before beginning signal analysis to create a data file. filename must begin with a letter and cannot exceed 32 characters; choose format using *.tim*, *.bintim*, *.ascig*, *.sig* extension. During analysis, the file is generated and saved to the data subdirectory under your current workspace (_wrk) directory.			filename	

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

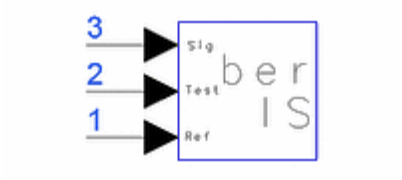
Notes/Equations

1. TimedDataWrite enables the generation of one of the following output files.
   - *.tim* and *.bintim* files use MDIF format (*.tim* in ASCII form, *.bintim* in binary form).
   - *.ascsig* and *.sig* files use a signal file format used in the Cadence Signal Processing Workstation product (*.ascsig* in ASCII form, *.sig* in binary form). For format information, refer to *Understanding File Formats* (ptolemy) in the *ADS Ptolemy Simulation* (ptolemy) documentation.

The file generated by TimedDataWrite can be used in combination with the source TimedDataRead (Timed Sources library) to provide source input data.

2. For general information regarding sinks, refer to *About Sinks* (sinks).

berIS



Description: Error Probability measurement using Improved Importance Sampling
Library: Sinks
Class: TSDF_berIS
Derived From: baseAutoDisplaySink

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRload will inherit from DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resitor physical temperature, in degrees C. DefaultRTemp will inherit from DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for sampling signals	DefaultTimeStart	sec	real	[0, ∞)
SymbolTime	Symbol time	1.0	sec	real	[TStep, ∞)†
NumThreshold	Number of thresholds for the error detection	1		int	[1, ∞)
ThresholdSetting	Type of threshold settings: automatically, manually	automatically		enum	
Threshold	Threshold values when user selects manually setting	0		real array	
DelayBound	Upper bound of delay for Synchronizing inputs; if DelayBound < = 0, the Synchronizer is turned off	-1.0	sec	real	{-1} or (0, ∞)††
berOutput	Type of ber output: ber vs time, ber vs time every 10 symbols, ber vs time every 100 symbols, ber vs time every 1000 symbols, ber only	ber only		enum	
NBw	Noise bandwidth	3.0	Hz	real	(0, ∞)
SystemType	System type: PAM, QAM, QPSK, DQPSK, PI4DQPSK	PAM		enum	
EstVar	Estimation relative variance	0.1		real	(0, 1\]
EsOverNo	Energy per Symbol over Noise Density in dB	3.0		real	(-∞, ∞)
EsOverNoRange	EsOverNo range	0.0		real	[0, ∞)
NumSwpsForEsOverNo	Number of sweep points for EsOverNo	1		int	[1, ∞)

† TStep is the simulation time step for the component input signals.
†† If DelayBound is set to -1, berIS will assume test and reference signals are already synchronized.

Pin Inputs

Pin	Name	Description	Signal Type
1	input1	reference data input	timed
2	input2	test signal input	timed
3	input3	receiving signal input	timed

Notes/Equations

1. Basic Principle
   The berIS probability of error estimation is based on the Improved Importance

Sampling (IIS) method [1 - 3], which can estimate quickly error probabilities for PAM, QAM, QPSK, DQPSK, and PI/4DQPSK systems. To speed simulation, error events are made to occur more frequently by modifying noise density in the IIS. Note that the IIS method is system dependent-berIS cannot be used to estimate error probability for other systems.

Symbol error probability detection is based on comparing modified test data to reference data waveforms, symbol by symbol. When calculating the probability, a weighting function is used to adjust the error probability to an unbiased estimation. To measure symbol error probability:

- Synchronize output test data (test data) with reference data.
- Sample reference and test data. Each sampling time must be at the center of each symbol. The sampling time interval is one symbol time.
- Compare the modified test data sample to the reference data sample.
- Detect an error event by using thresholds. An error occurs if the value of the test sample is not the same as reference sample in the threshold detection; otherwise, there is no error.
- When an error event is detected, calculate the weighting function and add the weight in the error probability buffer, then normalize the error probability by using total number of test samples N. Assume the total number of test data samples (or reference samples) is N, the IIS weighting function is w, and the indicating function for error is I(.), where when an error is detected I(.) = 1, otherwise I(.) = 0. Error probability can be mathematically described by

$$Ps = \frac{1}{N}\Sigma I(i)\ w_i$$

2. The input1 signal is the reference data input and should have no distortion or intersymbol interference. The input2 signal is the one against which the BER measurement is made. The input3 signal is used for setting the gaussian noise level that sets the Eb/No value against which the signal2 is evaluated. Typically, one can just connect the input2 signal to the input3 point.
3. Data Normalization
   To simplify detection, test and reference data need to be normalized from -1 to 1. All detection thresholds are normalized from -1 to 1 also. The thresholds result in symbol ranges equally spaced between −1 and +1 if ThresholdSetting = automatically.
4. Synchronizing Test and Reference Data
   Test and reference data can be synchronized automatically or by the designer:
   Automatic Synchronization
   - Set DelayBound > exact delay between test and reference data. Typically, set DelayBound = 5× to 10× exact delay. This approach does not require knowing the exact delay between test and reference data. Auto-synchronization estimates the delay based on calculating the auto-correlation function between the two waveforms.

     User Synchronization
   - Set DelayBound = −1.
   - Find the exact delay between test data and reference data.
   - Place a Delay component between the reference data source and the reference input pin of the berIS component.
     Continue synchronization with these parameters.
   - Start is used for specifying a sampling start time for reference signal, and also for positioning test and reference samples. As mentioned in note 1, each

sampling time must be at the center of each symbol. For example, for the reference signal from ADS data source, if DelayBound = −1, Start must be set to Delay + Stime/2, where the exact delay between test and reference data is Delay, and symbol time is Stime. Otherwise, Start can be set to Stime/2.

- NumThreshold is determined by signal levels; assume the signal level is M, NumThreshold = M−1.
- ThresholdSetting options:

  automatically setting thresholds for uniform threshold interval.

  manually setting thresholds for uniform or non-uniform interval
- Threshold: Threshold values when you select the *manually* setting
- NBw is the noise bandwidth specified by the designer. In the model the internal noise power is calculated using the EsOverNo and NBw parameters.
- SystemType can be set according to test system type: PAM, QAM, QPSK, DQPSK, or PI4DQPSK.
- EstVar is for controlling estimation accuracy, the simulation relative variance EstVar must be specified properly. A smaller EstVar takes more time and results in a more accurate simulation; a larger EstVar takes less time but the result is not as accurate. When EstVar = 0.01, which is a good estimation, a 10% standard deviation is received.
- EsOverNo is the minimum Es/No in dB. It can be set by the designer. The error probability performance is performed based on the EsOverNo (Es/No) set by the designer. In the model the internal noise power is calculated by using the EsOverNo and NBw parameters.
- EsOverNoRange is the range from minimum Es/No to maximum Es/No in dB.
- NumSwpsForEsOverNo is the number of plotting points for Ps Vs Es/No curve

5. Symbol Error Probability and Bit Error Probability
   berIS measures the symbol error probability Ps for a single channel. For systems with I,Q channels such as QAM, QPSK, DQPSK, and PI/4DQPSK, you must measure both channels to get Psi, Psq. To derive the bit error probability Pei and Peq, use

   $$Pei = Psi/L,\ Peq = Psq/L$$

   where *L* is the number of bits per symbol for I, Q channels.
   Then combine the results for the bit error probability of the system by using

   $$Pe = Pei + Peq - Pei\ Peq$$

   where *Pe* is the bit error probability for the system and *Pei* and *Peq* are the bit error probabilities for each channel.
6. Noise Source
   An external noise source is not required. Noise power is added by berIS in terms of Es/No and the noise bandwidth input from the NBw parameter.
7. Ps Vs Es/No Curve
   To generate a Ps curve, Ps Vs Es/No or BER Vs Es/No, you must have used the ADS sweep controller previously for sweeping values of Es/No. In that case, all signals must be re-generated for each Es/No. For the new version of berIS model, signals for testing BER can be re-used for different Es/No, and the simulation is more efficient. To generate a BER curve properly, the Es/No range can be specified by using

EsOverNo and EsOverNoRange, and the number of sweep points can be set by using NumSwpsForEsOverNo. Because the sweep controller is not used any more to re-generate test signals for different Es/No, the BER curve can be generated faster.

8. IIS Simulation

   For a QAM system at 10 $^{-6}$ of BER and Monte Carlo simulation, 10 ^8 ^ simulation samples are needed for a reasonable accuracy of 0.01 relative variance. However, for the IIS simulation, only 800 samples are needed for accuracy [1].
   An external noise source is not required. Noise power is added by berIS in terms of Es/No and noise bandwidth at input 3.
9. Delay between reference signal and test signal. If set DelayBound > 0, this component provides an estimation of the delay. The delay value can be found in the data display server.
10. For general information regarding sinks, refer to *About Sinks* (sinks).

References

1. D. Lu and K. Yao, "Improved Importance Sampling Technique for Efficient Simulation of Digital Communication Systems," *IEEE J. Select. Areas Commun*. vol. 6, pp. 67-75, Jan. 1988.
2. D. Lu and K. Yao, "Estimation Variance Bounds of Importance Sampling Simulations in Digital Communication Systems," *IEEE Trans. Commun*. vol 39, pp. 1413-1417, Oct. 1991.
3. J. Chen, D. Lu, J. Sadowski, and K. Yao "On Importance Sampling in Digital Communications - Part I: Fundamentals," *IEEE J. Select. Areas in Commun*. vol 11. No. 3, pp. 289-299, April, 1993.