Virtuoso Visualization and Analysis XL User Guide

Product Version 6.1.6
November 2014

Contents

A
Virtuoso Visualization and Analysis XL Tool Environment Variables

B
Calculator Functions 483

Preface

The Virtuoso® Visualization and Analysis XL is an analog and mixed-signal waveform display tool. This user guide describes this tool in detail and explains how you can use the various features of this tool.

What the Virtuoso Visualization and Analysis XL Tool Does

The tool helps you analyze the data generated by your simulator.

The tool consists of the following components:

- Results Browser displays simulation data in the hierarchical arrangement of your design.
- Graph offers features that simplify the processing of your signal data.
- Calculator provides an extensive expression building capability that addresses the needs of a wide variety of analysis types.

Supported Data Formats

The Virtuoso Visualization and Analysis XL can interpret the following data formats.

Format	Description
PSF	Format created by Virtuoso® Spectre Circuit Simulator (including the Virtuoso® Spectre RF Simulation Option) and other simulators integrated into the Virtuoso® Analog Design Environment. From the IC6.1.0 release, the visualization and analysis tool can read files larger than 2G.
SST2	Format created by Virtuoso® AMS and Spectre Verilog Simulators. Spectre and UltraSim can also create this format. This format can also be created by the digital simulators, such as NCsim.

Format	Description
PSF XL	Format created by Virtuoso® Spectre Circuit Simulator (including the Virtuoso® Spectre RF Simulation Option) and other simulators integrated into the Virtuoso® Analog Design Environment. This waveform format is supported by Virtuoso Visualization and Analysis XL tool in IC6.1.3 and later releases. This format provides a very high compression rate for large circuit designs. Starting IC6.1.5 ISR12, PSF XL aliases are also supported in SRR. This alias support is required for Spectre XPS and for unified waveform DB feature driven by AMS Designer.
RTSF	Format created by Virtuoso® Spectre circuit simulator and UltraSim simulators. RTSF is a PSF XL extension that provides improved viewing performance in Virtuoso Visualization and Analysis XL tool. RTSF is a fast waveform format extension that can plot extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds. This format is supported by Virtuoso Visualization and Analysis XL tool in IC6.1.2 and later releases. You can enable this format by using the `+rtsf` option along with PSF.
Touchstone	Format created by all simulator. This is an industry standard format and the Virtuoso Visualization and Analysis XL tool supports this format starting from IC6.1.4 ISR. The touchstone file is an ASCII file, also known as the *SnP* file, which includes a large signals S-parameter results. The touchstone files are of `.snp` extension, where `n` is the number of network ports of the device. For example, if the touchstone file contains the network parameters for a two port device, it has `.s2p` extension.

Note: The Virtuoso Visualization and Analysis XL tool does not support WSF format now.

Licensing Scheme for Virtuoso Visualization and Analysis XL

Following is the licensing scheme for the Virtuoso Visualization and Analysis XL tool:

Virtuoso Visualization and Analysis XL when opened from ADE L, XL, and GXL:

- Shares license tokens with ADE L, XL, or GXL.

- When you close the ADE window, the Virtuoso Visualization and Analysis XL continues to hold the ADE license tokens, which are in effect until all Virtuoso Visualization and Analysis XL windows are closed.

Virtuoso Visualization and Analysis XL when opened in stand-alone mode or from Virtuoso:

- Checks out either the Virtuoso Visualization and Analysis XL license or an ADE license tier, depending on the preferences you have set by using the `VIVA License Checkout Order` .cdsenv variable. By default, this variable is set to `ViVA, ADE`, which results in the following license check out tasks being performed:
  - Checks out the Virtuoso Visualization and Analysis XL license, if available.
  - If the check out operation in the previous step fails, you can choose between checking out an ADE license tier or two ADE GXL tokens, based on the order set in the `ADELicenseCheckoutOrder` .cdsenv variable, which controls the order in which ADE license tiers are checked out.

  If the `VIVALicenseCheckoutOrder` variable is set to `ADE, ViVA`, the license check out tasks are performed in the following order:
  - Checks out an ADE license tier or two ADE GXL tokens, based on the order set in the `ADELicenseCheckoutOrder` .cdsenv variable.
  - If the check out operation in the previous step fails, you can check out the Virtuoso Visualization and Analysis XL license.
- The license is released when all the Virtuoso Visualization and Analysis XL windows are closed.

For more information about licensing in the Virtuoso design environment, see *Virtuoso Software Licensing and Configuration Guide*.

Related Documents

For information about the related products, consult the sources listed below.

- *Virtuoso Analog Design Environment User Guide*
- *Virtuoso Simulator Measurement Description Language User Guide and Reference*

Installation, Environment, and Infrastructure

- For information about installing Cadence products, see *Cadence Installation Guide*.
- For information about Virtuoso Design Environment, see *Virtuoso Design Environment User Guide*.
- For information about the database SKILL functions, including data access functions, see the *Virtuoso Design Environment SKILL Reference*.
- For information about the library structure, library definitions file, and name mapping for data shared by multiple Cadence tools, see the *Cadence Application Infrastructure User Guide*.

Technology Information

- For information about how to create and maintain a technology file and display resource file, see the *Virtuoso Technology Data User Guide* and the *Virtuoso Technology Data ASCII Files Reference*.
- For information about how to access the technology file using SKILL functions, see the *Virtuoso Technology Data SKILL Reference*.

Virtuoso Tools

- For information about how to perform design tasks with the Virtuoso Layout Suite L layout editor, see the *Virtuoso Layout Suite L User Guide*
- For information on design rule driven editing, see the *Virtuoso Design Rule Driven Editing User Guide*.
- For information about how to use the Virtuoso Layout Suite wire editing capability, see “Interactive Wire Editing” in the *Virtuoso Space-based Router User Guide*.
- For information about how to use the automatic custom digital placer to place your design components, see the *Virtuoso Custom Digital Placer User Guide*.
- For information about creating parameterized cells using the graphical user interface or low-level SKILL functions, see the *Virtuoso Parameterized Cell Reference*.
- For information about Component Description Format, see the *Component Description Format User Guide*.
- For information about how to stream mask data, see the *Design Data Translator’s Reference*.

- For information about custom layout SKILL functions, see the *Virtuoso Layout Suite SKILL Reference*.

Additional Learning Resources

Cadence provides various Rapid Adoption Kits that you can use to learn how to employ Virtuoso applications in your design flows. These kits contain workshop databases, designs, and instructions to run the design flow.

Cadence offers the following training course on the Virtuoso Visualization and Analysis XL flow:

- Virtuoso Analog Design Environment
- Virtuoso Schematic Editor
- Analog Modeling with Verilog-A
- Behavioral Modeling with Verilog-AMS
- Real Modeling with Verilog-AMS
- Spectre Simulations Using Virtuoso ADE
- Virtuoso UltraSim Full-Chip Simulator
- Virtuoso Simulation for Advanced Nodes

For further information on the training courses available in your region, visit the Cadence Training portal. You can also write to training_enroll@cadence.com.

The links in this section open in a new browser. The course links initially display the requested training information for North America, but if required, you can navigate to the courses available in other regions.

Third Party Tools <*Optional*>

To view any `.swf` multimedia files, you need:

- Flash-enabled web browser, for example, Internet Explorer 5.0 or later, Netscape 6.0 or later, or Mozilla Firefox 1.6 or later. Alternatively, you can download Flash Player (version 6.0 or later) directly from the Adobe website.
- Speakers and a sound card installed on your computer for videos with audio.

Typographic and Syntax Conventions

Special typographical conventions distinguish certain kinds of text in this document.

- Boldface words represent elements of the syntax that must be used exactly as presented. Such items include keywords, operators, and punctuation marks. For example,

  statefile

- Variables are set in italic font,

  maxDirectories

- Vertical bars indicate alternatives. You can choose to use any one of the items separated by the bars. For example,

  -graphattributesfile "*mygraph***"**
  |**-readstatefile "true | false"**

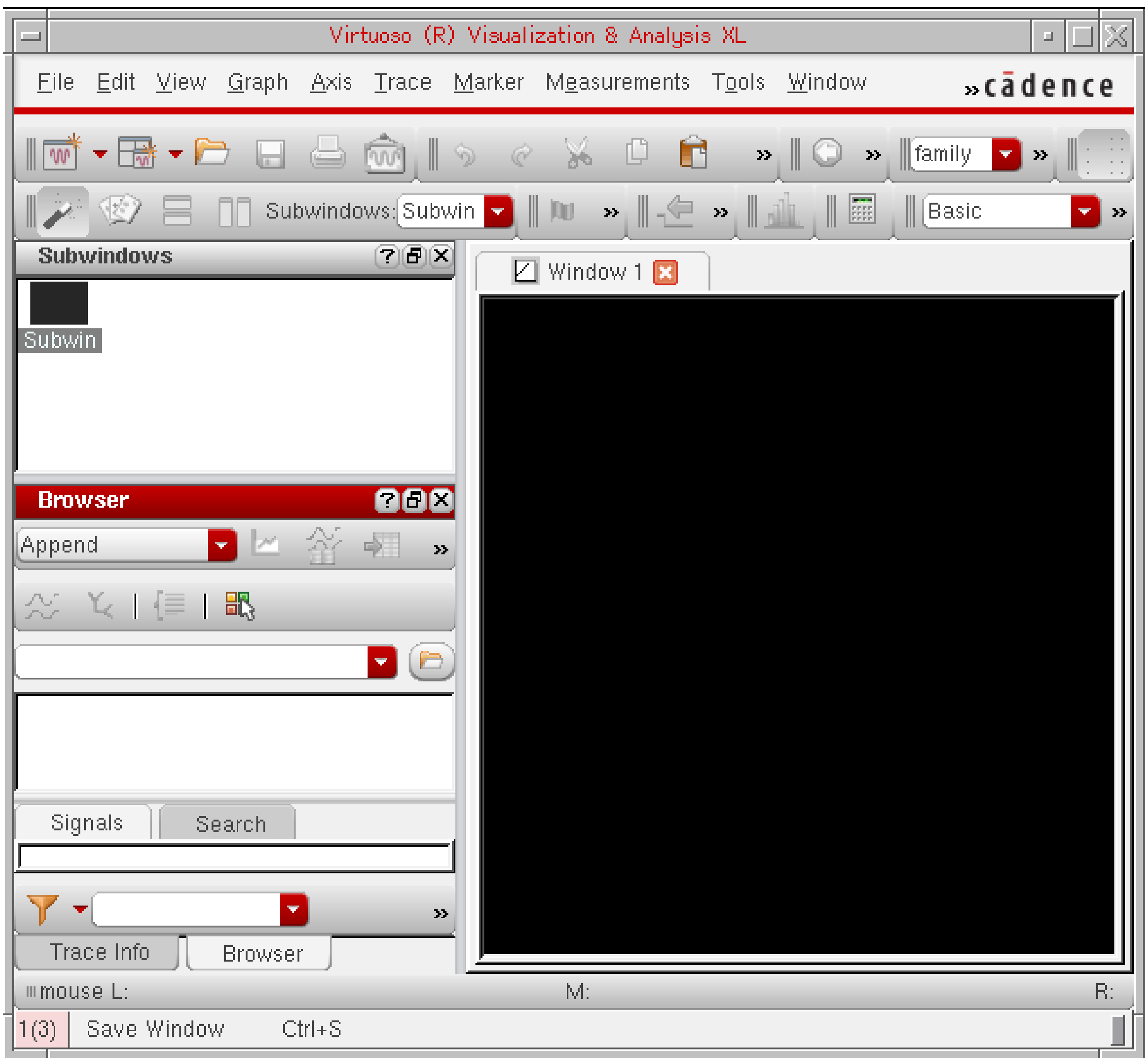
- Square brackets denote optional arguments. For example,

  viva [-expr SKILL]

1

Overview

You can use the Virtuoso Visualization and Analysis XL to browse, evaluate, analyze, and plot the simulation results.



Virtuoso Visualization and Analysis XL tool has the following three components:

- Results Browser
- Graph window
- Calculator

Results Browser is used to open the saved simulation results (signals). The signals are displayed in a hierarchical arrangement that corresponds to the hierarchy of your design, making it easy for you to locate and manage simulation data. For more information about the Results Browser, see Chapter 2, "Using the Results Browser,"

Graphs are displayed in the various graph windows and subwindows that can be opened in the graph display area in the Virtuoso Visualization and Analysis XL window. For detailed information about graphs, see Chapter 3, "Working with Graphs."

Calculator opens in a separate window and is used to create, store, and evaluate expressions for the signals displayed in the Results Browser. For detailed information about Calculator, see Chapter 4, "Working with the Calculator."

This chapter includes the following sections:

Opening the Virtuoso Visualization and Analysis XL

You can start the tool either from within ADE or in the standalone mode. When you open the tool from within ADE, you work on the simulation results for the latest run. However, in the standalone mode, you work on the saved simulation results that can be accessed through the Results Browser.

This section includes the following topics:

Opening the Tool in Stand-Alone Mode

To open the Virtuoso Visualization and Analysis XL window from CIW, do the following:

- Type the following command in a terminal window to open the tool in SKILL mode:

```
viva
    [-expr skill
    |-V
    |-W
    |-datadir
    |-mode XL]
```

Parameter	Description
`-h` or `-help`	Displays information about how to run the tool.
`-expr skill`	Starts the tool in the SKILL mode.
`-v`	Displays the version number for the tool.
`-w`	Displays the subversion number for the tool.
`-datadir`	Specifies the data directory to be opened on startup.
`-mode XL`	Specifies the product mode to be used. This determines the license (ADE L or Virtuoso Visualization and Analysis XL) to be checked out. The default mode is XL.

Note: The command line options for the Virtuoso Visualization and Analysis XL tool are not case-sensitive.

The Virtuoso Visualization and Analysis XL window appears.

- Type the following command in a terminal window to open Virtuoso Visualization and Analysis XL in MDL mode:

  `vivamdl`

  If you are using IC6.1.5 and previous releases, you can also use the following command to open Virtuoso Visualization and Analysis XL in the MDL mode.

  `viva -expr mdl`

Opening the Tool Using CIW

To open the tool using CIW, first you need to open Virtuoso and then do the following to open different tools in Virtuoso Visualization and Analysis XL:

- To open the Virtuoso Visualization and Analysis XL tool:
  - ❑ Choose *Tools – ViVA XL – Waveform.*
- To open the tool with the Results Browser displayed:
  - ❑ Choose *Tools – ViVA XL – Results Browser.*
- To open the Virtuoso Visualization and Analysis XL Calculator:
  - ❑ Choose *Tools – ViVA XL – Calculator*.

Opening the Tool Using Virtuoso® Analog Design Environment (ADE)

You can open the tool from within ADE L and XL.

To know about how to open the graph window from within ADE, see Opening the Graph Window from ADE L and ADE XL on page 71.

To know about how to open Results Browser from within ADE, see Opening the Results Browser from ADE on page 32.

To know about how to open Calculator from within ADE, see Opening the Calculator using ADE on page 349.

Closing the Virtuoso Visualization and Analysis XL

- To close the graph window:
  - ❑ In the graph window, choose *File – Close All Windows.*
  - ❑ Choose *File – Close Window*.

    Closes the active window. If there is only one open window, the *Close* command exits the tool.
- To close the Calculator window:
  - ❑ In the Calculator window, choose *File – Close*.

Saving a Session

When you save a session, the current state of the application is saved. When you reload a saved session, the same windows and settings as in the saved session come up. For more information about the graph attributes that are saved, see Saving and Loading Graphs on page 209.

Restoring a Session

To restore a previously saved session, type the following in an xterm window:

```
viva -statefile mystatefile.xml
```

where *mystatefile* is the name of your state file.

2

Using the Results Browser

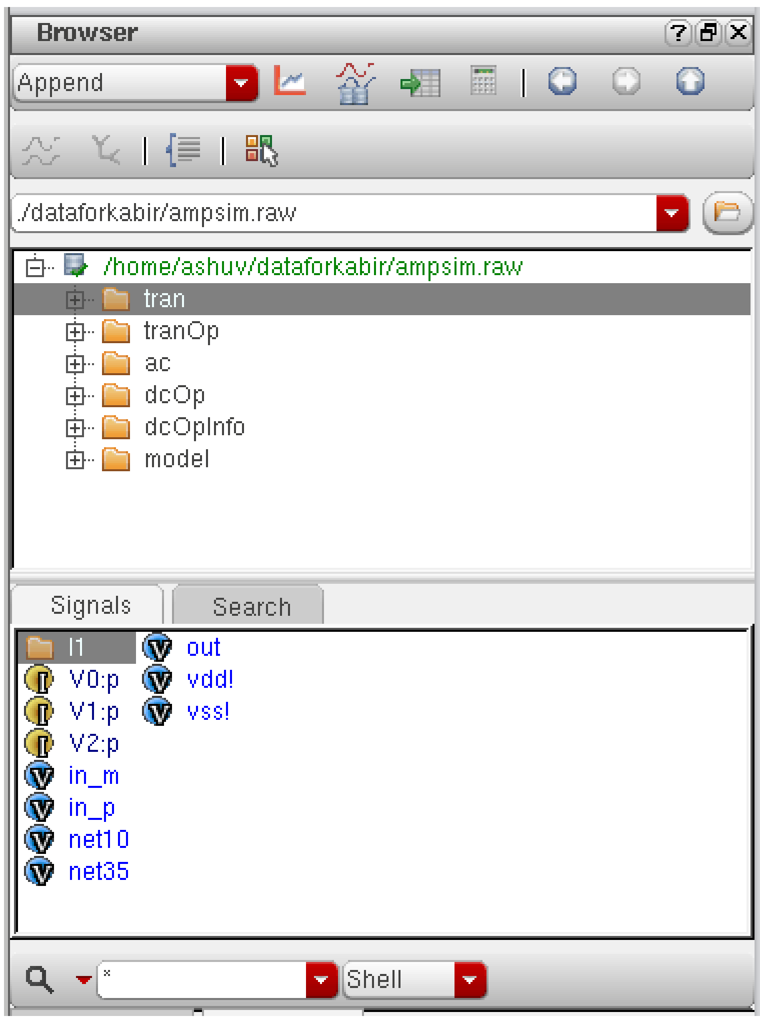
You can use the Results Browser, which is opened as an assistant in the Virtuoso Visualization and Analysis XL window, to open the stored simulation results. Results Browser displays simulation results (signals) in a hierarchical arrangement that corresponds to the hierarchy of your design, making it easy for you to locate and manage simulation data.

This chapter includes the following topics:

About the Results Browser

You use the Results Browser to access the simulation results that you save to work on later in a different session.

The results for each simulation are stored in a separate results directory. In addition, the results for different analysis types are stored in separate folders. For example, all simulation results for all transient analyses are contained in the `tran` folder. The various signals in a results directory are displayed in the Results Browser in a hierarchy that is determined by the design.



You can load multiple results directories in the Results Browser. The first directory that you load in the Results Browser becomes the in-context results directory. This means that the expressions in the Calculator are evaluated and the signals displayed in the graph are plotted by using the data available in the in-context results directory. To change the in-context results directory, see Changing In-Context Results Directory on page 42.

Important

Starting IC6.1.5, the Results Browser is opened as an assistant within the Virtuoso Visualization and Analysis XL window and the new browser is not available in the MDL mode.

Opening the Results Browser

The Results Browser is displayed as an assistant in Virtuoso Visualization and Analysis XL. You can open the Results Browser by using one of the following methods:

- Opening Results Browser in the Stand-Alone Mode on page 31
- Opening the Results Browser from Virtuoso on page 32
- Opening the Results Browser from ADE on page 32

Opening Results Browser in the Stand-Alone Mode

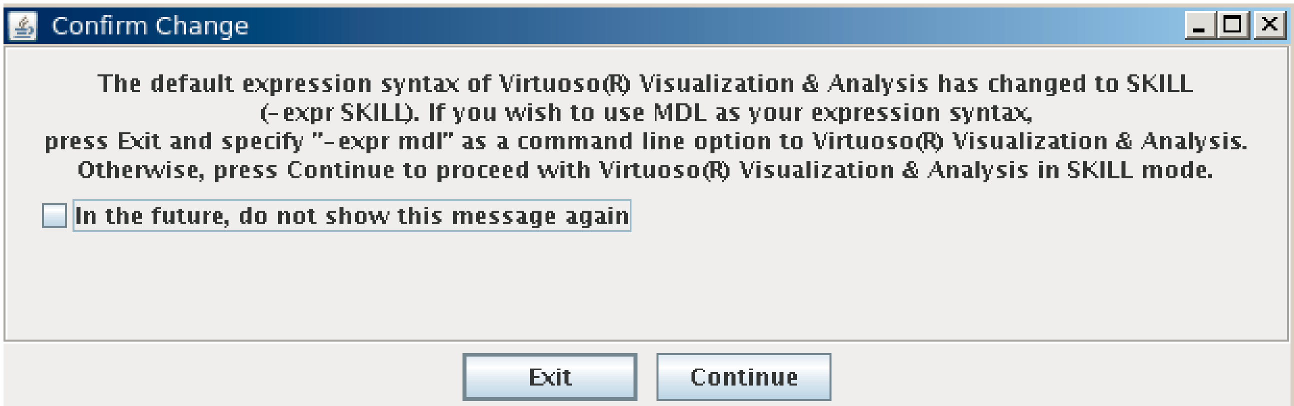
To open the Results Browser from Virtuoso Visualization and Analysis XL, do the following:

➡ Type the following in a terminal window:

```
viva -expr skill | mdl &
```

The *Virtuoso (R) Visualization & Analysis XL* window appears with the Results Browser displayed on the left. If you specify a data directory by using the `viva -dataDir` command, the directory is displayed in the Results Browser. Otherwise, the Results Browser is blank.

When you run this command, the tool checks for the `~/.cds_wavescan_expr_conf` file and if this file is missing, the following warning message is displayed:



Select the *In the future, do not show this message again* check box and run the command again.

Note: By default, Virtuoso Visualization and Analysis XL opens in the SKILL mode.

Opening the Results Browser from Virtuoso

To open the Results Browser from Virtuoso, do the following:

➡ In the CIW, choose *Tools – ADE L – Results Browser.*

The *Virtuoso (R) Visualization & Analysis XL* window appears with the Results Browser displayed on the left.

Opening the Results Browser from ADE

To open the Results Browser window from the Virtuoso® Analog Design Environment (ADE),

➡ In the ADE L or XL window, choose *Tools – Results Browser*.

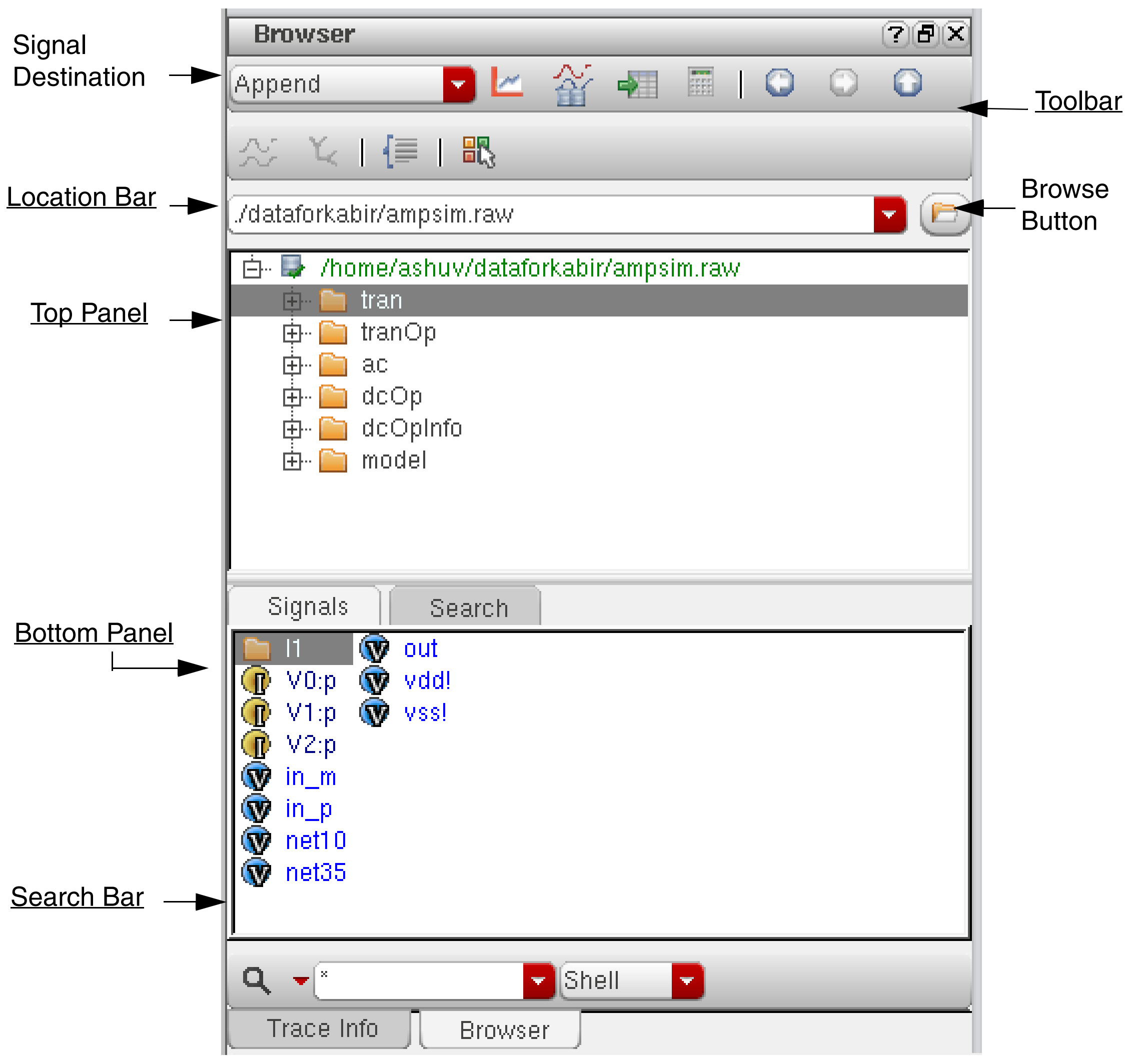
The *Virtuoso (R) Visualization & Analysis XL* window appears with the Results Browser displayed on the left.

If you open Virtuoso Visualization and Analysis XL after running a simulation in ADE, the Results Browser displays the current simulation results directory. Otherwise, the Results Browser is blank.

Results Browser Graphical User Interface

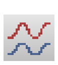
The Results Browser graphical user interface (GUI) consists of the following elements:

- Toolbar on page 34
- Location Bar on page 36
- Top Panel on page 36
- Bottom Panel on page 36
- Search Bar on page 37



Toolbar

The table below describes the toolbar buttons:

Button	Name	Description
	Signal Destination	Displays a list of destinations where a signal can be plotted—*Append*, *Replace*, *New Window*, and *New Subwindow*. By default the signal plot destination is *Append*. For more information, see Selecting the Signal Plot Destination on page 48.
	Plot signal	Plots the selected signal in the specified window. For more information, see Plotting Signals on page 47.
	Plot signal from all DBs	Plots in the append mode the selected signal from all open results directory. This helps compare signals that are common to all open results directories. For more information, see Comparing Signals on page 61.
	Send to Table	Displays the signal data in the Virtuoso Visualization and Analysis XL Table.
	Send to Calculator	Opens the Calculator and displays the expression for the selected signal in the Calculator Buffer. For more information, see Chapter 4, “Working with the Calculator.”
	Previous	Moves the control to the previous results directory selected in the Results Browser. When you open a new session, this button is not available.
	Next	Moves to control to the next results directory opened in the Results Browser.
	Up to Parent Directory	Moves the control to one level up in the results directory hierarchy.
	Diff	Plots the difference of two signals in the graph window. For more information, see Plotting the Difference of Two Signals on page 59.

	Y Vs Y	Plots the Y Vs Y of two signals. For more information, see Plotting YvsY for Two Signals on page 56.
	Select Sweep Data	Enables you to select the range for the sweep data. For more information, see Selecting and Plotting Signals in a Data Range on page 52.
	Select Color	Enables you to change the default color of different signal types. For more information, see Changing Signal Color on page 41.

If the results directories contain the simulation results for an AC analysis, the Results Browser toolbar displays the following two additional drop-down lists:

Name	**Options**	**Description**
Graph Type 		Available only for AC data
	Default	Plots the signal to the default graph type
	Rectangular	Plots the signal to a rectangular graph
	Polar	Plots the signal to a polar graph
	Impedance	Plots the signal to an impedance graph
	Admittance	Plots the signal to an admittance graph
	RealvsImag	Plots real versus imaginary
Graph Modifier 		Available only for AC data
	Magnitude	Plots magnitude versus frequency
	Phase	Plots phase versus frequency
	WPhase	Plots wrapped phase versus frequency
	Real	Plots real versus frequency
	Imaginary	Plots imaginary versus frequency

db10	Plots *db10* versus frequency
db20	Plots *db20* versus frequency
dbm	Plots *dbm* versus frequency

Location Bar

Lists the paths for the last 20 results directories opened in the Results Browser.

If the results directory path name is long, it is displayed in the right-justified format in the Location Bar. This helps differentiate between the name of the results files when multiple results directories are opened in the Results Browser.

To open a results directory, select it in the drop-down list. To open a new results directory that was loaded earlier in the Results Browser, enter the path in the box, or click the *Browse* button on the right. For more information, see Opening a Results Directory on page 38.

Top Panel

The top panel displays all loaded results directories in a tree view. The symbol displayed before a results directory indicates the PSF directories.

Right-click a results directory in top left panel and choose any of the following commands:

- *Close Results*—To close the selected results directory.
- *Open Terminal*—To open a terminal at the selected directory.
- *Set Context*—To set the selected directory as the in-context results directory.

Bottom Panel

The bottom panel displays in a list view the contents of the selected database.

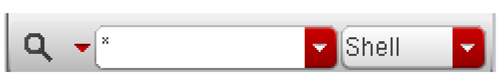
Right-click a signal in the bottom panel and choose any of the following commands:

- *Up One Level*—To moves up by one directory level.
- *Plot Signal*—To plot a signal in the specified graph window. By default, the signal is plotted in the append mode.

- *Plot from all DBs*—To plot in the append mode the selected signal from all the loaded results directory in the append mode to compare the signals. For more information, see Comparing Signals on page 61.
- *Export*—Exports a waveform. For more information, see Exporting Signals on page 62.
- *Calculator*—Opens the Calculator. For more information, see Chapter 4, "Working with the Calculator."
- *Table*—This button is not available in IC6.1.5.
- *Diff*—Plots the difference of two signals. For more information, see Plotting the Difference of Two Signals on page 59.
- *YvsY*—Plots Y versus Y for two signals. For more information, see Plotting YvsY for Two Signals on page 56.
- *Append, Replace, New Subwindow, and New Window*—Specifies the plot destination. For more information, see Selecting the Signal Plot Destination on page 48.

Search Bar

Enables you to filter and search signals across multiple databases in the Results Browser.



For more information about filtering and searching signals, see Filtering and Searching For Signals on page 43.

DB Type Identifier

When you point to an open database in the Results Browser, the following information is displayed in the given order for the folders in the database.

- Format
- Analysis Type
- Description
- Date
- Simulator

Working with the Results Directory

This section describes how you can open a results directory, select signals in the results directory, and perform filtering and searching operations on the signals in the results directory.

- Opening a Results Directory on page 38
- Selecting Signals on page 40
- Filtering and Searching For Signals on page 43
- Updating Results Directory Data on page 47

Opening a Results Directory

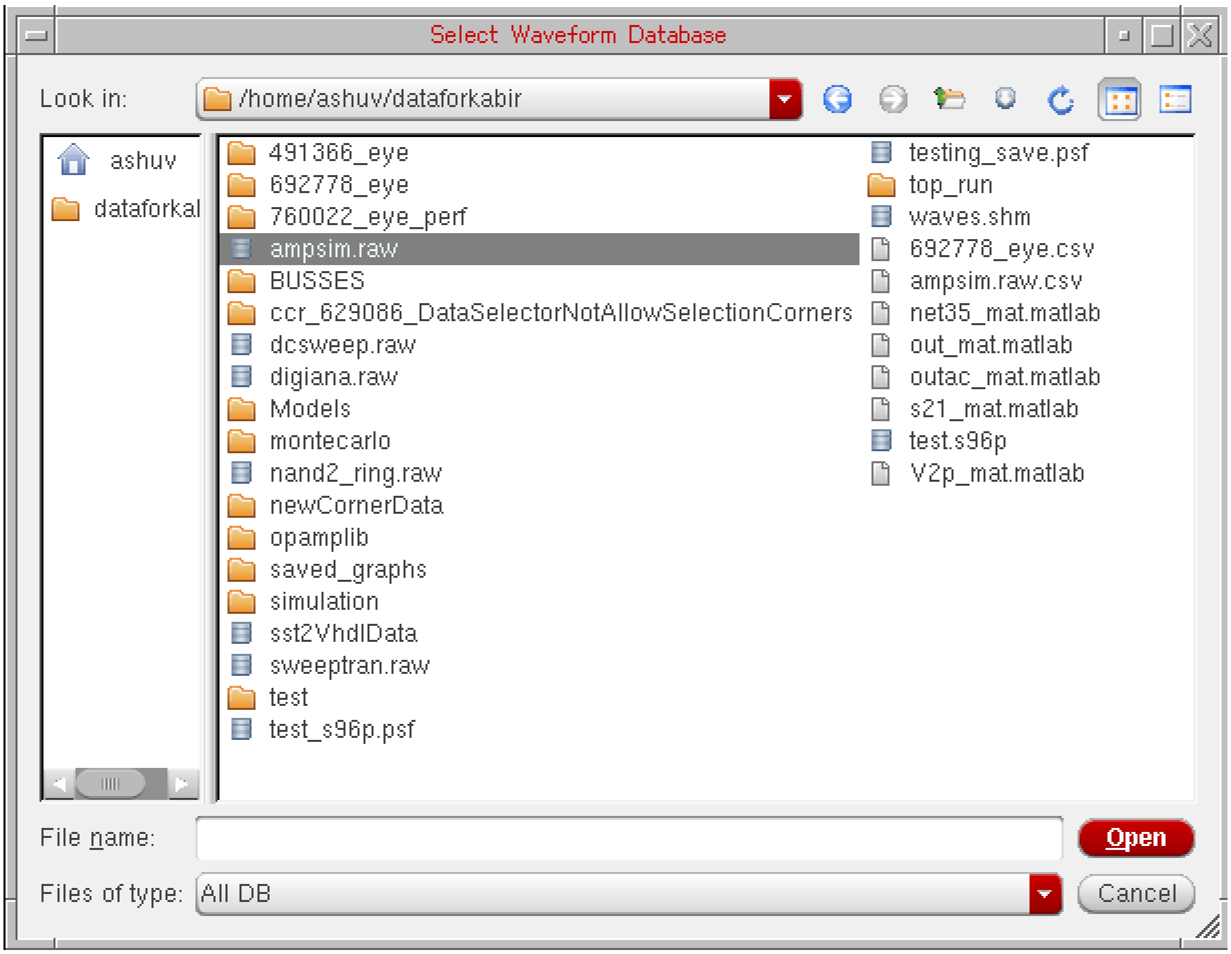
To open a results directory in the Results Browser, do one of the following:

- In the Results Browser window, type the results directory path in the location bar and press the `Enter` key.
- In the Results Browser window, select a path from the drop-down list. The drop-down list displays the paths for previously opened results directories. You can also click the browse button and select the required results directories.

  In the Virtuoso Visualization and Analysis XL window, do one of the following to open a results directory:

  - Choose *File – Open Results*.
  - Choose *Browser – Results – Open Results*.

    The *Select Waveform Database* form appears. This form has two panes—the directory pane, also called Favorites, on the left displays the parent directories and the content pane on the right displays the files or directories contained in the parent directory that you open. the directory pane includes the links to the home directory and the current working directory. You can drag the directories or files and place them in the directory pane to have an easy access to them. Now, when you open this form next time, the form is opened with the same settings.



In this form, do one of the following to select the required results directory:

- ❑ In the *Look In* field, browse to locate the results directory that you want to open. If required, you can open multiple results directories. By default, this field displays the path of the directory from where the you opened the results recently in the Results Browser.

- ❑ Select the results directory that you want to open. You can select multiple directories at a time by using the `Ctrl` key.

  The form includes the following three buttons in the upper-right corner:

  - ❍ *Parent Directory*—To open the parent directory. This button is available if you have nested results directories.

- *Descend Inside Directory*—To view the contents of the parent directory and to move to the next-level directory.
- *Refresh*—To refresh the results directories present in the selected location.
- *List View*—To display directories in a list view.
- *Detail View*—To display directories in the detail view.

- Click *Open.*

  The selected results directory is opened in the Results Browser. If this is the first directory that is opened in Results Browser, it is dispalyed in green and the icon is displayed next to the directory name to indicate that it is the in-context results directory.

Note: The Virtuoso Visualization and Analysis XL tool displays the icon next to the directories that contain simulation data.

Changing In-Context Results Directory

To change the database context directory from the Results Browser, perform the following steps:

- Open the results directory that you want to use to build expressions.
- Right-click the results directory name and choose *Set Context*.

  The database context is set to the selected directory.

Selecting Signals

In the Results Browser, the top pane displays the results directories and the bottom pane displays the datasets in the selected results directory.

Perform the following steps to select a signal in the Results Browser:

1. Double-click a results directory in the top panel.

   The results directory is expanded and the folders containing data from different analyses are displayed in the bottom panel.

   Note: The icon displayed next to a directory in a list a indicates that the directory contains subdirectories. You can double-click the directory to display or to hide the subdirectories.

2. Double-click the appropriate dataset folder.

The signals are displayed in the bottom panel. Each signal has an icon displayed next to it, indicating the signal type.

- ❑ indicates a voltage signal
- ❑ indicates a current signal
- ❑ indicates a digital signal
- ❑ indicates a digital bus
- ❑ indicates a power signal
- ❑ indicates all other signal types
- ❑ indicates s-parameters
- ❑ indicates a WREAL signal

3. Right-click the signal and choose the command to either plot it or send it to the Calculator Buffer. For more information about these operations, see the following chapters:

- ❑ Graphs: Chapter 3, "Working with Graphs."
- ❑ Calculator: Chapter 4, "Working with the Calculator."

Selecting Multiple Signals

Do one of the following to select a set of consecutive signals:

- Click the first signal. Hold down the `Shift` key, and then click the final signal in the set.
- Drag the pointer to select a set of consecutive signals.

Do the following to select a set of non-consecutive signals:

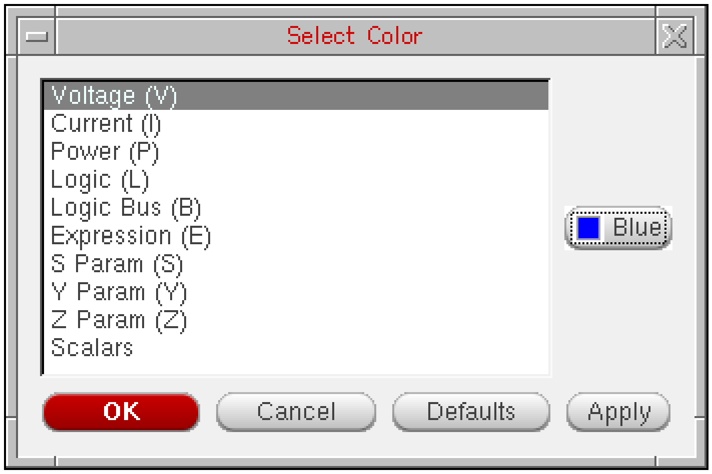
- Click the first signal. Hold down the `Ctrl` key, and click one by one the required signals.
- Drag to select appropriate signals.

Changing Signal Color

To differentiate between the various signal types, such as voltage, current, analog, and digital signals, you can change the color of signals. Do the following to change signal color:

1. Click the button on the Results Browser toolbar.

The *Select Color* form appears.



2. In this form, select the signal type for which you want to change the color.

   Note: Alternatively, select a signal and then click the button to open the *Select Color* form. The signal type that you have selected is highlighted in this form.

3. Click the color button and select the color you want to apply.

4. Click *OK*.

Changing In-Context Results Directory

The first results directory that you open in the Results Browser is set as the in-context results directory and displayed in green. The symbol is displayed to the left of the in-context results directory path.

To change the in-context results directory, open that results directory, which you want to make in-context, in the top panel and do one of the following:

- Right-click the results directory path and choose *Set Context.*
- Choose *Browser – Results – Set Context*.

The selected results directory becomes the in-context results directory.

Setting the in-context results directory plays an important role in performing the following tasks:

- Building expressions in the Calculator. For more information, see Results Toolbar on page 310.

- Reloading graphs in the graph window. For more information, see Reloading Graphs on page 220.

Filtering and Searching For Signals

You can filter and search for signals in a selected database according to the signal type or using the signal name. When you open Virtuoso Visualization and Analysis XL from within ADE XL, you can also search for signals across current datasets.

This section includes the following topics:

Filtering Signals

You can filter signals by using the filter toolbar located at the bottom of the Results Browser. By default, this toolbar is displayed when the Results Browser is opened. You can also click the *Signals* tab to view this toolbar.

Note: The filtering of signals is performed only in the currently selected folder in the Results Browser.

To filter signals:

➡ From the  drop-down list, select one or more options listed in the table below:

Item	Description
All	Displays all signals
Logic	Displays logic (digital) signals
LogicBus	Displays logicbus (digital bus) signals
V	Displays voltage signals
I	Displays current signals
W	Displays power signals

Item	Description
Enum	Displays Enum signals
Wreal	Displays WREAL signals
Show hierarchical nodes	Displays folders containing signals

The available options are determined by the selected dataset. For example, the available options for an analog dataset are *All*, *V*, and *I*.

1. Type the filter pattern in the provided text box.
2. Select `Shell` or `RegExp` from the drop-down list to filter signals based on a shell or regular expression.
3. Select the *Match case* check box to perform case-sensitive filtering.
4. Click  .
5. All the signals matching the specified filter pattern are displayed.

Searching For Signals

To display the search bar, click the *Search* tab in the Results Browser window. The search toolbar is displayed at the bottom of the Results Browser. By default, search is performed in the selected folder and its subfolders.

To search for a signal in the current database:

- In the  drop-down list, select one or more options as listed in the table in the previous section.

  Note: If you want folders that contain signals matching the search criteria to be displayed, ensure that the *Show hierarchical nodes* check box is selected.

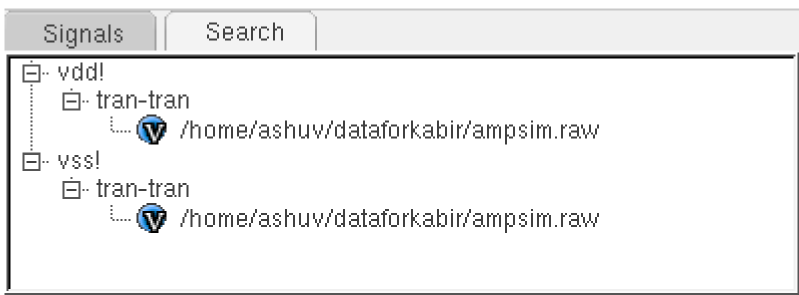
- Type the search string in the *Specify Search Pattern* field, which includes a text box.
- Select `Shell` or `RegExp` from the drop-down list to search signals based on a shell or regular expression.

- Select the *Match Case* check box to perform a case-sensitive search. By default, search is not case-sensitive.
- Select the *All DBs* check box to search the specified pattern in all the open results databases in the Results Browser.
- Click the button.

Examples

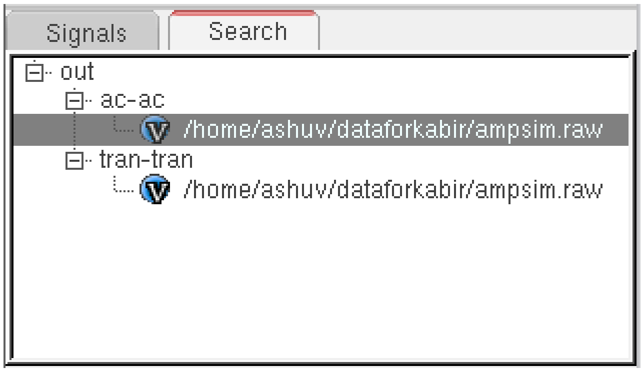
- Type `v(dd|ss)` to search for all the voltage signals that contain `vdd` and `vss` in their names.
- Type `n[1-9]` to search for numbered nets.

  The signals that meet the specified search criteria are displayed in the bottom panel of the Results Browser, as shown in the figure below:

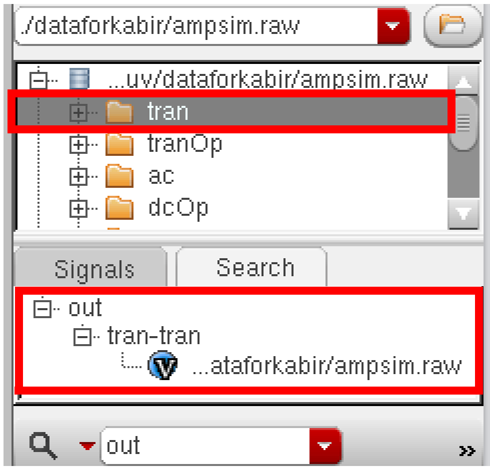


If you search for a signal by typing the signal name in the search pattern field and you get multiple occurrences of the signal, the different signals with the same signal name are listed out separately in the search output. You can view the path to find the difference between the signals. For example, in the figure below, the `out` signal is found in two folders, `ac-ac` and `tran-tran`.

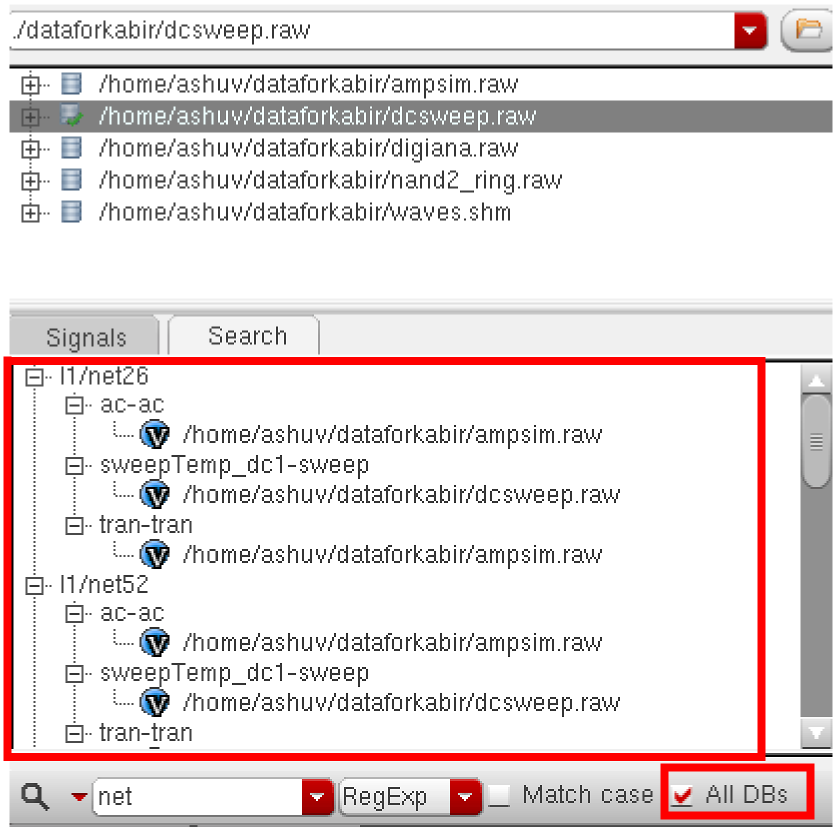
The search output also includes the icons to indicate the type of the searched signals. For example, is displayed in the figure below to indicate that `out` is a voltage signal. Also, if the results directory path is long to be fit in the search pane, it is displayed in a right-justified format in the search output to improve the readability. If you move the mouse pointer on this path, the tool-tip displays the complete path name.



By default, signal search is performed at folder-level, which means only the currently selected folder and its subfolders are searched to find the signal matching the specified search pattern. For example, if you select the `tran` folder, as shown in the figure below, and search for the `out` signal, the search is performed only in the `tran` folder and its subfolders. If the signal exists in this folder, it is listed in the search output.



However, you can also search for a signal in all the opened databases in the Results Browser by selecting the *All DBs* check box. The figure below displays the search results for the `net` signal when searched in all the opened databases and datasets.



Updating Results Directory Data

When you re-simulate your design, the Virtuoso Visualization and Analysis XL tool automatically refreshes the data directory if you perform an action that accesses the database. All new graphs that you create display the updated data, which helps you monitor long simulations. The graphs that are already open are not updated.

Plotting Signals

To plot a signal, right-click the signal in the Results Browser and choose *Plot Signal*. The signal is plotted in the specified graph window. For detailed information about how to create a graph, see Chapter 3, "Working with Graphs."

Note: You cannot plot scalar values in the graph window. Therefore, if you try to plot a scalar value from the Results Browser, a warning message is displayed on the CIW.

This section describes the available plot destination options and the graph types that you can use to display the plotted signal.

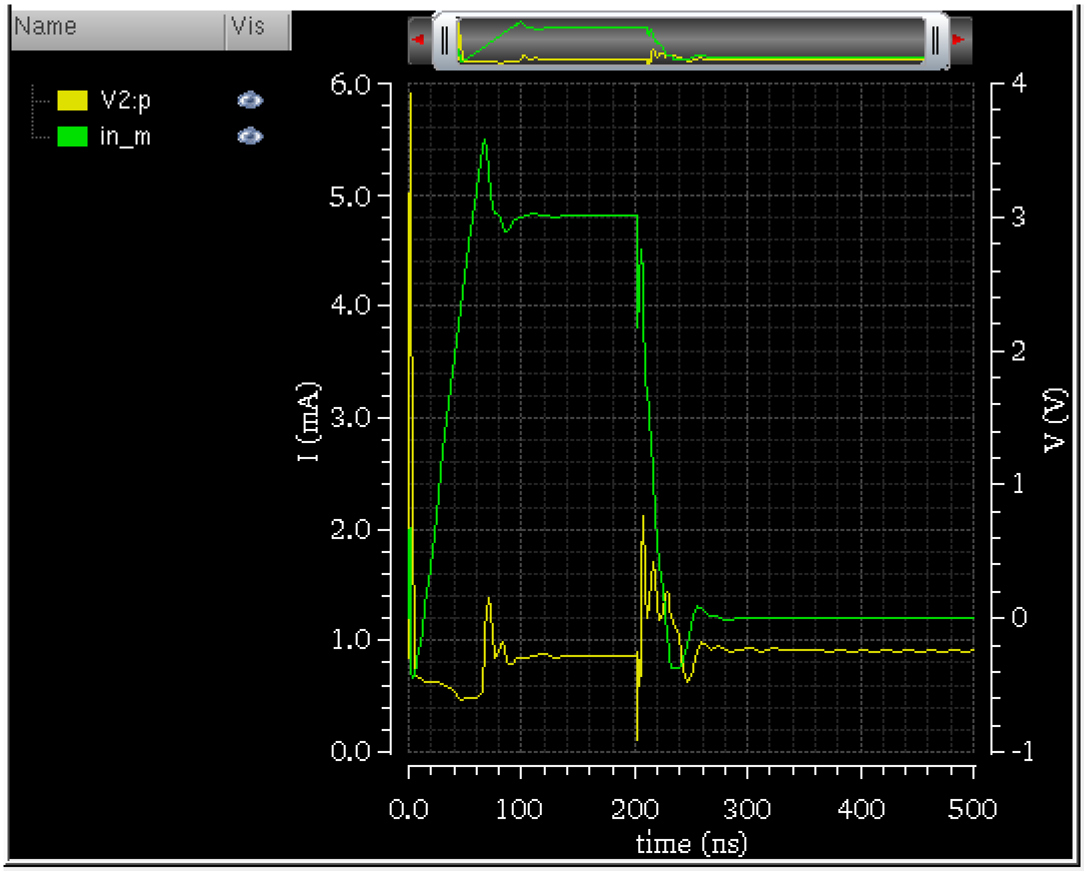
- Selecting the Signal Plot Destination on page 48
- Selecting the Graph Type on page 50

Selecting the Signal Plot Destination

The first graph that you plot in a session is displayed in a new graph window. If you already have a graph window open, you need to specify the destination to plot the new signal. This section describes the available destination options. For information about how to specify the signal destination, see step 2 in Selecting and Plotting Signals in a Data Range on page 52.

Appending to a Graph

You can append the trace for a signal to a graph that is already plotted in the graph window. If the traces share the same unit, the new trace is assigned to the same Y-axis. Otherwise, it is assigned to a new Y-axis. If the graph window already has four Y axes, or if the units do not match, the Virtuoso Visualization and Analysis XL tool plots the new signal in a new subwindow in that graph window.



The figure above shows the trace for the `V2:p` signal appended to the graph containing the trace for the `in_m` signal. The Y-axis unit for `in_m` is `V` and the Y-axis unit for V2:p is `mA`. Therefore, the trace for `V2:p` is assigned to a new Y-axis.

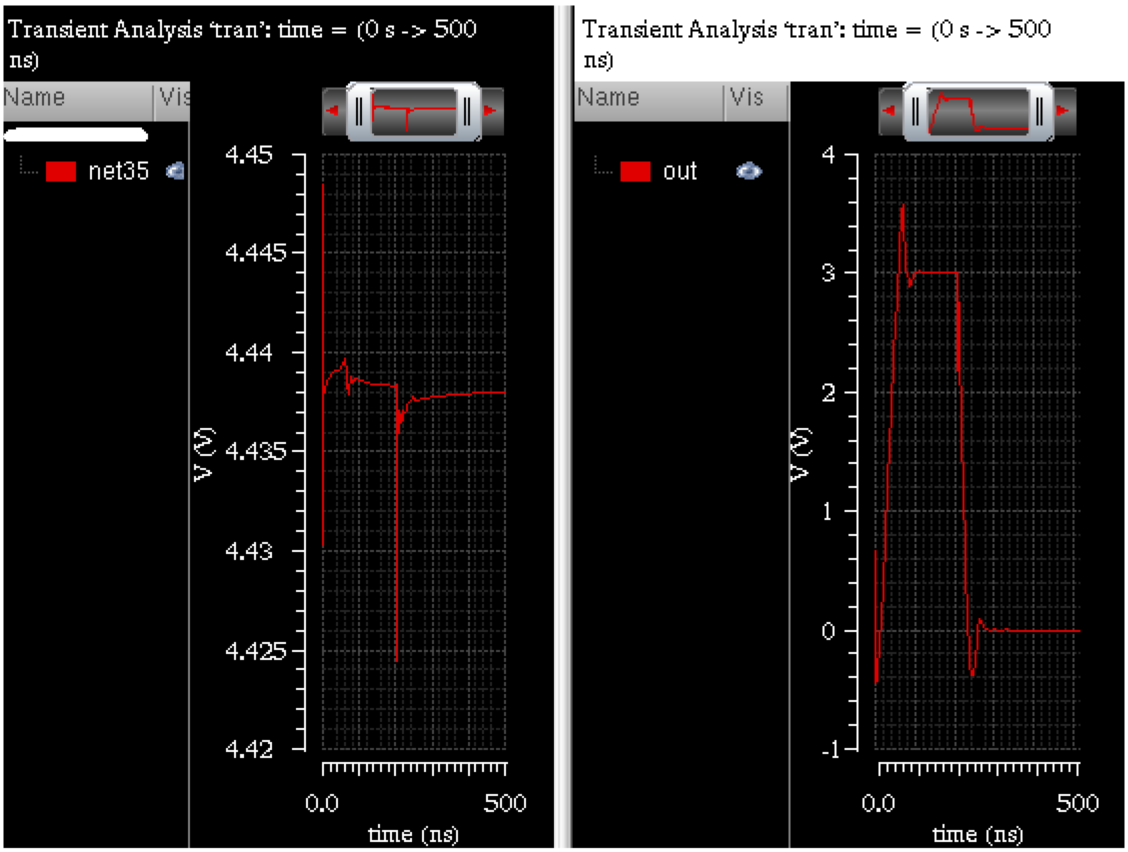
Replacing a Signal

You can plot a new signal to replace a signal or group of signals in the selected graph window or subwindow. The graph window or subwindow retains all its attributes. If the signal to be replaced contains markers, the markers are attached to the new trace if both signals have the same name.

If no graph window is open, a new graph window is created.

Plotting to a New Subwindow

You can plot a graph in a new subwindow within the selected graph window. The following figure shows the graph for the `V2:p` signal plotted in a subwindow.



Plotting to a New Window

You can plot a signal to a new graph window.

Important

If you attempt to plot a signal that do not include any data, the following warning message appears in the CIW: *No valid waveforms for plotting.*

Selecting the Graph Type

You can use different graph types to represent different data types. The Virtuoso Visualization and Analysis XL tool supports rectangular, polar, admittance, impedance, and real versus imaginary graph types. Each dataset represents a specific analysis type and can be plotted only to a specific graph type.

This section describes the available graph types. For information about how to specify the graph type, see step 3 in Selecting and Plotting Signals in a Data Range on page 52.

Default

The default graph type is determined based on the type of data in the simulator data file. For example, the default graph type for transient data is rectangular.

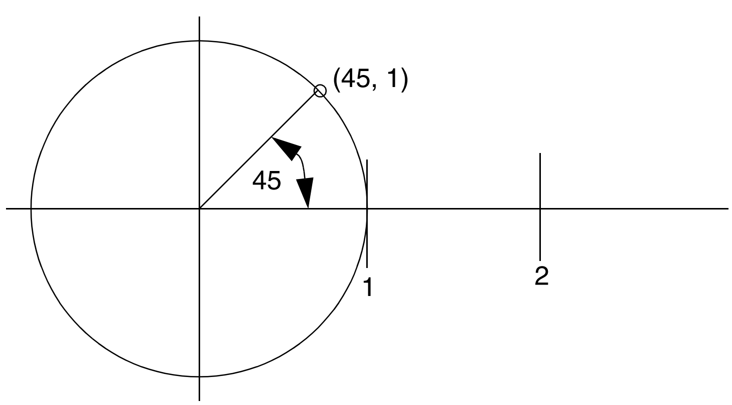
Rectangular Graphs

Transient and DC sweep data is always plotted in a rectangular graph. You can also plot portions of complex data in a rectangular graph by selecting the modifier: real, imaginary, magnitude, or phase. The Virtuoso Visualization and Analysis XL tool plots the selected modifier against the frequency.

Polar Graphs

Polar graphs represent data by using the polar coordinates system. Points are plotted at a given radial distance along a ray that creates a given angle with the positive X-axis.

The following example illustrates how you can plot a point (45 degrees, 1).



Admittance and Impedance Graphs

Admittance and impedance graphs are a direct graphical representation, in the complex plane, of the complex reflection coefficient. They reveal the complex impedance anywhere along a line.

The center of the chart normally represents 50 ohms but can be any impedance line you want—it is normalized to 1.0 units. Everything is scaled relative to the unit you choose. The nature of impedance is that of a real or resistive portion, and an imaginary, or reactive portion, combined in the Pythagorean style.

The circular graph has four goalposts spaced 90 degrees apart graphically and 45 degrees apart electrically. Two goalposts are resistive—one a short and the other an open—the left and right sides, respectively. The top and bottom posts are reactive, either inductive, or capacitive. Every point in between represents the various combinations resulting from a mismatched condition.

You can display either impedance or admittance grids in the Smith chart that you create—the grids are mirror images of each other.

Real Vs Imag

Real Vs Imag graphs plot the real part against the imaginary part. These graphs are available only for AC data.

Selecting and Plotting Signals in a Data Range

The data range feature in the Virtuoso Visualization and Analysis XL tool makes it easy to use a very large dataset efficiently by opening only the portion of the dataset that you need. You can specify a particular time range in a transient analysis, and then open the dataset and plot signals that fall within that range. For data families like corners or sweeps, you can load specific points in the analysis by using the data range feature.

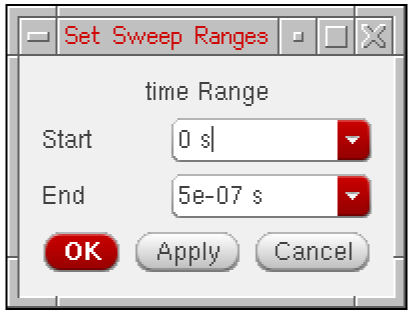
Any data filtered out by the data range is not available for plotting until it is enabled in the *Set Sweep Ranges* form, as described below.

Plotting a Signal over a Time Range

You can plot transient data over a time range. To specify the time range for a transient dataset, do the following:

1. Select a transient signal.
2. Click the  button on the Results Browser toolbar.

The *Set Sweep Ranges* form appears.



The *Start* and *End* fields display the signal range.

3. In the *Start* field, type the time at which you want the plot to begin.

   If you want the graph to be plotted from the first data point in the signal, select *Default* from the *Start* drop-down list.

4. In the *End* field, type the time at which you want the plot to end.

   If you want to display the time for the last data point in the signal, select *Default* from the drop-down.

5. Click *OK*.

6. Click .

   The graph window appears with the graph plotted for the specified time range.

Plotting Parametric Sweep Data

A parametric analysis sweeps a parameter or a group of parameters and runs one or more analyses for each combination of parameters.The Virtuoso Visualization and Analysis XL tool helps you analyze the resulting data efficiently.

To plot a part of parametric swept data, do the following:

1. Select a parametric signal.

2. Click the  button on the Results Browser toolbar.

   The *Set Sweep Ranges* form appears, displaying a list of inner sweep variables.

3. Select the values that you want to plot and click *OK*.

4. Click .

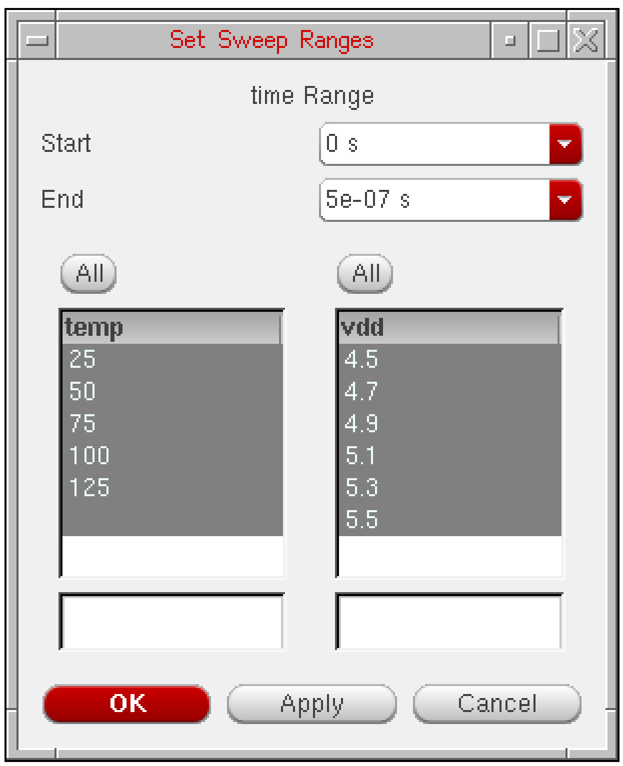
The graph window appears with graphs for all the combinations of sweep variables.

An Example of Plotting Data from a Parametric Sweep

To plot selected curves from the parametric data, do the following:

1. In the Results Browser window, open the data results for a parametric sweep.
2. Select a parametric signal in the results data directory.
3. Click the  button.

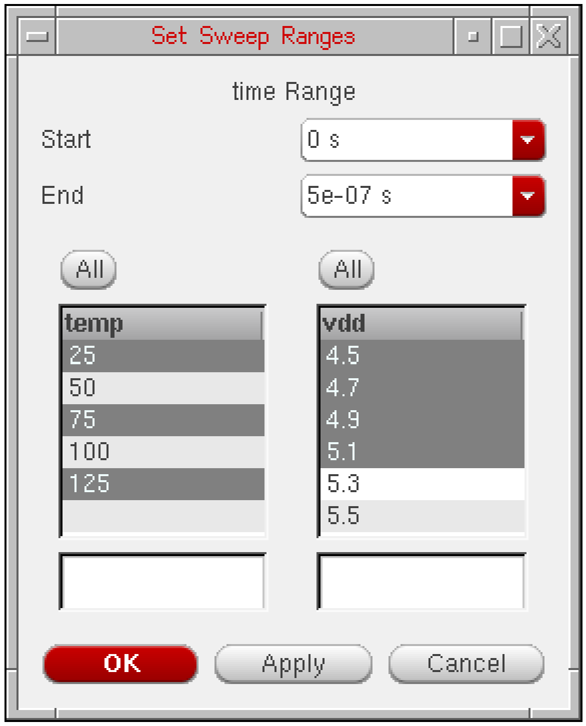
   The *Set Sweep Ranges* form appears. The *Start* and *End* fields display the range of the signal. All the `temp` (temperature) and `vdd` (inner sweep variable) values are selected by default.



   Note: You can also use this form to filter and sort sweep variables by typing their values in the boxes at bottom.

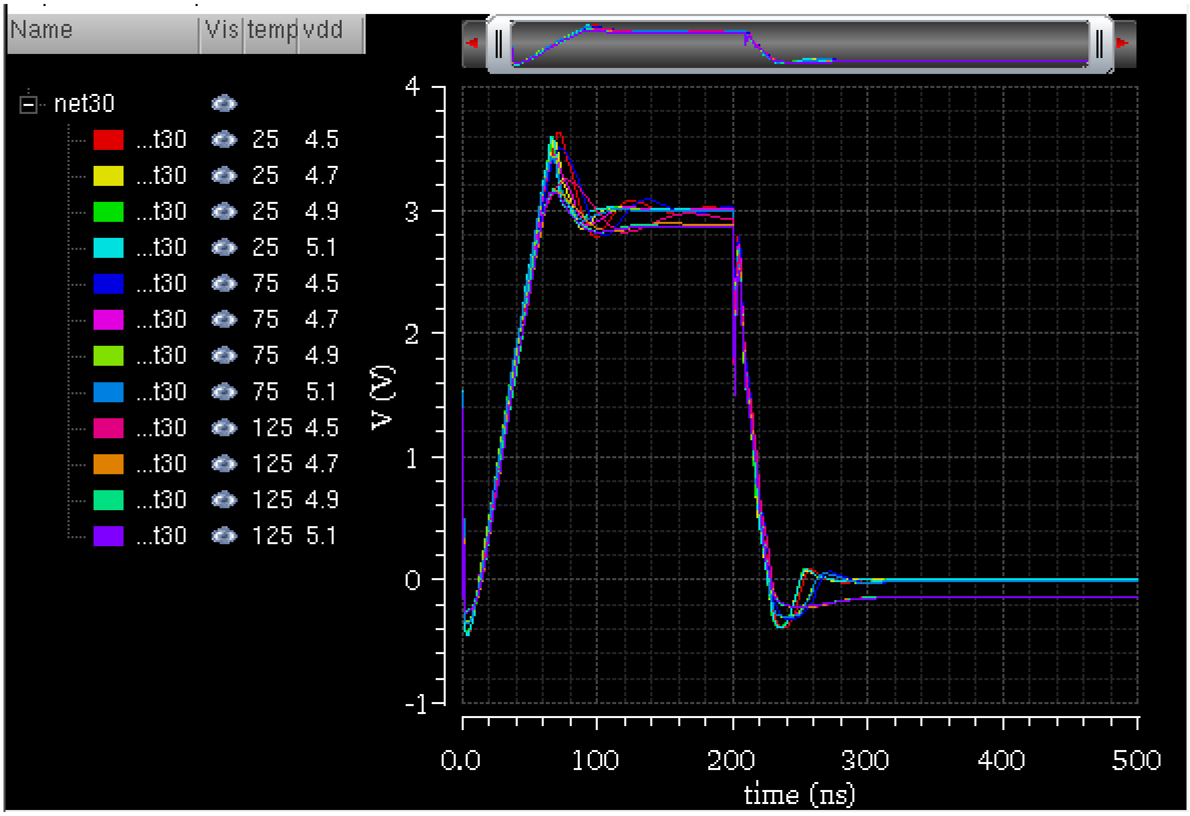
4. In the *temp* list, click `25.0`, hold down the `Ctrl` key, and click `75.0`, and `125.0`.
5. In the *vdd* list, click `4.5`, hold down the `Shift` key, and click `5.1`.

The form now displays the data ranges as shown in the figure below.



6. Click *OK*.
7. In the Results Browser window, right-click the `out` signal and choose *New Window*.

The graph window appears with the traces plotted for the parametric family. Each trace in the family is annotated by a sweep path that describes the parameter-value pair.



Creating Special Plots

This section describes how to plot the YvsY and the difference of two signals. This section also describes how you can compare signals for various datasets contained in different results directories.

- Plotting YvsY for Two Signals on page 56
- Plotting the Difference of Two Signals on page 59
- Comparing Signals on page 61

Plotting YvsY for Two Signals

This section describes how to plot the Y-axis values of one signal versus the Y-axis values of another signal. You can plot YvsY to measure input offset voltage, which displays the offset between the input and output of the circuit.

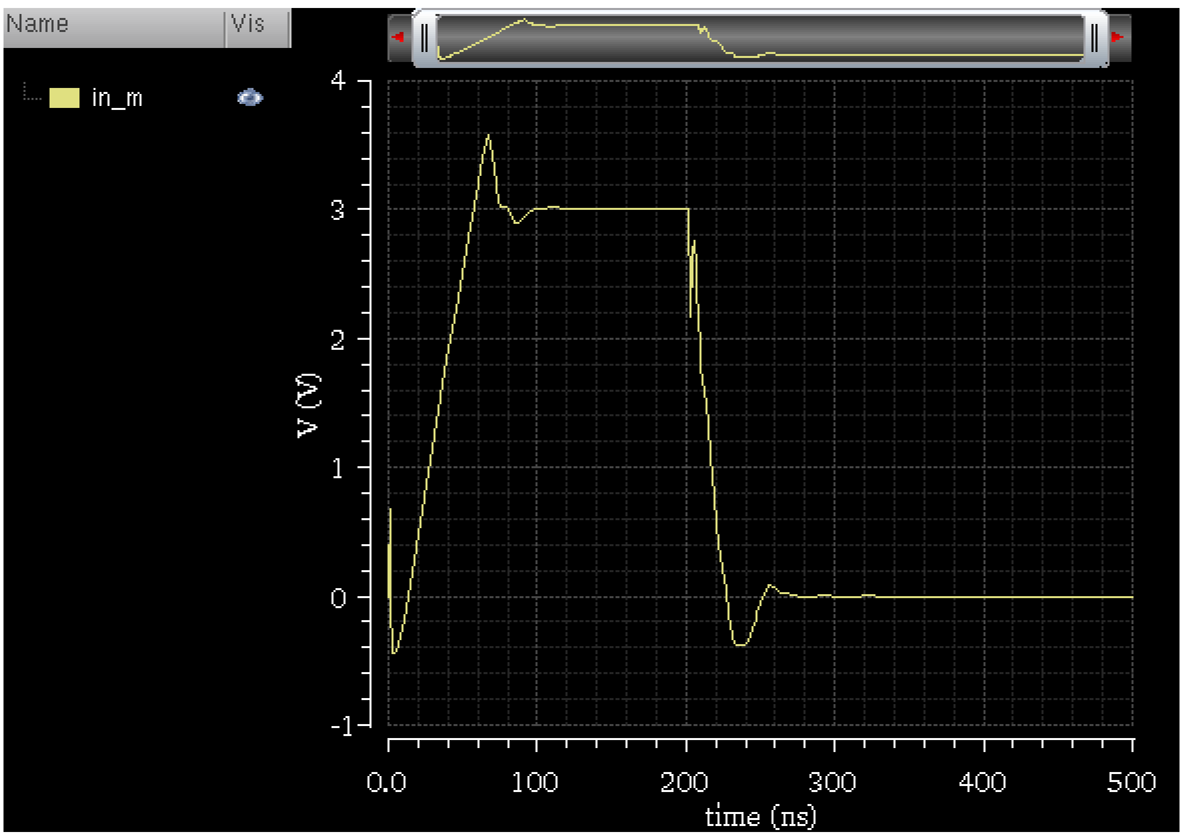
To plot YvsY, select a signal in the Results Browser and do one of the following:

- Click the  button and select the second signal
- Select the second signal and click the  button.

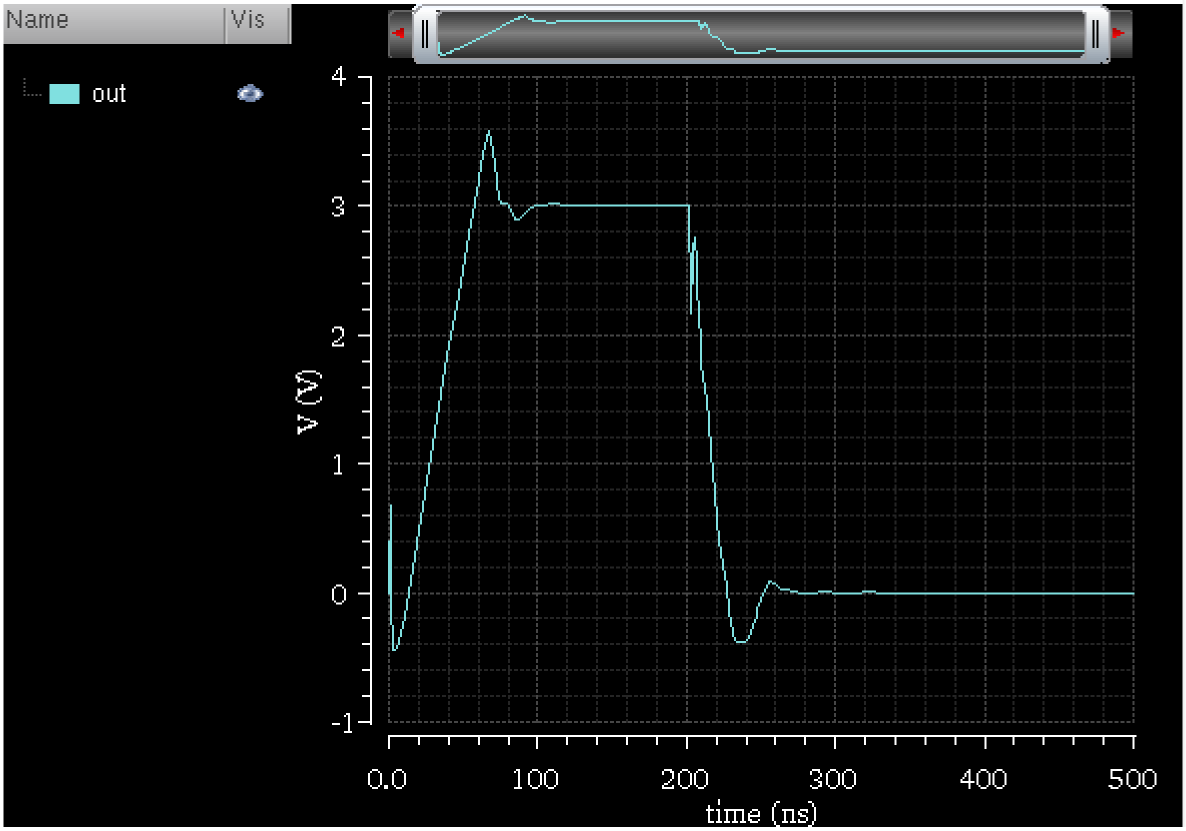
  The graph window appears with the trace plotted for Y versus Y.

Note: The YvsY plots cannot be created when the dataset includes parametric swept data. Therefore, the YvsY option is not available for signals resulted from a parametric sweep analysis.

The following figures illustrate the `out` and `in_m` signals.
Signal `in_m`:

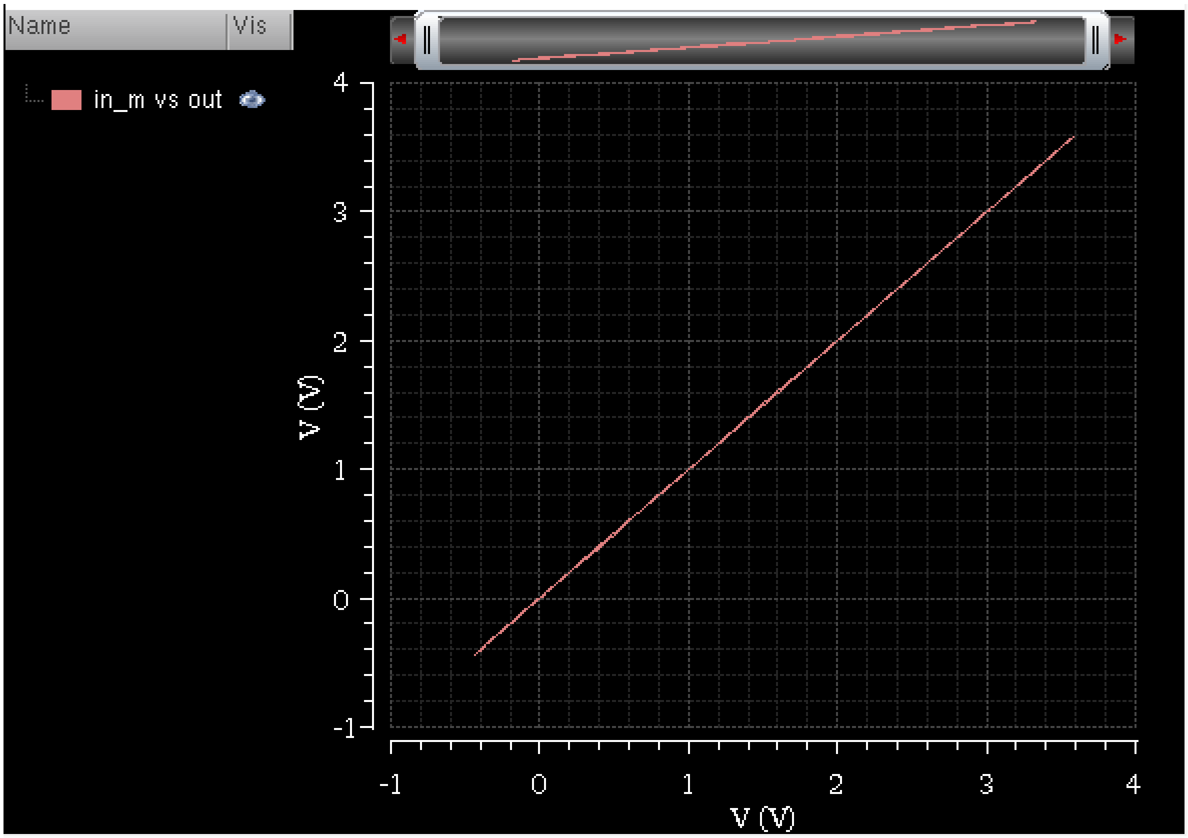


Signal `out`:



When you plot a YvsY graph for these signals, the result is a diagonal line because the signals are identical, as shown in the figure below.

YvsY for `in_m` and `out`:



Plotting the Difference of Two Signals

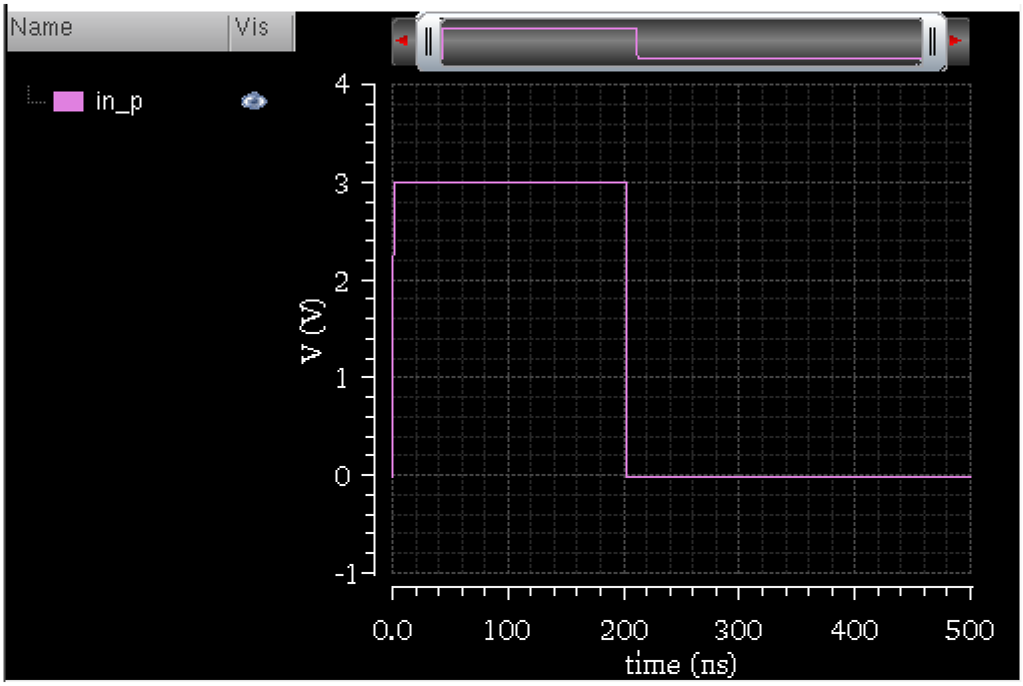
To plot the difference of two signals, select a signal in the Results Browser and do one of the following:

- Click the  button and select the second signal.
- Select the second signal and click the  button.

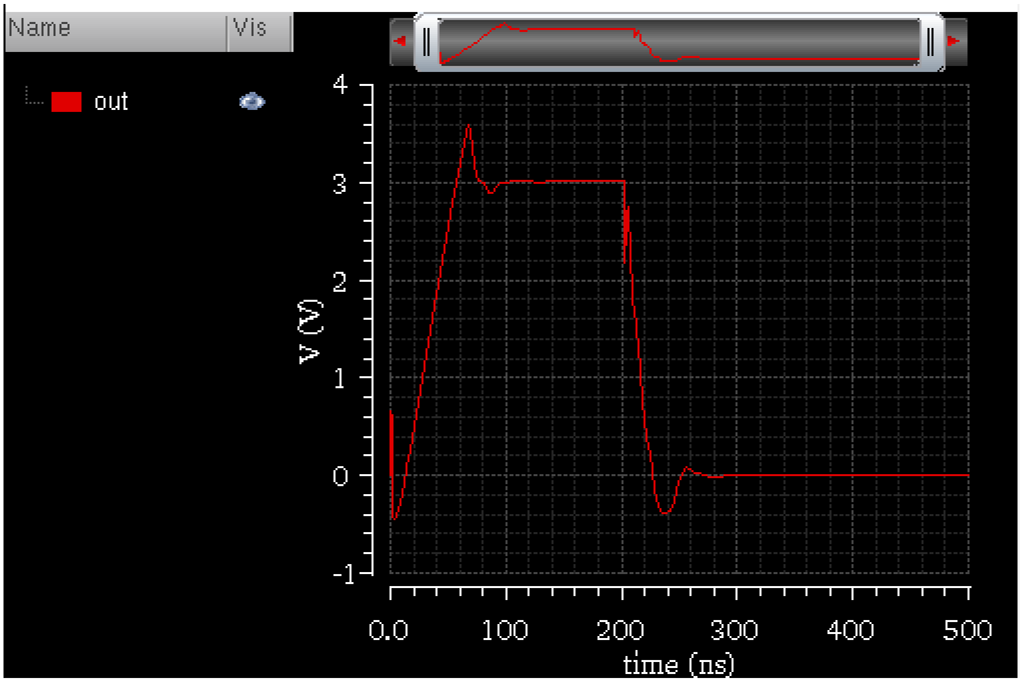
  The difference between the selected signals is plotted in a graph window.

The following figures illustrate an example:

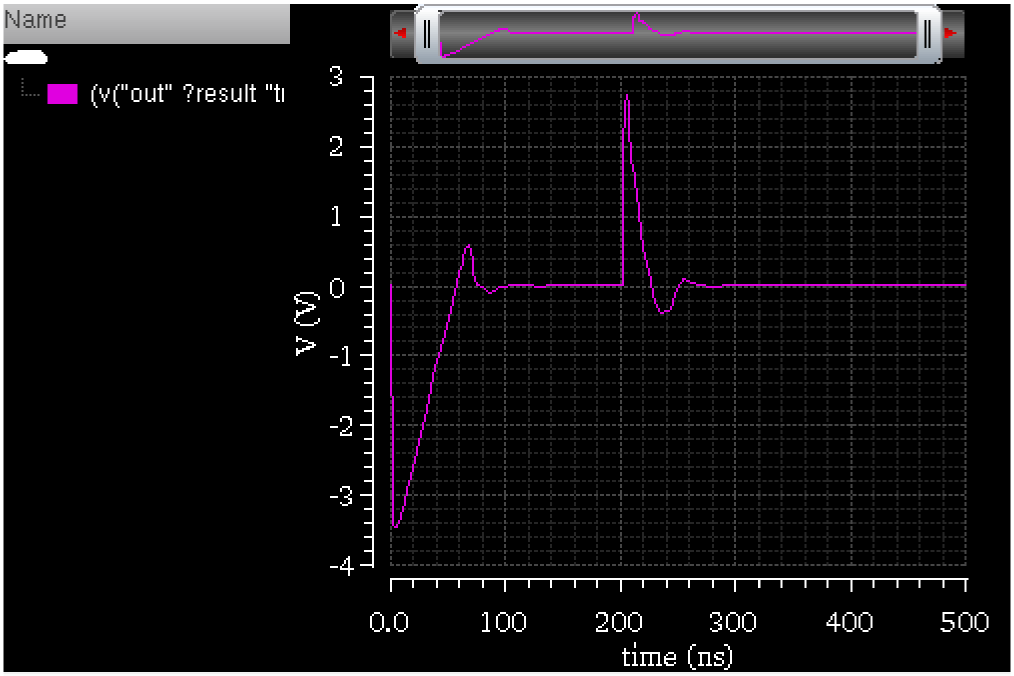
Signal `in_p`:



Signal `out`:



Difference `out` and `in_p`:



Comparing Signals

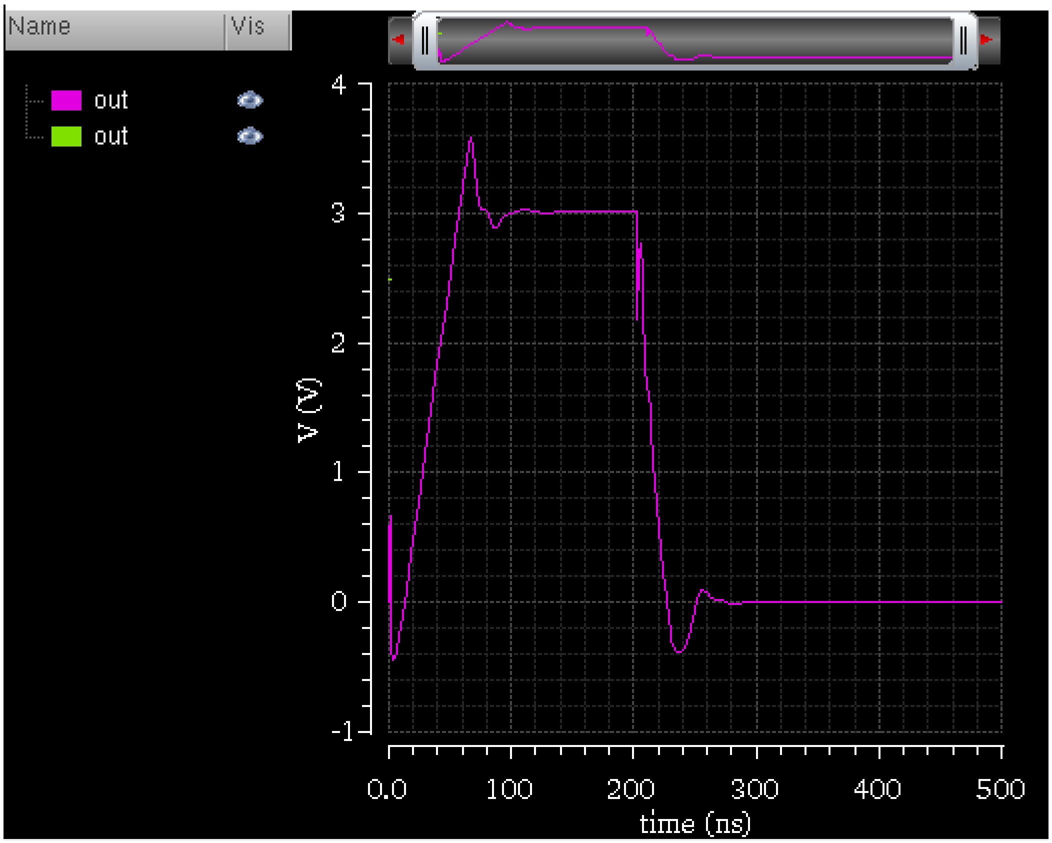
If you run multiple simulations on the same data type, multiple results directories containing the same signals with different values are generated.

You can open these results directories in the Results Browser to compare the signals contained in the results directories. The signals from all the results directories are plotted in a graph window in the append mode to facilitate comparison

You can use one of the following two ways to compare signals:

- Open the respective results directories and plot the signals, one by one, in the append mode in a graph window. This is a time consuming task.
- Right-click the signal in a results directory and choose the *Plot from all DBs*, or select a signal in a results directory and click  on the Results Browser.

The figure below shows the `out` signal plotted from two different results directories. The two signals are overlapped because they have same data values.



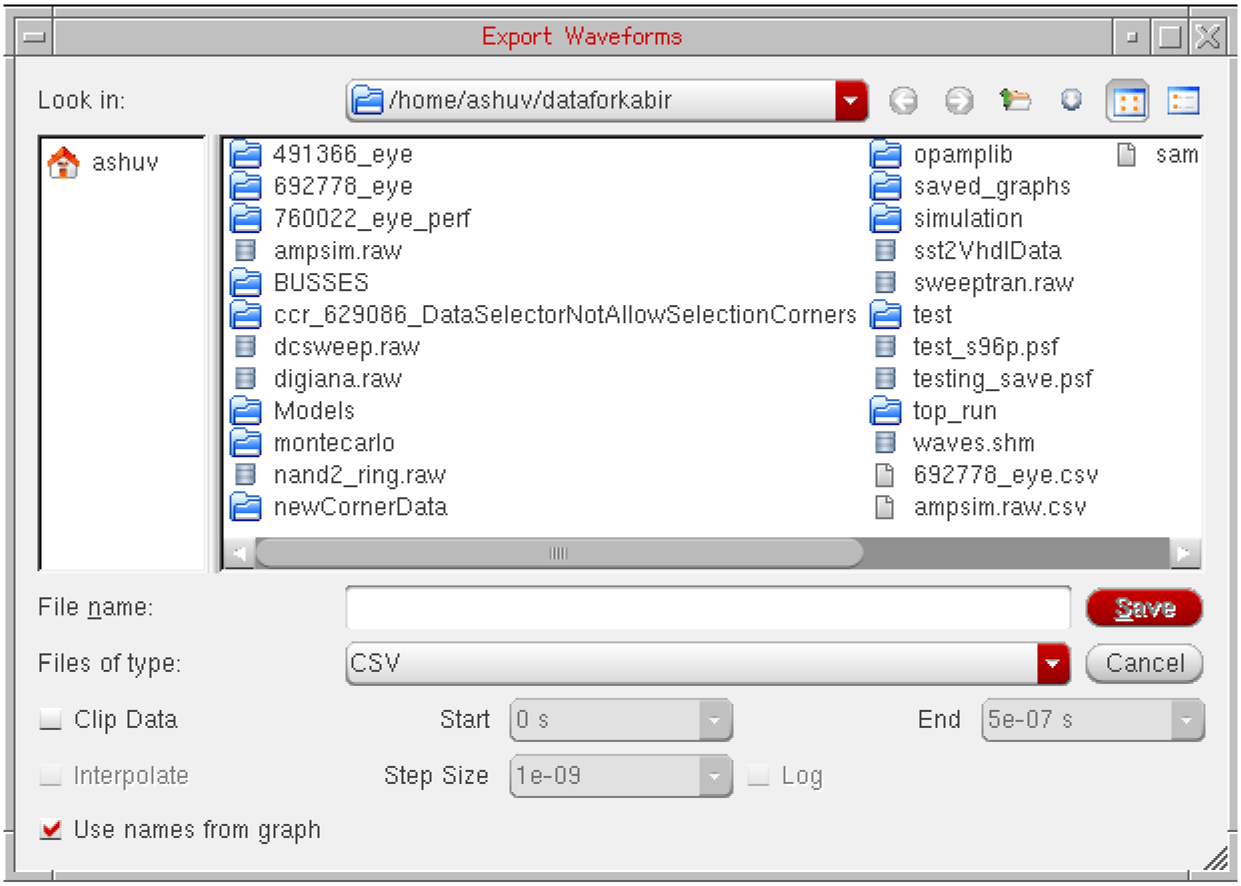
Exporting Signals

You can export signals from a results directory in a variety of formats and later load these signals in the required application. You can also save a part of the dataset by specifying the start and end values, or interpolate the data before saving it.

To export signals from a results directory, do one of the following:

- Right-click a signal in the Results Browser and choose *Export*. You can select multiple signals by holding down the `Ctrl` key while you click the required signals.
- In the Results Browser, select a signal, and then in the Virtuoso Visualization and Analysis XL window, choose *Browser – Results – Export*.

The *Export Waveforms* form appears. This form has the similar settings as of the *Select Waveform Database* form. In both forms, the left pane includes the links to the home directory and the current working directory.



In this form, do one of the following to export the selected a signal:

- In the *Look In* field, browse to locate the path of the data directory where you want to save the selected signal. By default, this field displays the path of the directory from where the you opened the results recently in the Results Browser.

  The form includes the following three buttons in the upper-right corner:

  - *Parent Directory*—To open the parent directory. This button is available if you have nested results directories.
  - *Descend Inside Directory*—To view the contents of the parent directory and to move to the next-level directory.
  - *List View*—To display directories in a list view.
  - Detail View—To display directories in the detail view.

- ❑ In the *File name* field, specify a file name.
- ❑ In the *Files of type* field, select the file type. The file type (indicated by the file extension) determines the format in which the signal is saved. You can also specify the file type with the file name.

  The supported file types are as follows:

  - ❍ `CSV` (`Comma Separated Value)` format allows saved traces to be imported to spreadsheet and other tools.
  - ❍ `Matlab` format allows saved traces to be imported into Matlab. The Matlab format can be imported to Matlab by using the import wizard.
  - ❍ `SPECTRE` (`Spectre Input)` format allow saved traces to be used as inputs to the Spectre PWL input voltage or current sources.
  - ❍ `PSF` (`Parameter Storage Format`) format is available for use only in the Virtuoso Visualization and Analysis XL tool. The PSF format does not support digital data.
  - ❍ `SST2` format.
  - ❍ `VCSV` (`Visualization & Analysis File)` format (default) allows you to save traces that can be loaded from within the Virtuoso Visualization and Analysis XL tool for use in the future. You can also add comments to the VCSV file after the standard header line.

  Note: When multiple signals are selected, the signals are placed in a directory specified that you specify in the *Filename* field with each signal saved in a separate file.
- ❑ Select the *Clip Data* check box if you want to export a clip in the signal. Specify the clip start and end values in the *Start* and *End* fields.
- ❑ Select the *Interpolate* check box, if you want to export interpolated data. In the Step Size field, specify the step size. Select *Log* if you want to save the value by using a logarithmic scale.

  Note: Do not specify the step size in the *Step* field when the *Log* check box is selected.
- ❑ The *Use names from graph* check box is disabled when you export a signal from the Results Browser. If you export a trace from the graph, you can select this check box to use the trace names that are displayed on the graph. When this check box is not selected, you can enter any name
- ❑ Click *Open.*

The selected signals are exported to the location that you specified.

Using the Calculator

You can use the Calculator to evaluate signals stored in a results directory in the Results Browser. To load the signal from the Results Browser to the Calculator, do one of the following:

- Right-click a signal and choose Calculator.
- Select a signal and click the  button.

  The Calculator window appears and the selected signal is displayed in the Buffer.

For more information, see Chapter 4, "Working with the Calculator."

Supporting s-parameters

The Virtuoso Visualization and Analysis XL supports the industry standard Touchstone format that can read the data files created by any simulator. You can use the touchstone s-parameter file to plot the s-parameter and the L, Q, and R data. This format also helps plot nport s-parameter data if you use Spectre and SpectreRF as simulators.

The touchstone file is an ASCII file, also known as the SnP file, which includes a large signals S-parameter results. The touchstone files are of `.snp` extension, where n is the number of network ports of the device. For example, if the touchstone file contains the network parameters for a two port device, it has `.s2p` extension.

3

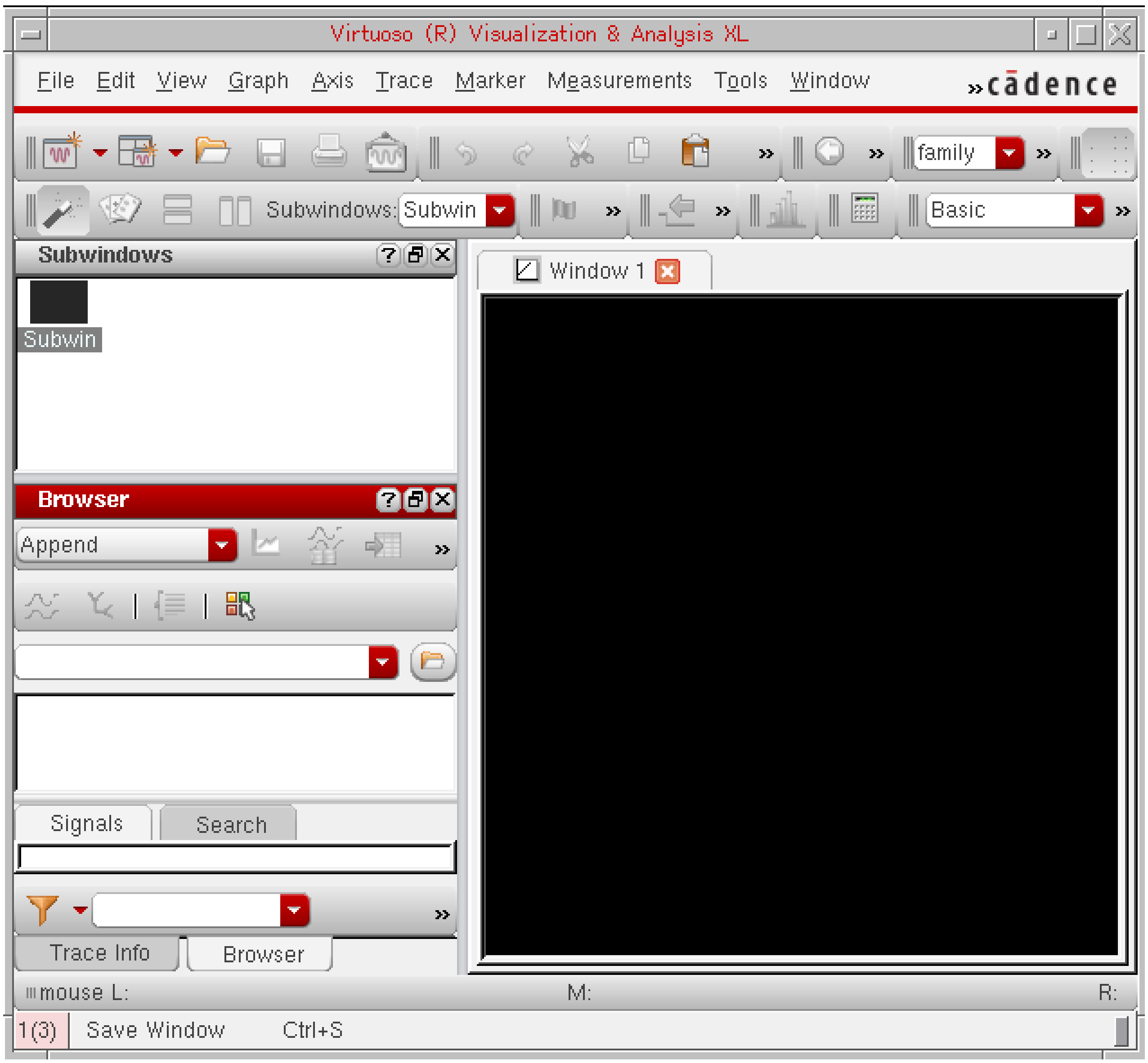
Working with Graphs

The Virtuoso Visualization and Analysis XL graph window is a tool that you use to present the simulation data in a graphical format.

This chapter includes the following topics:

About the Graph Window

Virtuoso Visualization and Analysis XL graph window is a tool that you use to present simulation data in a graphical format. This helps you analyze simulation results. The ability to plot multiple graphs at a time enables you to compare simulation results. You can also customize your graphs by changing the background color and layout, and add markers and labels to annotate the graphs.



When you run the tool for the first time in a new session, a default graph is opened in a tab named *Window 1*. You can rename the window tab names by double-clicking the tab name or by setting the `viva.graphFrame` .cdsenv variable. You can also close the tabs that are not required. In a window, you can open multiple subwindows.

The graph window terminology is explained below:

- Window—Window is the plotting area where you plot signals and open multiple subwindows. A window consists of a trace legend area, X-axis zoom and pan bar, dependent and independent axes, and graph objects. The window names appear on different window tabs, which means when you open a new window, it is opened in a new tab. You can also close and rename the windows if required.
- Subwindow or Graphs—A window can be divided into multiple subwindows, which include all properties of a window, such as trace legend area, pan bar, dependent and independent axes, and graph objects. The subwindow names are displayed in the Subwindows assistants or in the subwindows drop-down list on the Graph toolbar. By default, a new graph window always has one subwindow defined, which is listed in the Subwindows assistant. For example, *Subwin(1)* shown in the figure above. For more information about subwindows, see Subwindows on page 171.
- Strip—A subwindow can be further divided into different strips that include one or more signals. All the strips in a subwindow share the same independent axis, and X-axis zoom and pan bar.
- Traces, markers, labels, and axes are the graph objects that can be inserted in a window, subwindow, or a strip.

In the graph window tool, you can also choose the assistants you want to display, which is defined by the selected workspace. The default workspace is *Classic*. For more information about Workspaces, see Working with Workspaces on page 173.

You can open the graph window from Virtuoso Visualization and Analysis XL or from the Analog Design Environment (ADE). If you open the graph Window from the Virtuoso Visualization and Analysis XL, you can work with previously saved simulation data. However, if you open the graph window from ADE, you work with the simulation data for the latest run. In both cases, when you select a signal, the graph window appears with the selected signal plotted.

Opening the Graph Window

You can use the following methods to open the graph window:

- Opening the Graph Window from Virtuoso Visualization and Analysis XL on page 71
- Opening the Graph Window from Virtuoso in Stand-Alone Mode on page 71

- Opening the Graph Window from ADE L and ADE XL on page 71

Opening the Graph Window from Virtuoso Visualization and Analysis XL

You can open the graph window in either SKILL or MDL mode from the Virtuoso Visualization and Analysis XL tool. By default, the graph window is opened in the SKILL mode.

Perform the following steps to open the graph window from Virtuoso Visualization and Analysis XL tool:

➡ Start the Virtuoso Visualization and Analysis XL tool by typing the following command in a terminal window:

```
viva -expr skill &
```

The *Virtuoso (R) Visualization and Analysis XL* appears.

Note: If you type only `viva &` in the terminal window, the default mode is SKILL.

For information about how to create a graph in the graph window, see Creating a Graph on page 94.

Opening the Graph Window from Virtuoso in Stand-Alone Mode

To open the graph window from Virtuoso in the stand-alone mode, perform the following step:

- From the CIW, choose *Tools – Analog Environment – Waveform.*

  The *Virtuoso (R) Visualization and Analysis XL* appears.

Opening the Graph Window from ADE L and ADE XL

You can run simulations in ADE L and ADE XL and plot the simulation results in the Virtuoso Visualization and Analysis XL graph window. The Virtuoso Visualization and Analysis XL graph window supports simulation analysis types such as transient, AC, DC, and RF measurement.

To open the graph from ADE L, do the following:

➡ In the ADE L window, choose *Tools – Waveform*.

The *Virtuoso Visualization and Analysis XL* appears.

The graph window also appears after a simulation is run in ADE L, displaying the output signals from the selected analysis types.

In ADE XL, after you run the simulation, you can specify whether you want to save the simulation results to a results database or plot the simulation results in a window. Each item that appears on the ADE XL *Outputs Setup* tab has a *Plot* check box and a *Save* check box. Select the *Plot* check box to display the selected outputs in the window after the simulation run is complete. Select the *Save* check box to save the selected output results to a results database.

Notice the following when you open the window by using ADE L and ADE XL:

- The graph window tab names in Virtuoso Visualization and Analysis XL correspond to the test names in ADE L and ADE XL.
- The subwindow titles display the measurement or analysis names.
- The subwindow titles also display the simulation time or measurement evaluation time.

Graphical User Interface

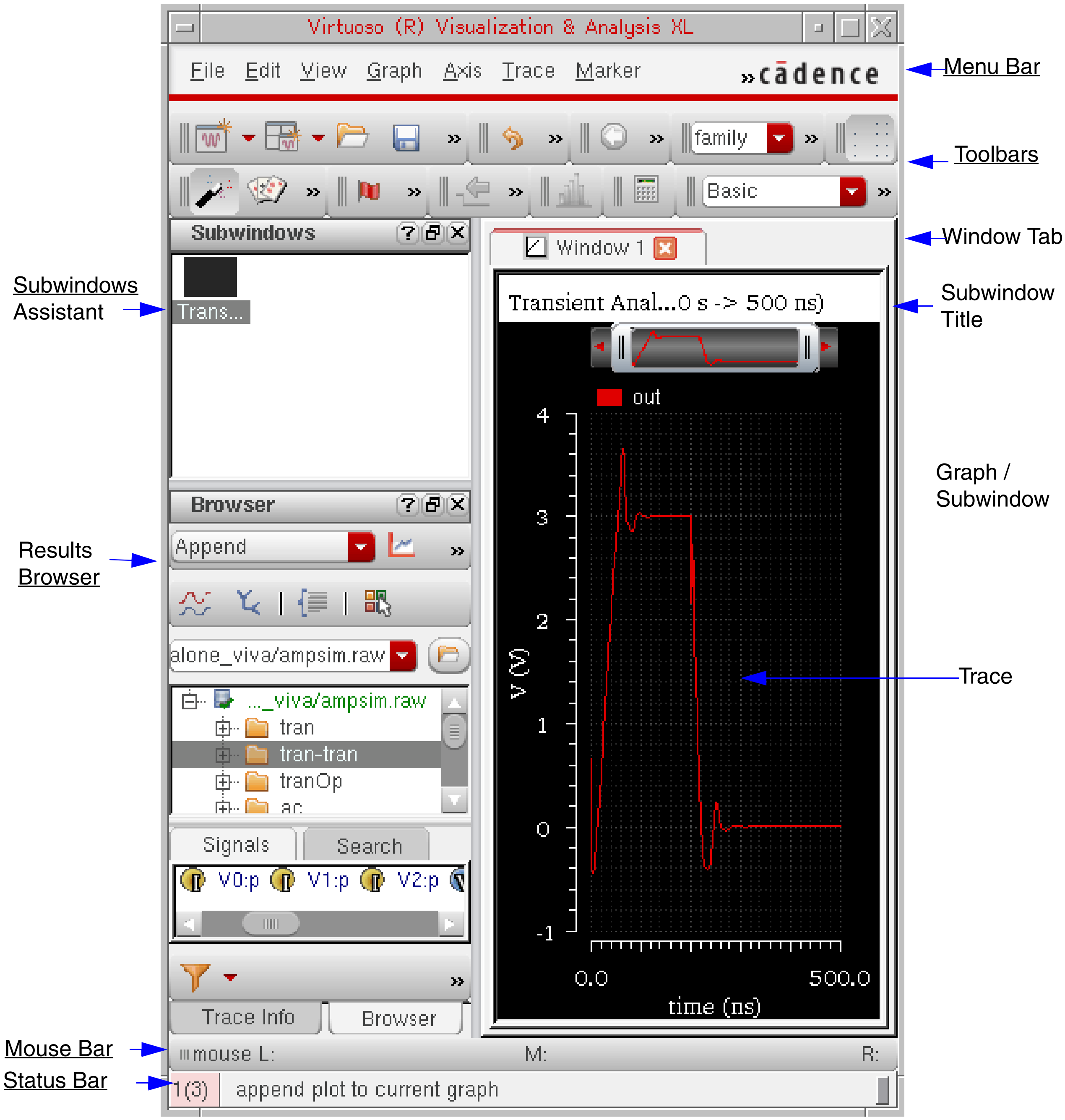
The Virtuoso Visualization and Analysis XL window user interface consists of a menu bar, toolbars, dockable assistants, and subwindows that are displayed according to the selected workspace. You can hide and show these GUI components based on the workspace you select. By default, the assistant panes appear on the left and the graphs appear in the display area on the right. When you plot a signal in the new window, a new window tab is created. You can click the tab to view the required graph and can rename window tabs by double-clicking the tab name. You can also close the window tabs that are not required.

In a window, you can open multiple subwindows. The subwindows includes all properties of a graph window and can be further divided into subwindows.

Note: By default, the window background is black. If required, you can set the background to white. For more information, see Setting the Graph Colors on page 103.

The Virtuoso Visualization and Analysis XL window includes the following elements:

- Menu Bar on page 73
- Toolbars on page 87
- Status Bar on page 93
- Assistants on page 93



Menu Bar

The menu bar has the following menus:

- File on page 74

File

The table below lists the *File* menu commands.

Command	Description
Open Results	Opens the *Select Waveform Database* form that you can use to select a results database. This form displays the current results directory.
Close Results	Closes the selected results directory in the Results Browser.
New Window	Opens a new graph window in the display area. This creates a new window tab.
New Subwindow	Opens a new subwindow in the active window.
Load Window	Loads a file containing a graph in the active window. For more information, see Loading a Graph on page 212.
Save Window	Saves the graph in the active window to a file. For more information, see Saving a Graph on page 210.

Command	Description
Save Window As	Saves a copy of the graph in the active window to a file. For more information, see Saving a Graph on page 210.
Reload	Updates data for all the traces in the active subwindow or all the subwindows. For more information, see Reloading Graphs on page 220.
Print	Sends the graph to a printer or saves the graph in a PDF or Postscript format. For more information, see Printing Graphs on page 228.
Save Image	Saves the graph displayed in the active window as an image file. For more information, see Printing Graphs on page 228.
Close Window	Closes the active window. If there is only one open window, the *Close* command exits the tool.
Close All Windows	Closes all windows and exits the tool.

Edit

The table below lists the *Edit* menu commands.

Command	Description
Undo	Undoes the most recent action in the active window.
Redo	Redoes the most recent action in the active window.
Cut	Moves the selected graph objects to the clipboard.
Copy	Copies the selected graph objects to the clipboard.
Paste	Pastes the contents of the clipboard to the selected location.

Command	Description
Delete	Deletes selected objects, such as labels, markers, traces, or graphs. **Note:** If no object is selected, a message appears confirming the deletion of the active subwindow. If only one window is open, the *Delete* command is not available.
Delete All	Deletes all the objects and graphs in the active window. If only one window is open, the *Delete All* command is not available.
Properties	Enables you to modify the properties of the recently selected graph object. By default, the graph properties form appears if you do not select any object.

Important

You can cut, copy, and paste between multiple Virtuoso Visualization and Analysis XL sessions within the same Virtuoso process. You can move entire graphs in this manner, or individual traces along with the associated markers. The clipboard contents are retained even when a Virtuoso Visualization and Analysis XL session is closed and another new session is invoked from same Virtuoso process.

View

The table below lists the *View* menu commands.

Command	Description
ZoomIn by 2	Zooms in the graph by a factor of two.
ZoomOut by 2	Zooms out the graph by a factor of two.
Fit	Returns the graph to the actual size to fit data in the window. This command works for both rectangular and circular graphs.

Command	Description
Previous	Enables you to view the graph at the magnification specified before the last zoom in or zoom out command was run. You can use this option when you zoom in or out a graph multiple times.
Next	Undoes the *Previous* command. You can use this option when you zoom in or out a graph multiple times. **Note:** The *Next* and *Previous* commands help you navigate through the zoom and the pan stack.
Fit Trace	Returns the selected trace to its actual size to fit in the window. When you select this option, the X-axis zoom of all the strips is displayed in the actual size, where as the Y-axis zoom of only selected axis is changed. This command works for both rectangular and circular graphs.
Fit Y to Visible X	Fits the visible part of the trace to Y-axis. This command finds the minimum and maximum Y-axis values that are visible in a strip and then performs a Y-axis zoom of those Y values. This command works only for the zoomed-in graphs.
Fit Smith	Returns the selected Smith chart to its actual size so that it fits into the window. This command is available only for the circular graphs.

Graph

The table below lists the *Graph* menu commands.

Command	Description
Layout	Specifies how subwindows are displayed in the active window. You can select the layout as *Auto*, *Vertical*, *Horizontal*, and *Card*. For more information about graph layouts, see Specifying the Graph Layout on page 98.

Command	Description
Add Label	Adds a label to the graph. For more information about graph labels, see Working with Graph Labels on page 201.
Lock	Locks the graph from any data updates. For more information, see Locking Graphs on page 132.
Visible	Shows or hides the active graph.
Split Current Strip	Splits the graph into as many strips as there are traces and displays each trace in the graph in a separate strip. You can also select this command from the Strip toolbar. For more information, see Working with Strips on page 185.
Split All Strips	Splits the traces in all the strips in the graph into individual strips. This is useful if the graph contains more than one strip.
Plot to New Strip	Plots the selected trace in a new strip.
Combine All Analog Traces	Combines all the individual analog traces into a single graph. For more information, see Combining Graph Strips on page 188.
Filter By Sweep Var	Displays the traces for the selected sweep variable range.
Redraw	Refreshes the graph and plots the updated graph in the same window. This command also refreshes the trace legend area.
Toggle Major and Minor Grids	Displays or hides the major and minor grids in the selected axis. Alternatively, you can use bindkey `G` to toggle between the major and minor grids. To use this bindkey, ensure that an axis is selected.
Properties	Sets the graph properties. You can set the general graph properties as well as the strip properties in the *Graph Properties* form that appears when you select *Properties*. For more information, see Editing Graph Properties on page 108.

Axis

The table below lists the *Axis* menu commands.

Note: These commands are available only if you select an axis in the graph.

Command	Description
Major Grids	Displays the major grid lines for the selected X or Y axis.
Minor Grids	Displays the minor grid lines for the selected X or Y axis.
Log	Displays the logarithmic scale for the selected X or Y axis.
Select Attached Traces	Selects all the traces that are attached to the axis you select.
Y vs Y	Displays the YvsY plot of the selected axis in the window. This command is available only for the sweep data. For more information, see Plotting YvsY Graph on page 207.
Swap Sweep Var	Enables you to swap sweep variables. This command is available only if you select the sweep data. For more information, see Swapping Sweep Variables on page 191.
Properties	Sets the attributes for the selected X or Y axis. For more information, see Editing Graph Axis Attributes on page 111.

Trace

The table below lists the *Trace* menu commands.

Command	Description
Symbols On	Displays symbols on individual data points for the selected trace. **Note:** This command is available only if you select one or multiple trace in the graph.

Command	Description
Select by Family	Selects all the traces with the parametric sweep data that belong to a family. When you enable this command, and select a trace in the family, all traces that belong to the same family are selected.
Strip by Family	Displays traces that belong to the same family in a single strip when you split traces into strips. If more than one family of traces are present, each family is displayed in a separate strip.
Fit Trace	Returns the selected trace to its actual size to fit in the window. When you select this option, the X-axis zoom of all the strips is displayed in the actual size, where as the Y-axis zoom of only selected axis is changed.
Fit Y to Visible X	Fits the visible part of the trace to Y-axis. This command finds the minimum and maximum Y-axis values that are visible in a strip and then performs a Y-axis zoom of those Y values. This command works only for the zoomed-in graphs.
Disable Reload	Disables the reloading of a trace by locking the database context and the trace is not reloaded with new data when the in-context results directory is changed. For more information, see Disabling Trace Reload on page 226.
Select All	Selects all traces in a graph.
Delete All	Deletes all the traces displayed in the active graph. The independent axis, window title, and pan bar are not deleted.
Move to	Moves the selected trace to the following locations: ■ *New Window*—Moves the selected trace to a new window. ■ *New Subwindow*—Moves the selected trace to a new subwindow. ■ *New Strip*—Moves the selected traces to a new strip.

Command	Description
Copy to	Copies the selected trace to the following locations: ■ *New Window*—Copies the selected trace to a new window. ■ *New Subwindow*—Copies the selected trace to a new graph subwindow. ■ *New Strip*—Copies the selected trace to a new graph strip.
Bus	This command has the following options: *Create*—Creates a bus from the selected digital traces. For more information, see Creating a Bus on page 245. *Expand*—Expands a bus to its component signals. For more information, see Expanding a Bus on page 248. **Note:** This option is available only if you select a digital bus. *Collapse*—Collapses the bus components to display the complete bus. **Note:** This option is available only if you expand the bus.
Export	Exports the selected trace in the active window in a variety of formats and later loads it in the required application.
Properties	Enables you to specify properties of the selected trace. For more information, see Setting Trace Properties on page 180. **Note:** This option is available only if you select a trace in the graph.

Marker

The table below lists the *Marker* menu commands.

Command	Description
Tracking Cursor	Enables or disables the tracking cursor for the graph. When you move the mouse pointer on a trace or on a graph object, the tracking cursor displays the trace name and the graph object information.
Snap Tracking Cursor	Snaps the tracking cursor to the simulation points. When you move the mouse pointer on the simulation points on a trace, the tracking cursor displays the trace name and the graph object information
Create Marker	Creates a new marker for the trace in the graph. For more information, see Adding Markers on page 251.
Create Delta Marker	Creates a new delta marker. To create a delta marker, you need to place a point marker on the trace or select an existing point marker. For more information, see Adding AB Marker on page 273.
Show Delta Child Labels	Shows or hides marker labels for the delta markers.
Delete all	Deletes all the markers displayed in the active subwindow or a graph.
Export Table	Exports the selected marker information in a given format.
Properties	Specifies the properties for a marker. For more information, see Setting Marker Properties on page 258. **Note:** This command is available only if you select a marker on the graph.

Measurement

The table below lists the *Measurement* menu commands.

Command	Description
Eye Diagram	Plots an eye diagram for the selected graph. The eye diagram divides the waveforms into fixed time periods, which are then superimposed on each other. When you select this command, the Eye Diagram assistant appears. For more information, see Eye Diagram on page 144.
Spectrum	Plots a spectrum for the selected graph. When you select this command, the Spectrum assistant appears. For more information, see Spectrum on page 134.
Analog to Digital	Converts an analog signal into a corresponding digital signal. For more information, see Converting a Digital Signal to an Analog Signal on page 236. **Note:** This command is available only in the SKILL mode.
Digital to Analog	Converts a digital signal into a corresponding analog signal. For more information, see Converting an Analog Signal into a Digital Signal on page 234. **Note:** This command is available only in the SKILL mode.
Derived Plots	Generates the derived plots that are the risetime or falltime waveforms derived from the entire set of edges and plotted against time. For more information, see Generating Derived Plots on page 238.
Histogram	Generates the histogram plot directly on a graph. For more information, see Plotting Histogram on page 240.
Transient Measurement	Opens the Transient Measurement assistant that displays the calculated measurements for the transient markers on specific edges. For more information, see Transient Measurement on page 155.

Tools

The table below lists the *Tools* menu commands.

Command	Description
Calculator	Opens the Virtuoso Visualization and Analysis XL Calculator window. For detailed information about working with the Calculator, see Chapter 4, “Working with the Calculator.”

Window

The table below lists the *Window* menu commands.

Command	Description
Assistants	Displays or hides the selected assistant panes. The available assistants are—Spectrum, Browser, Marker Toolbox, Eye Diagram, Horiz Marker Table, Trace Info, Vert Marker Table, Customize Trace Groups, and Subwindows. For more information, see Assistants on page 93. For more information about assistant panes, see the *Virtuoso Design Environment User Guide*.
Workspaces	Displays, saves, loads, and configures the selected workspace. The available workspaces are—`B`asic, Browser, Classic, and MarkerTable. For more information, see Working with Workspaces on page 173. For more information about workspaces, see “Getting Started with Workspaces” in *Virtuoso Design Environment User Guide.*

Command	Description
Toolbars	Displays or hides the selected toolbars. The available toolbars are—Edit, View, Graph, Calculator, Snap, Marker, Strip, Measurement, Axis, and Workspaces. For more information about toolbars, see Toolbars on page 87.

Browser

The table below lists the *Browser* menu commands.

Command	Description
Results	Includes the following Results Browser commands: ■ *Open Results*—Opens the results directory in the Results Browser. When you select this command, the *Select Waveform Database* form appears that you can use to select the results database. ■ *Export*—Exports a selected signal from the Results Browser. ■ *Close Results*—Closes the results directory in the Results Browser. This is available only if you select a results directory in the Results Browser. ■ *Reload*—Reloads the results directory that was last open into the Results Browser. ■ *Set Context*—Enables you to set the database in-context results directory, which you use to plot signals in the Results Browser. The first results directory that you load in the Results Browser is set as the in-context results directory.

Command	Description
Options	Includes the following commands: ■ *Graph Modifier*—Includes commands that you can use to specify how the graph is plotted: ❑ *Magnitude*—Plots magnitude versus frequency. ❑ *Phase*—Plots phase versus frequency. ❑ *WPhase*—Plots wrapped phase versus frequency. ❑ *Real*—Plots real value of the dependent data versus frequency. ❑ *Imaginary*—Plots imaginary value of the dependent data versus frequency. ❑ *dB10*—Plots *dB10* value of the dependent data versus frequency. ❑ *dB20*—Plots *dB20* value of the dependent data versus frequency. ❑ *dBm*—Plots mili dB value of the dependent data versus frequency. ■ *Plot Style*—Enables you to select the mode in which a graph is to be plotted. The signal in the graph can be plotted in the following modes: ❍ *Append*—Adds the signal to the selected graph. ❍ *Replace*—Replaces the signal in the selected graph with the new signal. ❍ *New Subwindow*—Plots the signal in a new subwindow within the active window. ❍ *New Window*—Plots the signal in a new window. ■ *Select Data*—Sets the sweep range for the data. When you select this command, the *Set Sweep Ranges* form appears. This command is available only if the dataset selected in the Results Browser supports ranging, such as the PSF transient dataset, or contains parametric sweep data.

Command	Description
	■ *Enable Fast Waveforms*—Enables the fast waveform format in which the Virtuoso Visualization and Analysis XL tool can render extremely large datasets within seconds.

Help

The table below lists the *Help* menu commands.

Command	Description
Contents	Displays the *Virtuoso Visualization and Analysis Tool User Guide.*
Cadence Online Support	Displays the Cadence customer support website in your default Web browser.
Online User Forum (cdnusers.org)	Displays the online users' forum website in your default Web browser
Known Problems and Solutions	Displays *Virtuoso Visualization and Analysis Tool Known Problems and Solutions.*
What's new	Displays *Virtuoso Visualization and Analysis Tool What's New.* ■ *In this Release*—Displays *Virtuoso Visualization and Analysis Tool What's New.* ■ *Videos*—Displays the What's New videos. ■ *Overview*—Displays the Virtuoso Visualization and Analysis XL Overview window.
About Visualization and Analysis	Displays the version number of the Virtuoso Visualization and Analysis XL tool.

Toolbars

Do one of the following to show or hide toolbars in Virtuoso Visualization and Analysis XL:

- Choose *Windows – Toolbars – Toggle Visibility*.

- Right-click anywhere in the menu bar and select the toolbars that you want to show.
- Press the bindkey `Ctrl+F11` to toggle the visibility of toolbars and press the bindkey `Shift+F11` to toggle the visibility of toolbars and assistants.

The Virtuoso Visualization and Analysis XL has the following toolbars:

- Mouse Bar on page 88
- Edit Toolbar on page 88
- View Toolbar on page 89
- Graph Toolbar on page 89
- Calculator Toolbar on page 90
- Snap Toolbar on page 90
- Marker Toolbar on page 90
- Measurement Toolbar on page 91
- File Toolbar on page 91
- Strip Toolbar on page 92
- Axis Toolbar on page 92
- Workspace Toolbar on page 93

Mouse Bar

Displays at the bottom of the Virtuoso Visualization and Analysis XL window to indicate the left, middle, and right mouse movements.

Edit Toolbar

The Edit toolbar contains the following buttons:

- Undo
- Redo
- Cut

- Copy
- Paste
- Delete

For information about these toolbar buttons, refer to the Edit menu commands.

View Toolbar

The View toolbar contains the following buttons:

- Previous
- Next
- Fit
- ZoomIn by 2
- ZoomOut by 2
- Fit Trace
- Fit Y Visible
- Fit Smith

For information about these toolbar buttons, refer to the View menu commands.

Graph Toolbar

The Graph toolbar contains the following icons:

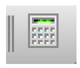
- *Layout Icons*—Shows the graph layouts that you can use to change the layout of the active window:
  - Auto
  - Vertical

- ❑ Horizontal
- ❑ Card

For more information about graph layouts, see Specifying the Graph Layout on page 98.

- *Subwindows*—Lists all the subwindows that are open in an active window. When you select a subwindow in this list, the selected subwindow is highlighted in the window as well as in the Subwindows assistant. For more information about subwindows, see Subwindows on page 171.

Calculator Toolbar

Displays the Calculator button  to send the selected trace to the Calculator Buffer.

Snap Toolbar

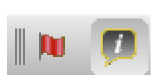
The Snap toolbar contains the following buttons:

- *Previous Edge*—Moves the selected marker to previous edge based on the snapping criteria selected.
- *Next Edge*—Moves the selected marker to the next edge based on the snapping criteria selected.
- *Snapping Criterion*—Displays the criterion based on which the selected marker is snapped.
- *Value*—Displays the value of the snapping criterion.

This toolbar is available for both analog and digital signals and the toolbar options work if you select a marker. For more information, see Snapping Markers on page 263.

Marker Toolbar

The Marker toolbar contains the following buttons:

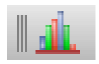


- *Create Marker*—Creates a marker on the selected trace. For more information about how to create a marker, see Adding Markers on page 251.

- *Tracking Cursor*—Turns the tracking cursor on and off for the selected window. For more information, see Tracking Cursor on page 179.

Measurement Toolbar

The Measumerent toolbar includes the following button:



- *Histogram*—Opens the Histogram form that you can use to plot a histogram for the selected signal. For more information, see Plotting Histogram on page 240.

File Toolbar

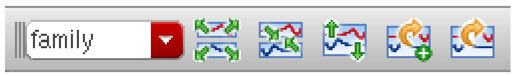
The File toolbar has the following buttons:

- *Create New Window*—Creates a new window. You can choose the type of the window to be created from the drop-down list that includes the following options:
    - *Rectangular*
    - *Polar*
    - *Impedance*
    - *Admittance*
- *Create New Subwindow*—Creates a new subwindow. You can choose the type of the subwindow to be created from the drop-down list that includes the following options:
    - *Rectangular*
    - *Polar*
    - *Impedance*
    - *Admittance*
- *Load Window*—Loads a graph window. For more information, see Loading a Graph on page 212
- *Save Window*—Saves the selected graph window. For more information, see Saving a Graph on page 210.

- *Print*—Prints the graph window. for more information, see Printing Graphs on page 228.
- *Save Image*—Saves the graph window in an image format. For more information, see Saving a Graph as an Image on page 213.

Strip Toolbar

The Strip toolbar contains the following buttons:



- *Strip By*—Specifies how you want to strip the traces in a graph. The available options in the drop-down list are family, leaf, and trace. If you work on the traces from sweep data, the sweep variables are also included in this drop-down as strip options.
- *Combine All Analog Traces*—Displays all analog traces from individual strips to a single g. For more information, see Combining Graph Strips on page 188.
- *Split Current Strip*—Displays the traces in a window in individual strips. For more information, see Working with Strips on page 185.
- *Copy to a New Strip*—Copies the selected trace to a new strip in the same window.
- *Move to a New Strip*—Moves the selected traces to a new strip.

Axis Toolbar

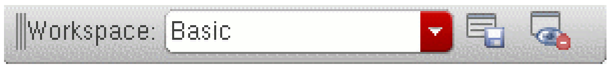
You can use the Axis toolbar to turn on or turn off the grid from a graph.

Alternatively, you can do the following:

- Right click on the graph and select *Toggle Major and Minor Grids* to turn on or off the grids.
- Select an axis and press the bindkey `G`.

Workspace Toolbar

You can use the Workspace toolbar to work with the available workspaces.



For more information about workspaces, see Working with Workspaces on page 173.

Status Bar

The status bar displayed at the bottom of the window displays the following information:

- Warnings and error messages.
- Static information, such as the name of the toolbar button selected in the window.
- Dynamic information, such as the toolbar names are displayed when you perform mouse-hover on toolbars.

Assistants

The Virtuoso Visualization and Analysis XL includes the following assistants:

- Spectrum
- Browser
- Marker Toolbox
- Eye Diagram
- Horiz Marker Table
- Trace Info
- Vert Marker Table
- Subwindows

For detailed information about assistants, see Working With Assistants on page 133.

Creating a Graph

You can create a graph by plotting a signal selected in the Results Browser in the window. To group similar graphs or to compare two graphs, you can open multiple subwindows in a window.

To create a graph, perform the following steps:

1. In the Results Browser, open a results directory and select the signal you want to plot.
2. To select the window where you want to plot the signal, do one of the following:
   - ❑ Choose *Browser – Options – Plot Style.*

   The *Plot Style* can be of the following types:

     - ❍ *Append*—Adds a signal to the selected graph.
     - ❍ *Replace*—Replaces the graph in the active window with a new graph.
     - ❍ *New Window*—Plots the signals in a new window. When you create a graph for the first time, it is always displayed in a new window.
     - ❍ *New Subwindow*—Plots the graph in a new subwindow within the active window.
   - ❑ Right-click the required signal in the Results Browser and choose the destination graph. The plot options that appear on the shortcut menu are the same as those explained above.
3. After you specify the destination graph, do one of the following to plot the signal:
   - ❑ Double-click the signal.
   - ❑ Right-click the signal and select the *Plot Signal* option.
   - ❑ Click the  button in the Results Browser.

   The graph appears in the selected destination window.

Note: You can also drag signals from the Results Browser to Subwindows assistant. If you drop the signal on a selected subwindow icon displayed in the Subwindows assistant, the signal is plotted in the selected subwindow in append mode. However, if you drop the signal anywhere in the Subwindows assistant, the signal is plotted in a new subwindow.

Dragging Graphs Across Multiple Virtuoso Visualization and Analysis XL Sessions

You can drag traces and graphs across different Virtuoso Visualization and Analysis XL opened within the same Virtuoso session. To copy a trace or a group of traces from one Virtuoso Visualization and Analysis XL session to another, select the traces by using the `Ctrl` key and then drag and drop the selected traces in the destination window of the another session. To copy the entire subwindow, drag the subwindow from the Subwindows assistant from one session to the Subwindows assistant of another session. In this case, a new subwindow is created in the destination session.

Note: You cannot drag traces across two Virtuoso Visualization and Analysis XL sessions that are opened from within different Virtuoso processes. In this case, you can export or import graph files to move traces from one session to another. For more information about how to export a trace, see Exporting a Trace on page 184.

While dragging, only the waveform data is copied to the subwindow of the another session. The drag operation does not save trace properties, such as color, linestyle, symbol, the markers and the Y-axis modifiers, such as dB10, dB20, phase.

Limitations

Dragging is not supported in the following cases:

- Traces or graphs generated after swapping the sweep variable
- YvsY plots
- When you drag traces from one session to another session, the database context of the copied waveform does not change, which means the copied waveform has the database context from the source session.

Customizing a Graph

After you have created a graph, you can customize it to analyze the graph data.

This section contains the following topics:

- Handling Graph Objects on page 103
- Panning and Zooming Graphs on page 104
- Editing Graph Properties on page 108

Determining the Active Window

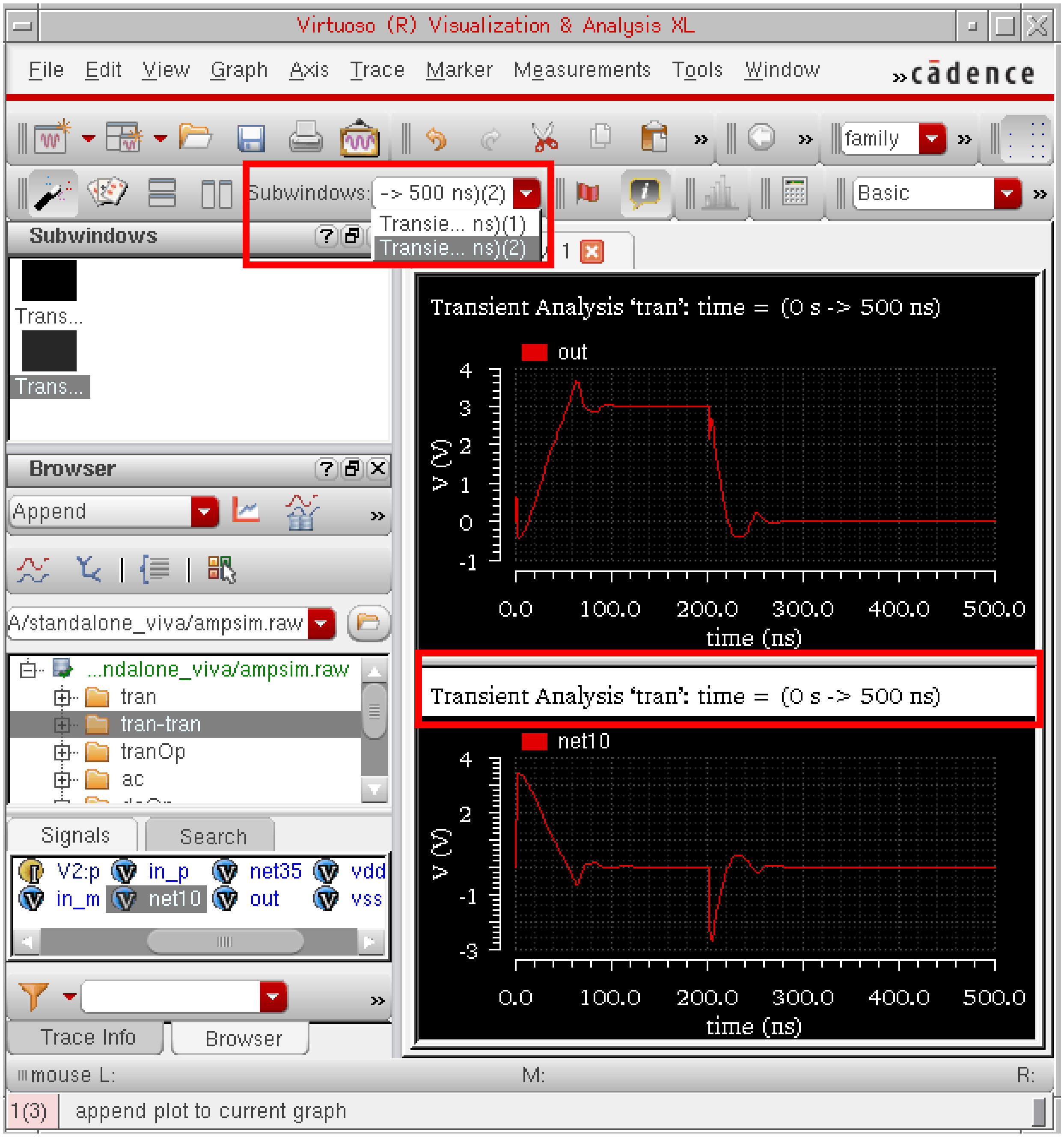
The active window tab appears white and the inactive window tabs appear grey. When you click a window tab, that window becomes active and the tab color changes to white.

Determining the Active Subwindow

A window can contain subwindows. The following features help you determine the active subwindow:

- The name of the subwindow displayed below the thumbnail image in the Subwindows assistant is highlighted in grey.
- The *Subwindows* drop-down list on the Graph toolbar displays the name of the active subwindow.

- In the active subwindow, the title displaying the simulation analysis type and the simulation run date and time. The title font is in contrast with the background.



Note: You can also make a graph active by clicking anywhere in the graph or by dragging a trace to the graph. For more information about dragging traces, see Dragging Traces on page 179.

Specifying the Graph Layout

A window can have several subwindows that are displayed in the specified layout.

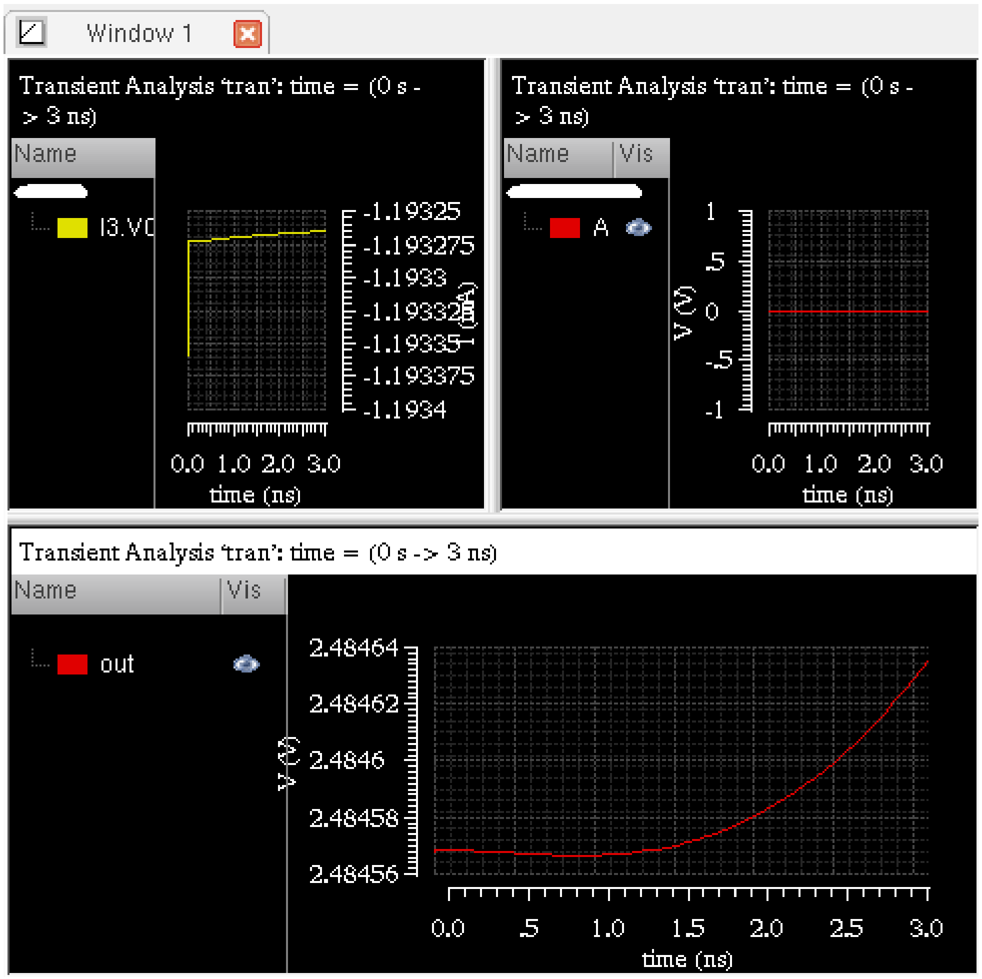
To specify the layout for subwindows, do one of the following:

- Choose *Graph – Layout* and select the layout that you want to apply.
- In the *Layout* drop-down list on the Graph toolbar, select the layout you want to apply.
- Right-click anywhere in the subwindow and choose the required layout from the *Layout* menu.

The following graph layouts are available:

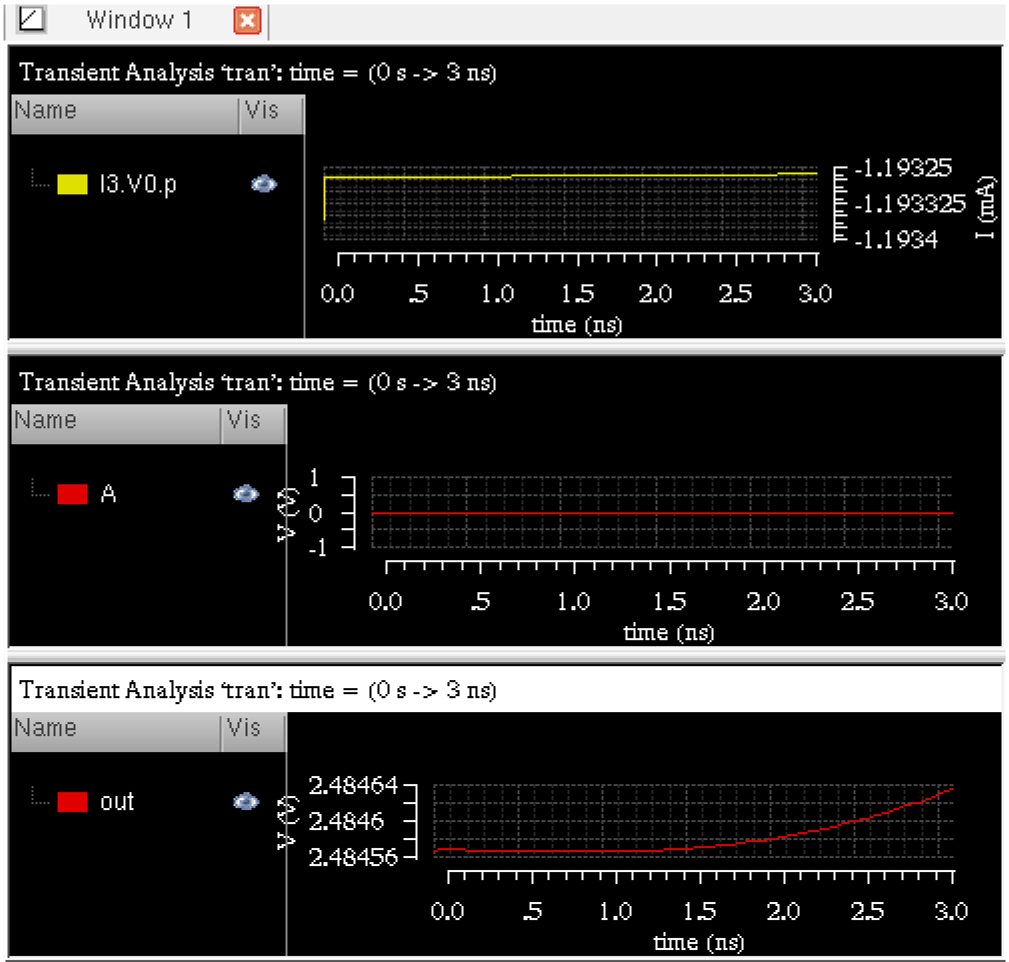
- *Auto*

This is the default layout. In this layout, subwindows are displayed by dividing the active window vertically and horizontally. The aspect ratio determines how the active window is divided. The following figure displays subwindows arranged in the auto graph layout.



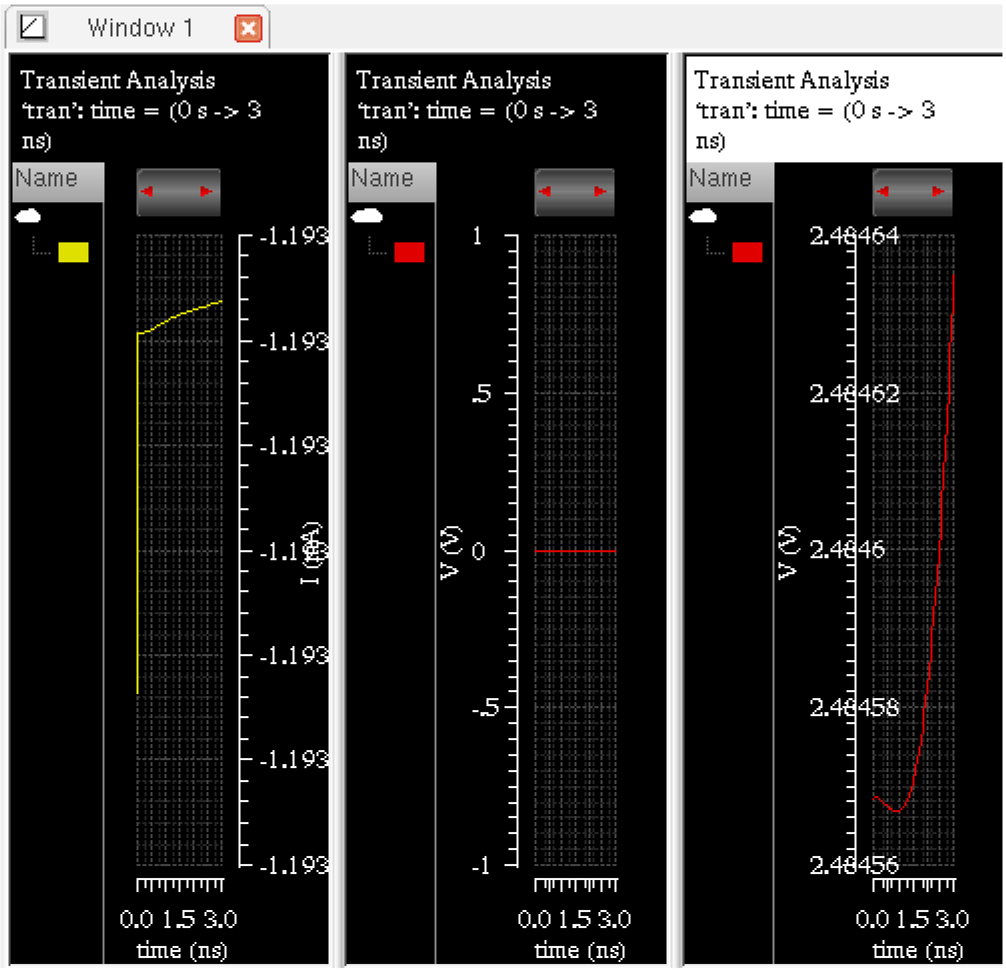
- *Vertical*

  In this layout, subwindows are displayed one below the other in the active window. The following figure displays subwindows arranged vertically.



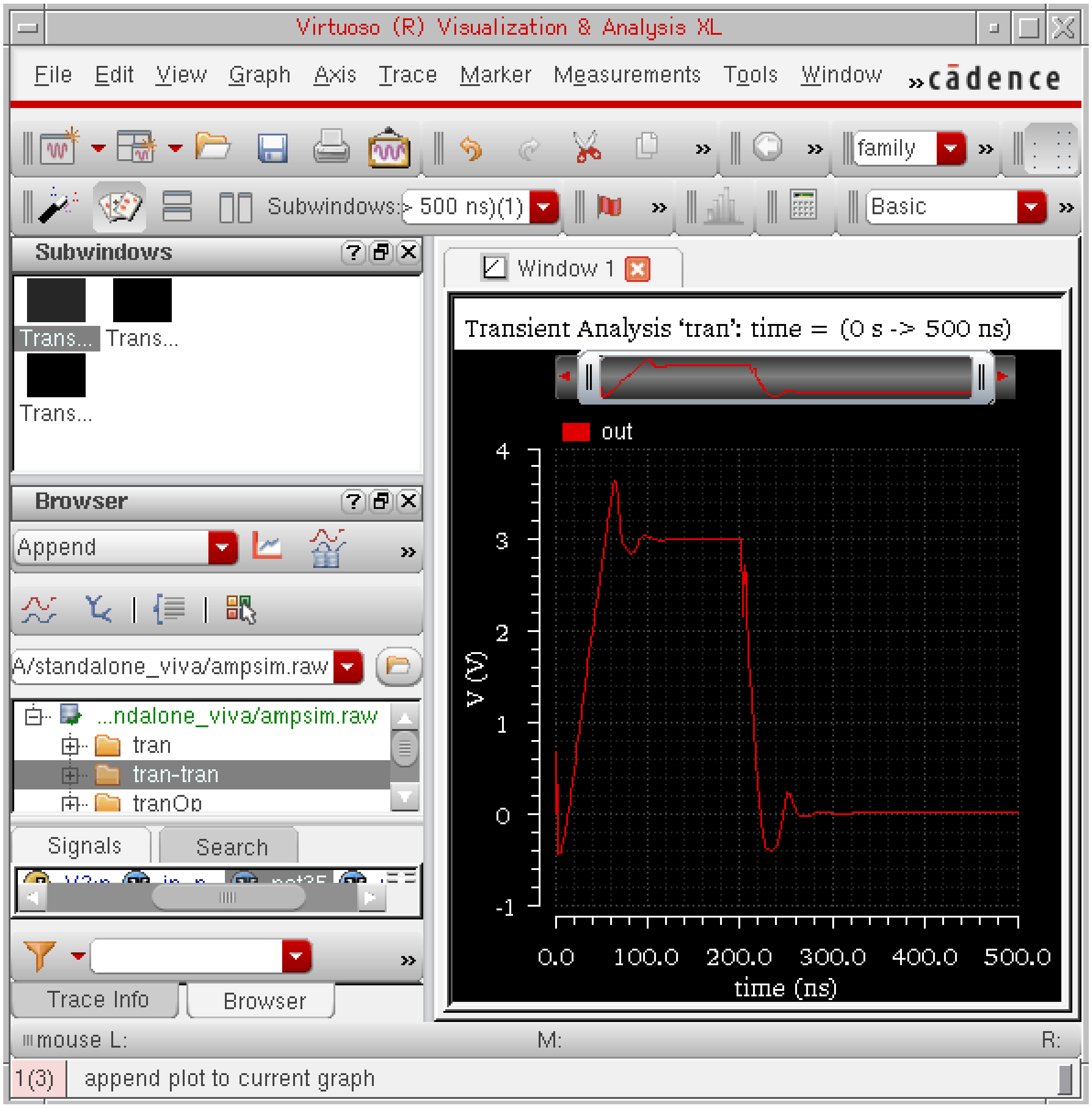
- *Horizontal*

  In this layout, subwindows are displayed side by side in the active window. The following figure displays subwindows arranged horizontally.



- *Card*

  In this layout, subwindows are stacked, like a deck of card, in the active window one on top of the other with only one graph is visible at a time. If you want to view another subwindow, you can select the required subwindow in the Subwindows assistant or in drop-down list on the subwindows toolbar. The following figure displays three subwindows arranged in the card layout.



Video

The video Using Subwindows in Qt Graph demonstrates how to change the graph layout and how to plot signals in different subwindows.

Setting the Graph Colors

The default color scheme for rectangular graphs is determined by the `viva.rectGraph background` and `viva.rectGraph foreground` variables in the `.cdsenv` file. The default color scheme for circular graphs is determined by the `viva.circGraph background` and `viva.circGraph foreground` variables in the `.cdsenv` file. The default color scheme is as follows:

- Black when you open the graph by using Virtuoso Analog Design Environment (ADE) mode or in the stand-alone SKILL mode
- White in the stand-alone MDL mode

To change the background color of a window, do the following:

1. In the window, choose *Graph – Properties.*
2. Click the *General* tab and select a background color in the *Color* field.

   Note: When you change the background color of the graph, Virtuoso Visualization and Analysis XL automatically adjusts the color contrast of various graph objects, such as traces, markers, labels, and tracking cursor to make them clearly visible on the graph.

Renaming and Closing Window

To rename a window, double-click the window tab and type the new name.

To close the window, close the window tab by clicking the cross button.

Handling Graph Objects

This section describes how you can select and delete a graph or its components, such as traces, axes, markers, and labels.

Selecting Objects

Click the graph or its component, such as trace, marker, or label, to select it. You can select multiple objects by holding down the `Ctrl` key while you click the required graph objects.

Deleting Objects

You can delete the objects, such as graphs, labels, markers, legends, and traces. You can also delete a window or a subwindow.

To delete an object, do the following:

1. Select the object you want to delete.
2. Choose *Edit – Delete*, or press the `Delete` key.

   The object selected in the window is deleted.

To delete all objects, select an object or a subwindow and do one of the following:

- Choose *Edit – Delete All.*
- Press `E`.

To delete all markers in a window, choose *Marker – Delete All* or press `Ctrl+E`.

To delete all traces in a window, choose *Trace – Delete All* or press `Shift+E`.

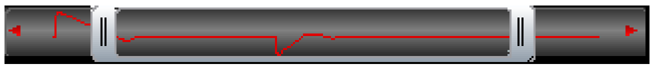
Panning and Zooming Graphs

You can pan and zoom a graph by using the pan bar and scroll bar displayed in the graph window.

This section covers the following topics:

Panning a Graph

You can use the pan bar located at the top of the window to pan a graph. When you plot a signal in the graph, the width of the pan bar is adjusted so that the entire graph is visible. Resize the pan bar by dragging either end of the pan bar inward to view the required portion of a graph. The selected portion of the graph is zoomed in to display greater detail. When the size of the pan bar is less than maximum, drag the pan bar to the left or right to view portions of the graph that are currently outside the display area. You can also click anywhere in the pan bar area to move the pan bar to that location.



Alternatively, to pan a graph, do one of the following:

- Right-click anywhere in a graph and choose *View – Pan Left / Pan Right / Pan Up / Pan Down*.
- Press the arrow keys to pan the graph in the required direction.
- Hold the `Ctrl` and `Alt` keys simultaneously. The mouse pointer changes to a hand symbol. Now, you can pan the graph by using the mouse left button.

Note: You can hide the pan bar by de-selecting the *Display zoom bar* check box in *Graph Options* tab of the Graph Properties form.

Zooming a Graph

The Virtuoso Visualization and Analysis XL tool supports multiple zooming operations.

To zoom in or out a graph, choose one of the following options from the *View* menu or click the relevant button on the zoom toolbar.

- *ZoomIn by 2*

  Zooms in the graph by a factor of two. A vertical scroll bar appears on the right side to help view graph areas outside the current window. To move horizontally to view areas outside the window use the pan bar displayed on the top of the window.

- *ZoomOut by 2*

  Zooms out of the graph by a factor of two.

- *Fit*

  Fits the graph in the window. Alternatively, right-click the trace and choose *Fit Trace* to fit the graph in the window. You can also use the bind key `f` to perform the zoom fit.

- *Previous*

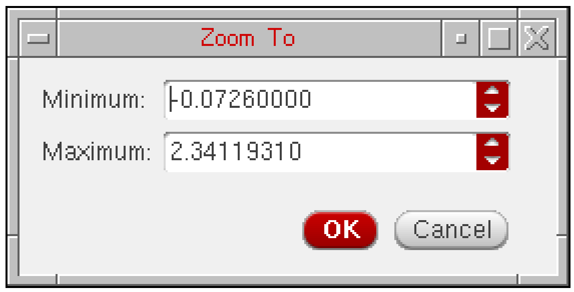
  Incrementally reverses a series of zoom and pan actions.

- Next

  Incrementally undoes the effect of the *Previous* command.

Alternatively, to pan and zoom in or out a graph, do the following:

- To pan a graph, right-click anywhere on the pan bar and choose one of the following options:
  - *Zoom to*—Zooms out the graph according to the specified size of the pan bar. When you select this option, the *Zoom To* form appears. In this form, select the maximum and minimum values for the pan bar.



  - *Zoom in X2*—Zooms in the graph in X direction by a factor of 2.
  - *Zoom out X2*—Zooms out the graph in X direction by a factor of 2.
  - *Fit X*—Fits the graph to the X-axis.
- To zoom in or out a graph, right-click anywhere on the scroll bar that is displayed at the right and choose one of the following options:

  Note: Scroll bar appears only if you zoom out the graph or strip.
  - *Zoom to*—Zooms out the graph according to the specified size of the scroll bar. When you select this option, the *Zoom To* form appears. In this form, select the maximum and minimum sizes for the scroll bar.
  - *Zoom in X2*—Zooms in the graph in Y direction by a factor of 2.
  - *Zoom out X2*—Zooms out the graph in Y direction by a factor of 2.
  - *Fit Y*—Fits the graph to the Y-axis.
  - *Fit Y to Visible X*—Fits the visible part of the trace to Y-axis based on the X-axis . This option is specific to an individual strip. This option finds the minimum and maximum Y-axis values that are visible in a strip and then performs a Y-axis zoom of those Y values.

Zooming In a Trace Along X- and Y-Axis

To zoom in a trace along X- and Y-axis, hold down the mouse button and drag the pointer to select the area on the graph that you want to zoom in. When you release the mouse button, the area you selected is zoomed in.

To zoom in the trace along one axis, do the following:

1. Press bindkey `X` to zoom in the graph along the X-axis or press bindkey `Y` to zoom in the graph along the Y-axis.
2. Hold down the right mouse button and drag the pointer to select the graph area that you want to zoom in.

   After you release the mouse button, the zoom is complete and the right mouse button zoom is reset to XY zoom, which means you can now zoom in or out the graph along both the axes.

Note: You cannot zoom in the graph by using the left mouse button. Also, ensure that you zoom the area toward the right of the Y-axis; otherwise, a shortcut menu appears.

Tip

If you have multiple strips in a window that you want to zoom in, place the pointer on the left or right edges of the strip container to start the zoom.

Panning And Zooming Graph With Mouse

To pan a graph with the help of mouse, perform the following steps:

1. Hold down the `Ctrl` and `Alt` keys simultaneously. Notice that the mouse pointer is changed to a hand symbol, which indicates that you pan the graph now.
2. Drag the mouse pointer to pan the graph in left or right direction.

Note: If a graph includes multiple strips, the panning procedure is performed on all the strips at the same time.

To zoom in or out a graph or a strip in Y direction with the help of mouse, do the following:

➡ Hold down the `Ctrl` key and move the mouse wheel button upward or downward.

If you move the wheel button upward, the selected strip is zoomed in, and if you move the wheel button downward, the selected strip is zoomed out. After the graph is zoomed in, a vertical scroll bar appears on the right side of the strip that you can use to view the remaining portion of the trace in the graph. You can also view the complete trace by using the mouse wheel button.

To zoom in or out all the strips in a graph in X direction, do the following:

➡ Hold down the `Shift` key and move the mouse wheel button.

When you move the wheel button upward, the selected strip is zoomed in, and if you move the wheel button downward, the selected strip is zoomed out.

Video

The video Panning and Zooming Qt Graph demonstrates how you can pan and zoom the Qt Graph.

Note: The zoom operations that you perform on the graph are added to a zoom stack and you can switch between the various zoom levels by selecting zoom commands in the View toolbar. The zoom stack can store maximum of 100 zoom operations.

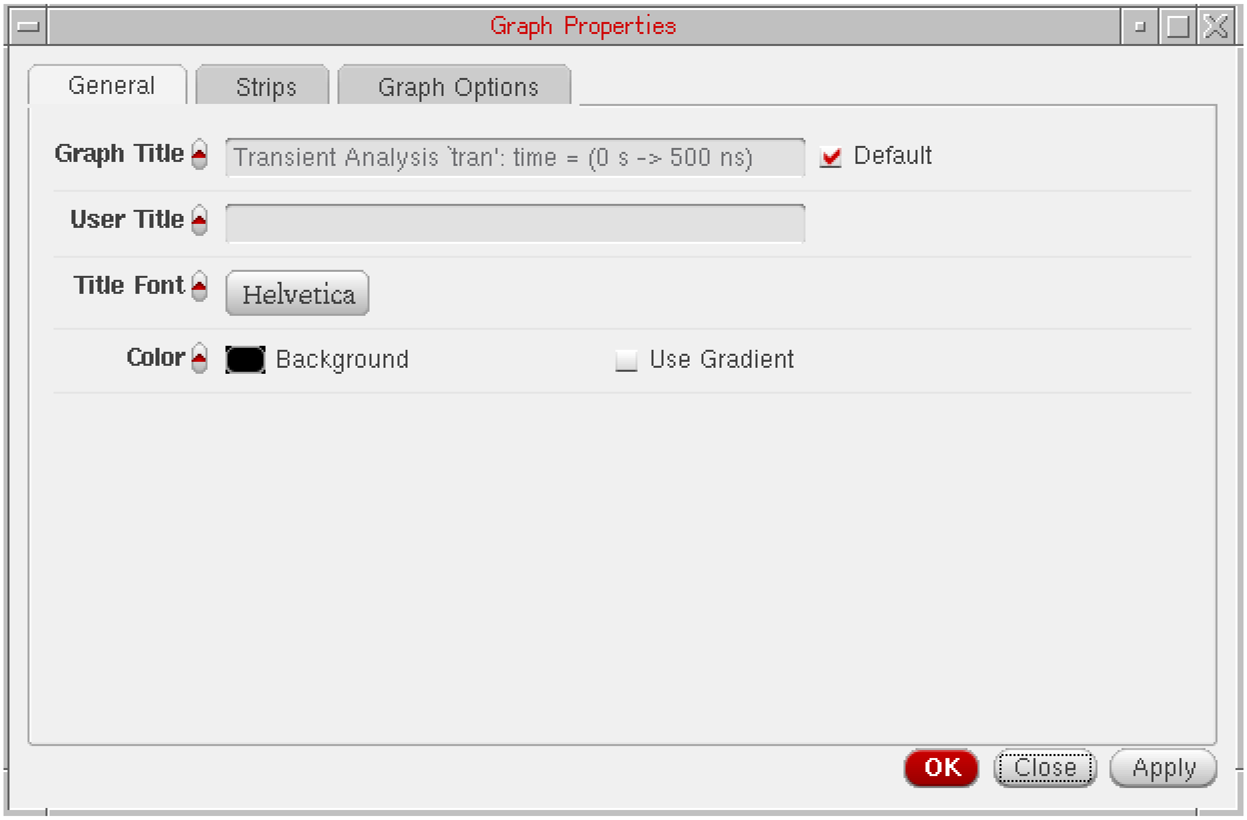
Editing Graph Properties

To set the properties of a graph, do one of the following:

- Choose *Graph – Properties.*
- Right-click anywhere in the window and choose *Graph Properties*.
- Double-click anywhere in the trace legend area.
- Press the bindkey `Shift+Q`.

The *Graph Properties* from appears. This form includes three tabs—*General, Strips,* and *Graph Options.*

Note: Double-click in the graph area does not open the graph properties form.



On the *General* tab, specify the following values:

- *Graph Title*—The title of the graph that is displayed at the top of the graph. When you select the *Default* check box next to this field, you cannot edit the graph title or provide a new graph title. The default graph name includes the name of the analysis and the Y-axis name. If you want to include in the title the date on which the simulation was run or the measurement was evaluated to obtain the signal that you plotted in the graph, select the *Simulation Date* check box.

  Note: In addition, to edit the graph title directly in the graph window, double-click the title. The title becomes editable and you can specify a new title.

- *User Title*—A name for the window. You can edit this field if the default check box is not selected. Specify a name and click *OK*, this name is saved, and the *Graph Properties* form title is displayed as *Graph Properties for <graph-name>* when you open the form next time.

- *Title Font*—The font properties for the window title.

- *Color*—The background color for the selected window.

For information about the fields on the *Strips* tab, see Setting Strip Properties on page 189.

On the *Graph Options* tab, specify the following values:

- *Font*—The font properties for the graph and its components, such as labels, axes, and markers.
- *Notation*—The graph notation that can be `Scientific`, `Engineering`, and `Suffix`.
  Default value: `suffix`.
- *Reload using current context*—Select this check box if you want the graph to be reloaded with data from the current in-context results directory. If you do not select this option, the signals in the graph are loaded from their individual databases.
- *Legend position*—Set the trace legend position as `left`, `inside`, or `above`.
  Default value: `left`.
- *Display zoom bar*—Shows or hides the zoom bar displayed on the current window or subwindow.
- *Show delta child labels*—Select this check box to hide the labels for all point markers that are used to form a delta marker.
- *Marker Significant Digits*—Set the marker significant digits to `Auto` or `Manual`. If you select `Manual`, you need to provide the number of significant digits.

- Click *OK*.

Note: A red button is displayed with each form field in the properties form for graph and graph objects, which acts as a toggle switch. By default all the form fields are opened in the edit mode. If you click this button, you cannot edit the form fields.

Editing the Graph Title

To edit a graph title, double-click anywhere in the graph title area. The mouse pointer changes into a cursor. You can now delete the existing grapg title and type the new title. While editing, you can also use the keyboard arrow, `Home`, or `End` keys.

Working with Graph Axis

This section covers the following sections to describe how to use X- and Y-axis while plotting and analyzing signals in a graph.

Editing Graph Axis Attributes

The X-axis attributes set the attributes for the X-axis. The X-axis attributes provide a mechanism to easily create YvsY plots.

Note: Eye diagrams can also be plotted from the Calculator, while a limited version of YvsY plot is available from the Results Browser.

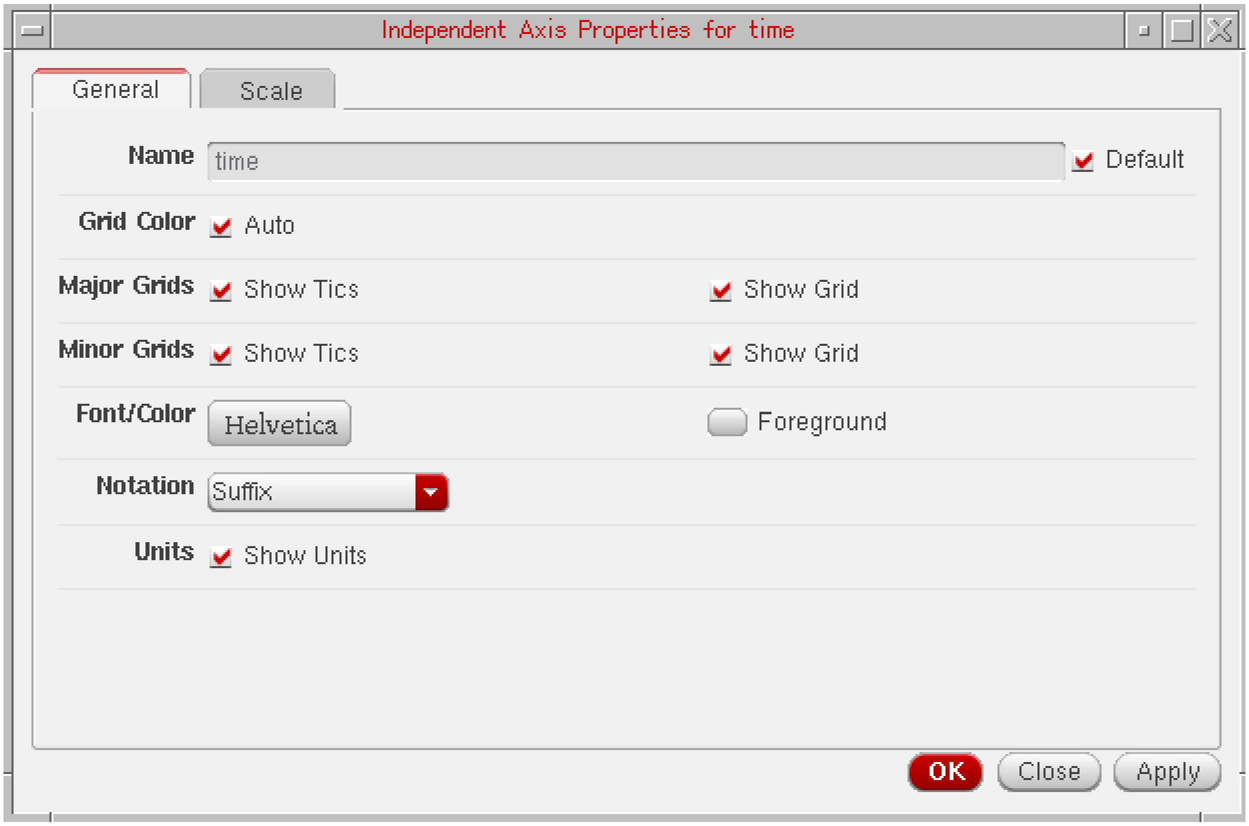
The default graph attributes are controlled by the values assigned to variables in the `.cdsenv` file. For more information, see Appendix A, "Virtuoso Visualization and Analysis XL Tool Environment Variables."

You can edit the attributes of the axes by doing one of the following:

- Double-click an axis in the window.
- Select an axis and choose *Axis – Properties*. You can select more than one axis in the subwindows by using the `Shift` key.
- Right-click an axis and choose *Axis Properties*.

  The *Independent Axis Properties for <X-axis-name>* form appears for X-axis and *Dependent Axis Properties for <Y-axis>* appears for Y-axis.

This form includes two tabs—*General* and *Scale*.



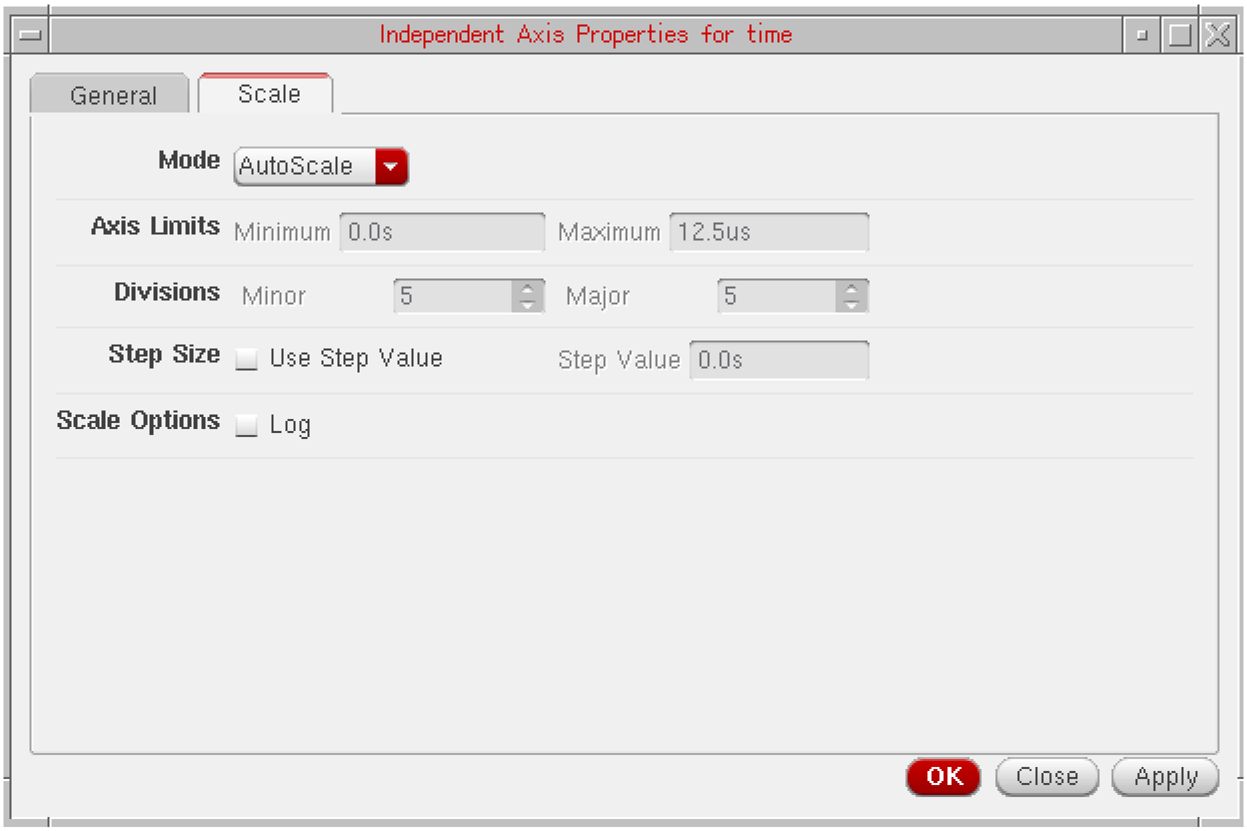
On the *General* tab, specify the following values:

- ❑ *Name*—The default name of the selected axis. You can change the name, if required. The changed axis name is displayed when you click *OK*. If you select the *Default* check box next to this field, you cannot change the axis name.
- ❑ *Label with Axis Number*—Select this check box if you want to display the axis number in the selected axis name. This field is displayed only for the Y-axis (dependent axis) properties form.
  For more information about axis number, see Adding Multiple Y-Axes on page 119.
- ❑ *Grid Color*—Select to set the default grid color.
- ❑ *Major Grids*
  - ❍ *Show Tics*—Select to display the major axis divisions.
  - ❍ *Show Grid*—Select to display the major grids.
- ❑ *Minor Grids*

- ❍ *Show Tics*—Select to display the minor grid divisions.
- ❍ *Show Grid*—Select to display minor grids.

- ❑ *Font/Color*—The font of the axes labels and divisions.
- ❑ *Notation*—The notation displayed for the axis labels values. The available values are—`Engineering` and `Suffix`. Default value: `Suffix`.
- ❑ *Units*—Select to display the axes units. Alternatively, you can display or hide the axes units by right-clicking the axes and choosing *Show Units*.

On the *Scale* tab, specify the following values:



- ❑ *Mode*—The scaling mode as *AutoScale* or *Manual*.
- ❑ *Axis Limits*—The maximum and minimum range of the selected axis on which a signal can be plotted.

  Note: If you select the *AutoScale* mode, you cannot change the axis limits. To change the axis limit, select the *Manual* option. The minimum size supported by axes is `1e-24`. This covers the full range supported by Si suffixes.

- ❑ *Divisions*—Specify the minor and major axis divisions for the selected axis. If you selected the *AutoScale* mode, you cannot change the axis divisions. To change the axis divisions, select the *Manual* option.
- ❑ *Step Size*—Select the *Use Step Value* check box to specify a step value for major grids. This step value indicates the spacing between major grids on the graph.
- ❑ *Scale Options*—Select the *Log* check box to display the axis in logarithmic scale.

- Click *OK*.

Changing Digital Dependent Axis Properties

To change the properties of the digital dependent axis, right-click the trace and choose Digital Axis Properties. The Digital Dependent Axis Properties form appears. You can use this form to change the axis font and foreground color by setting the *Font/Color* field.

Changing Axes Scale to Logarithmic

To display the dependent or independent axes scale in logarithmic values, do one of the following:

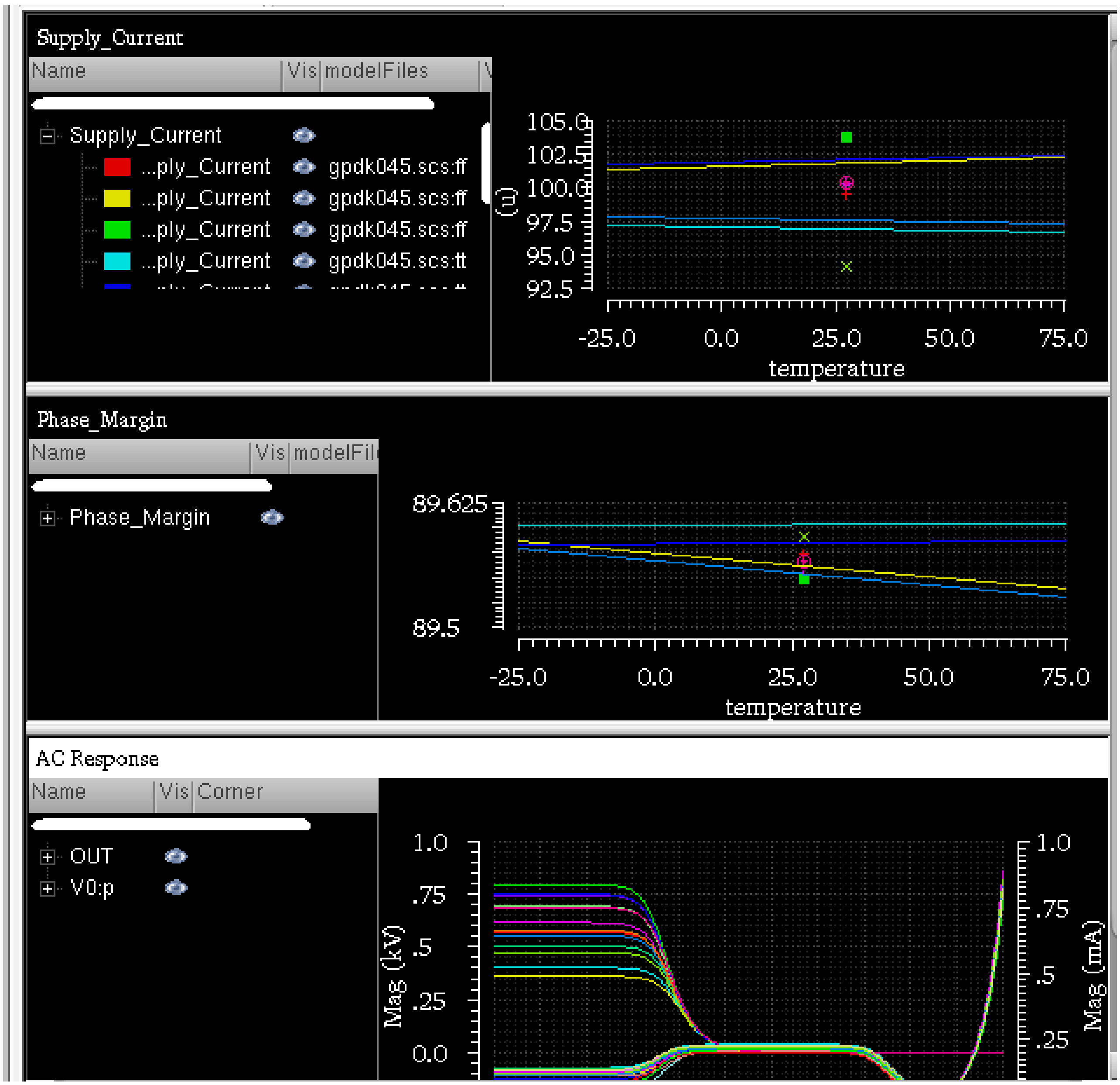
- Right-click the axis and choose *Log Scale*.
- Select an axis and choose *Axis – Log Scale*.
- Select the *Log* check box under *Scale Options* in the Axis Properties form.

  The scale for the selected axis changes to Logarithmic. Now, if you drag another signal from the Results Browser and plots the signal in the same graph in append mode. The dragged signal is plotted on the logarithmic scale on X-axis.

Displaying X-Axis Labels in String Format

If you select the X-axis variable as model file or Corner while plotting the results for a simulation run in ADE XL for sweep data, the labels on X-axis are displayed in string format.

In the figure below, you can see the simulation results plotted in the graph window. The sweep variables for this simulation are—*VDD*, *modelFiles*, and *temperature*. This simulation also contains corner values. After you run the simulation, the different outputs are listed in the *Output* section of ADE XL. When you plot all the outputs, the waveforms are plotted in individual subwindows. See the figure below. Notice that the plots shown in the figure below have temperature as the sweep variable on X-axis. You can change the X-axis variable to *modelFiles*, *VDD*, or *Corner*. When you change the X-axis variable to *modelFiles* or *Corner*, the X-axis labels are displayed as string values.



Note: Labels with string values are not supported for transient data.

To change the sweep variable to *modelFiles* or *Corner*, do the following:

1. Right-click the X-axis and choose *Swap Sweep Var*.

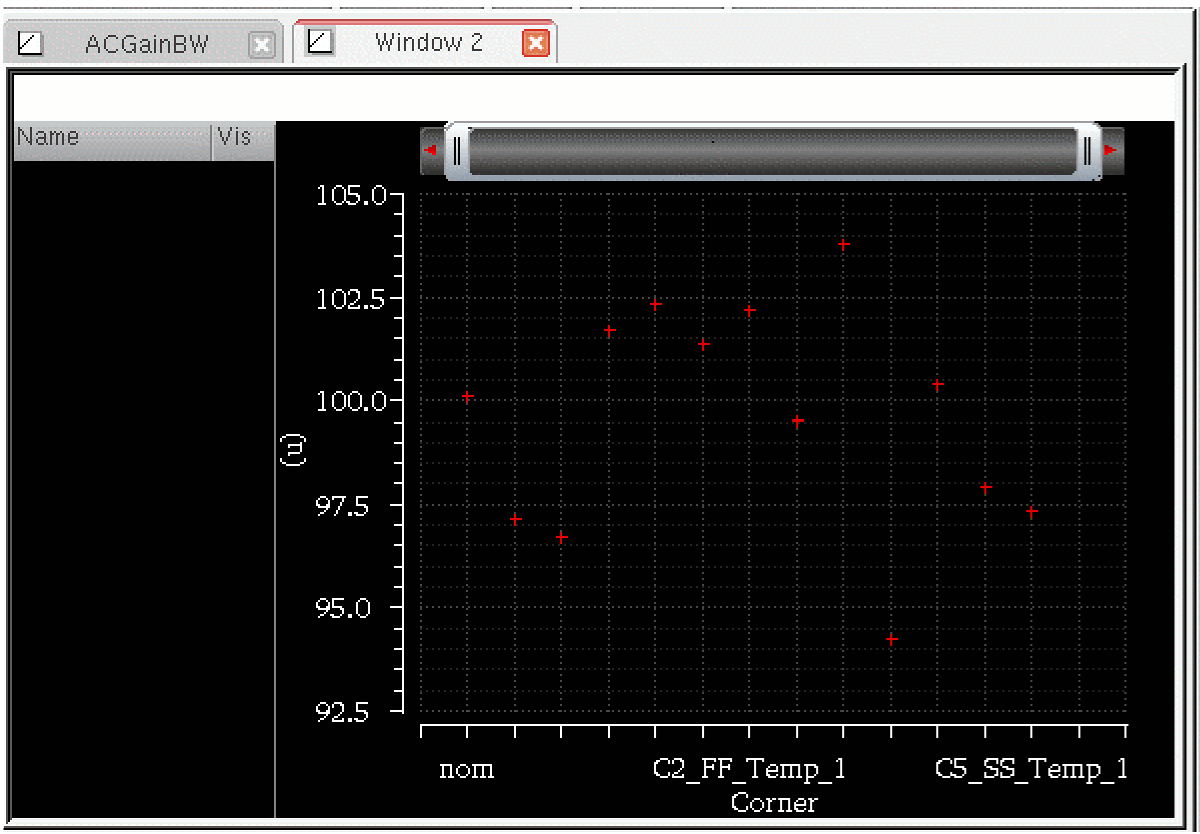
   The *Swap Sweep Var* form appears.

2. Select *modelFiles* or *Corner.*

   The graph is plotted with the selected variable displayed on X-axis.

   In the following figure, the sweep variable plotted on X-axis for `Supply_Current` plot, shown in the figure above, is changed to *Corner*. Notice that the X-axis labels for corners are displayed as string values and the trace is displayed as a sequence of points.

   Note: To display the trace as a continuous line, right-click the trace and choose *Type – Continuous line.*



Also, note that the string for all the intercept points may not be visible on the X-axis when the trace is displayed in its normal size. To view any specific string for a data point, you need to zoom in the graph. For information about how to zoom and pan a graph, see Panning and Zooming Graphs on page 104.

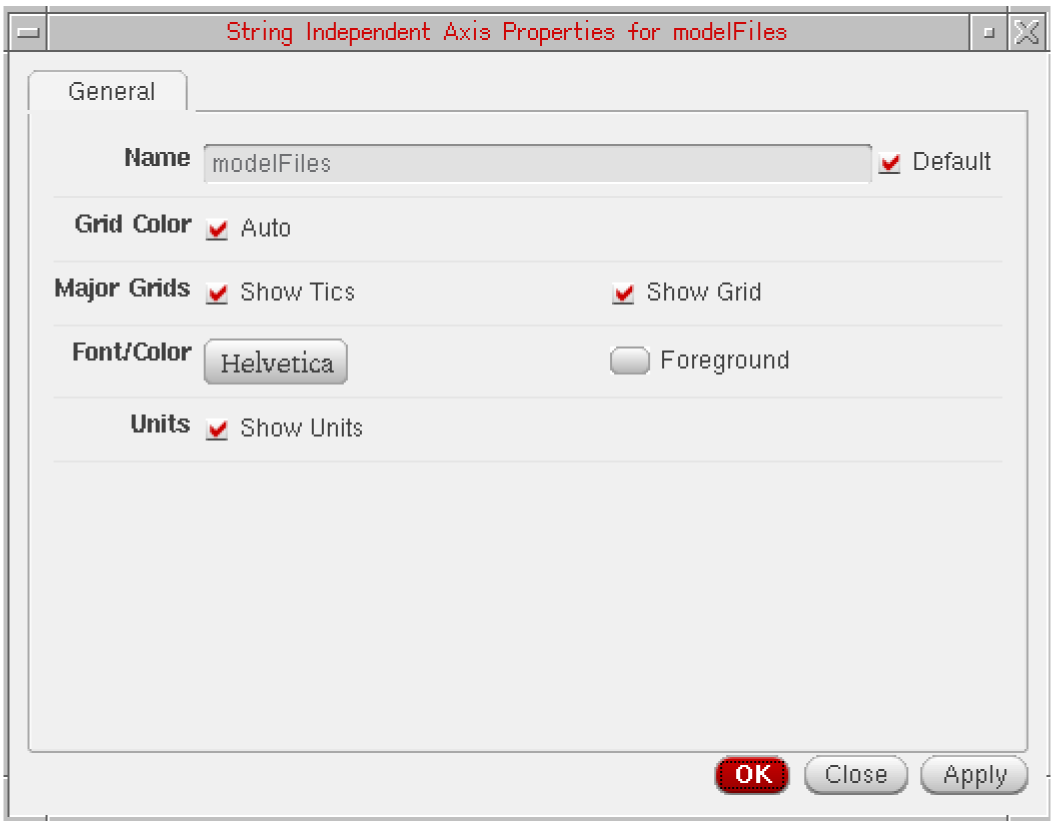
If the string is long, it is displayed as an elided string with the ... symbol, for example, `CO_VDD...Temp_1`. To view the complete string, place the pointer on a data point.

Changing Properties of the String Independent Axis

To change the properties of the independent axis (X-axis) that includes the string intercept values, do the following:

- Right-click the Independent axis and choose *Axis Properties*.
- Select the independent axis and choose *Axis – Properties*.

The *String Independent Axis Properties for <axis name>* form appears.



This form has only the *General* tab, which includes the following fields:

- *Name*—The default name of the selected axis. You can change the name, if required. The changed axis name is displayed when you click *OK*. If you select the *Default* check box next to this field, you cannot change the axis name.
- *Grid Color*—Select to set the default grid color.
- *Major Grids*
  - *Show Tics*—Select to display the major axis divisions.

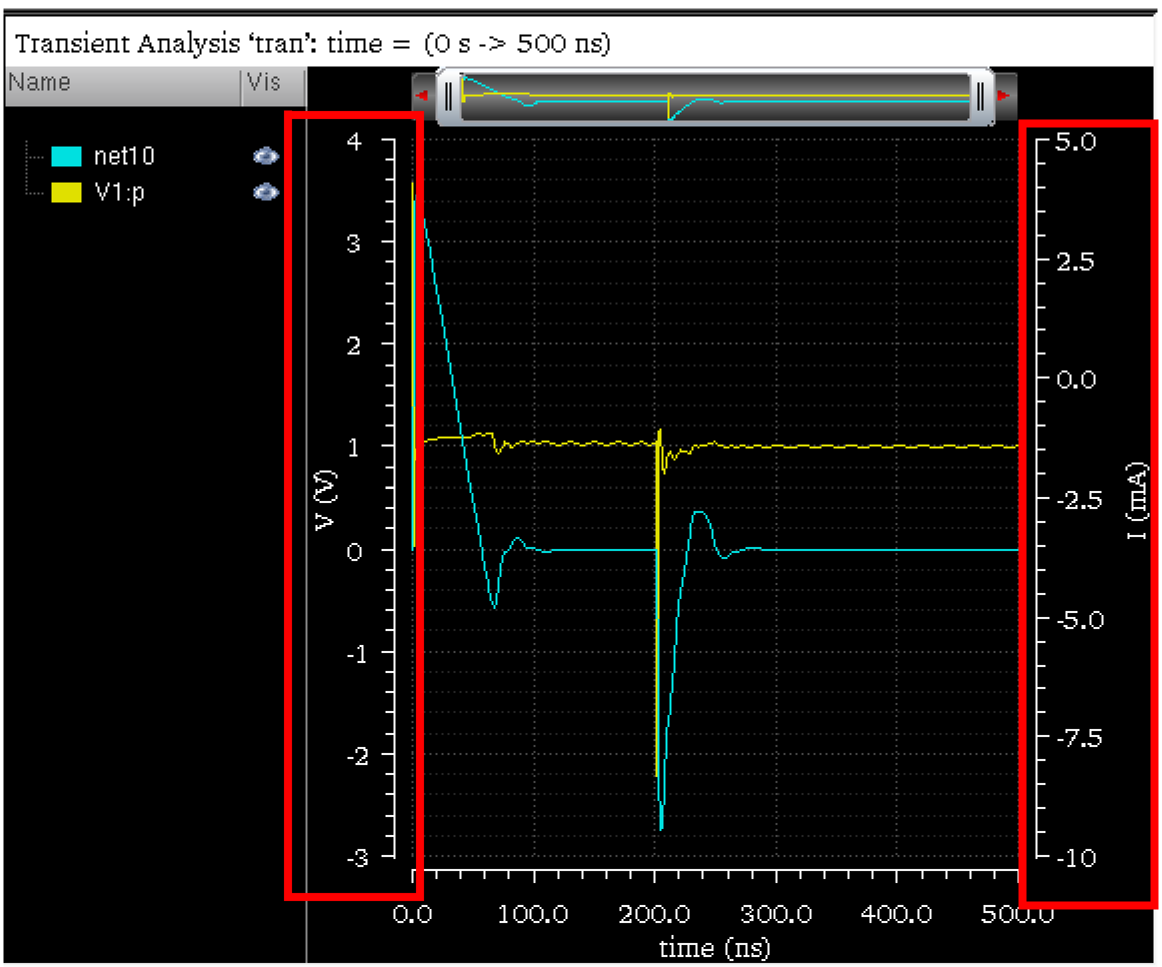
- ❑ *Show Grid*—Select to display the major grids.

- *Font/Color*—The font of the axes labels and divisions and foreground color.

- *Units*—Select to display the axes units. Alternatively, you can display or hide the axes units by right-clicking the axes and choosing *Show Units*.

Adding Multiple Y-Axes

From Results Browser or ADE, if you plot two or more signals that contain different Y-axis (dependent axis) data in the same window, the graph displays separate Y-axes for both the signals. For example, when the voltage (`net10`) and current (`V1:p`) signals are plotted in the same graph, the graph displays two Y-axes, displayed on the left and the right of the graph respectively, as shown in figure below:



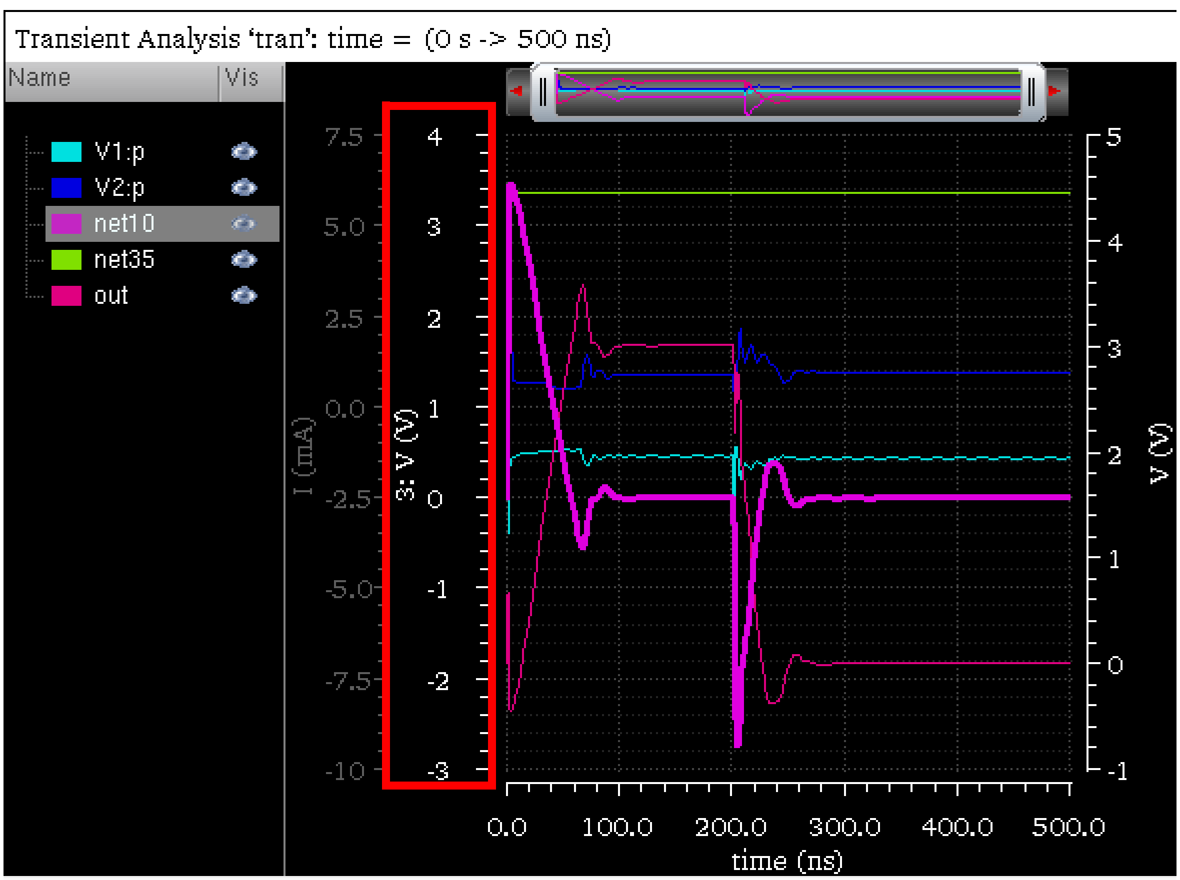
Note: A graph or a strip can have four Y-axes at the maximum. If the graph already contains four axes and you plot a trace that requires a new Y-axis to be added to the graph, then the new trace is plotted in a new strip.

When you have more than one trace plotted in a graph and you want to analyze a particular trace, you can move the selected trace to a new Y-axis. You can also change the Y-axis of a trace to another existing Y-axis if the graph contains two or more Y-axes. To know more about how to change the axis of a trace, see Changing Dependent Axis (Y-Axis) on page 121.

To assign a new Y-axis to the selected trace:

➡ Right-click the trace and choose *Change Y Axis – New*.

A new Y-axis is added in the graph and the selected trace is detached from the existing axis and is attached to the new axis. For example, the figure below contains five traces (two voltage and three current) plotted in a graph. When you assign a new Y-axis to the trace for the `net10` signal, the trace moves to a new Y-axis (`3: V(V)`) displayed on the left of the graph. The new axis is the third Y-axis in the graph; therefore, the axis number is 3.



Important

If an axis does not have any traces attached to it, the axis is removed from the graph.

When you add a new Y-axis for a trace, the new axis name is displayed in the following format:

`axis_number: axis_title(axis_unit)`

For example, the above figure displays the following axis name for the new axis that you have added manually to the `net10` trace.

`3: V(V)`

where,

- `axis_number` is `3` because this is the third Y-axis in the graph
- `axis_title` is `V`, which indicates this is a voltage signal
- `axis_unit` is `V`, Volts

By default, axis name displays the `axis_number`. To hide `axis_number` from the axis name, do one of the following:

- Right-click the axis and de-select the *Show Axis Number* check box.
- In the *Dependent Axis Properties* form, on the *General* tab, deselect the *Label with Axis Number* check box and click *OK*. To know how to open the *Dependent Axis Properties* form, see Editing Graph Axis Attributes on page 111.
- Choose *Axis – Axis Number*.

  The `axis_number` is removed from the selected axis name.

Changing Dependent Axis (Y-Axis)

If you have more than one Y-axis in a graph, you can change the Y-axis of the trace to another Y-axis. You can assign a common axis to the traces that have the same signal type. However, you cannot assign the same axis to signals of different data types. For example, a voltage signal cannot be assigned an axis of the signal representing current.

To change the Y-axis of a selected trace:

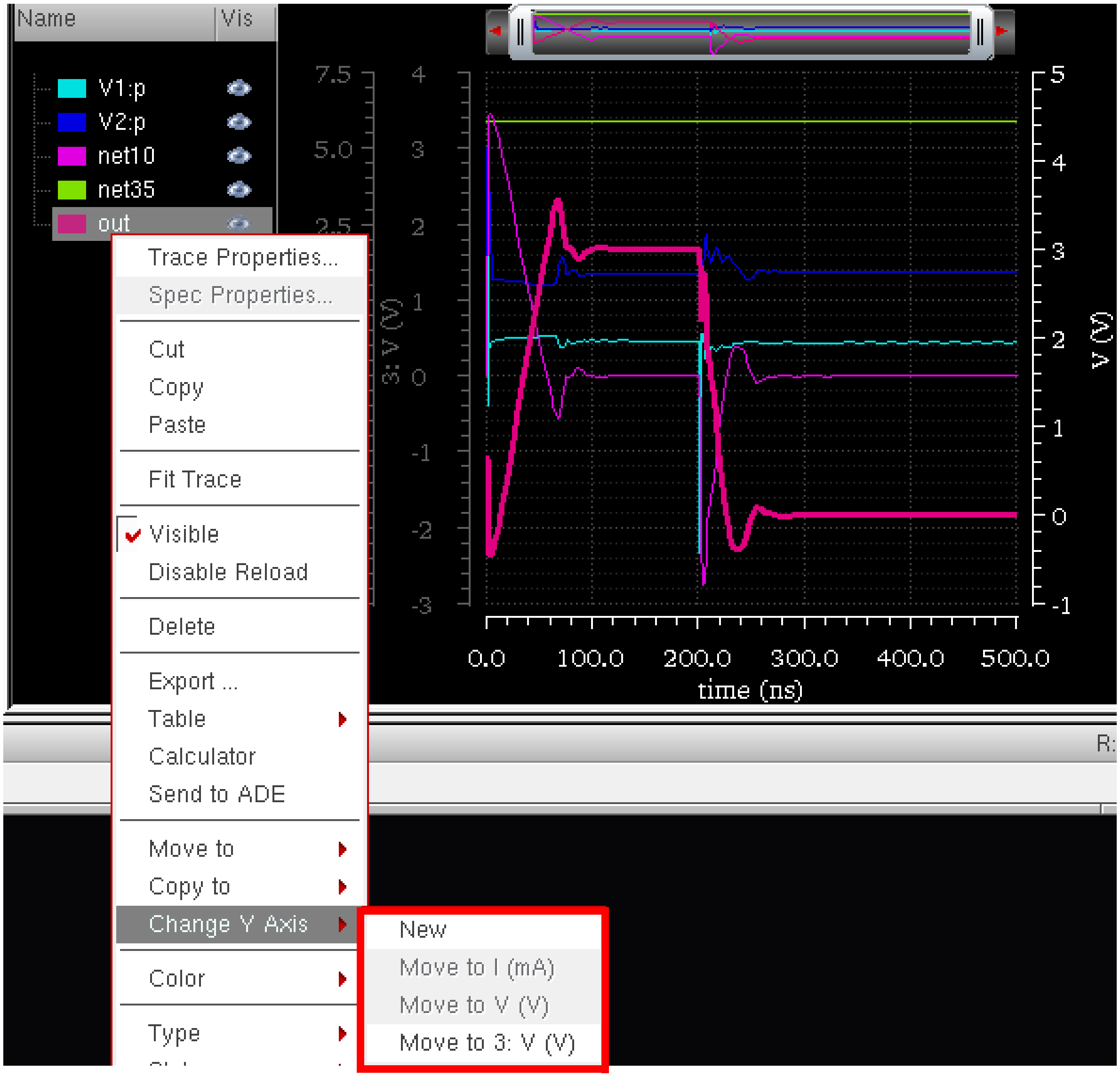
➡ Right-click the trace and choose *Change Y Axis – Move to axis_name*.

  The shortcut menu displays the names for all the Y-axes that are currently visible in the graph. The name of the axis to which the trace is currently attached is disabled.

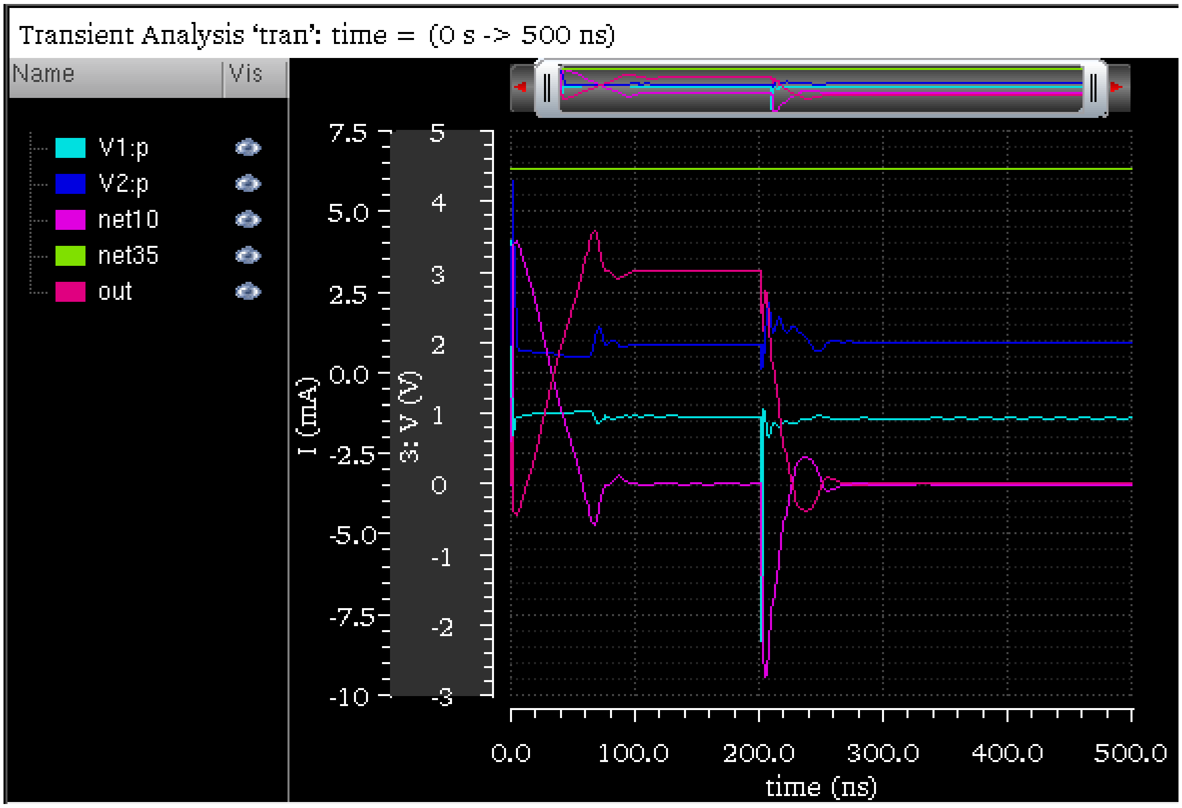
For example, in the following figure, the graph includes five traces:

- `V1:p`—Current signal plotted on `I(mA)` axis
- `V2:p`—Current signal plotted on `I(mA)` axis
- `net10`—Voltage signal plotted on `3:V(V)` axis
- `net35`—Voltage signal plotted on `V(V)` axis
- `out`—Voltage signal plotted on `V(V)` axis

The `net10`, `net35`, and `out` signals are the voltage signals. Therefore, you can move these signals to any axis that represents the voltage signal. In this example, you can move the `out` signal to only `3: V(V)` axis. The `I(mA)` axis is disabled in the shortcut menu because it is incompatible with the `out` signal. The `V(V)` axis is disabled because the `out` signal is already assigned to this axis.



When you move both the `out` and `net35` signals to the `3: V(V)` axis, the `V(V)` axis is removed from the graph because no signal is attached to this axis (see the figure below).

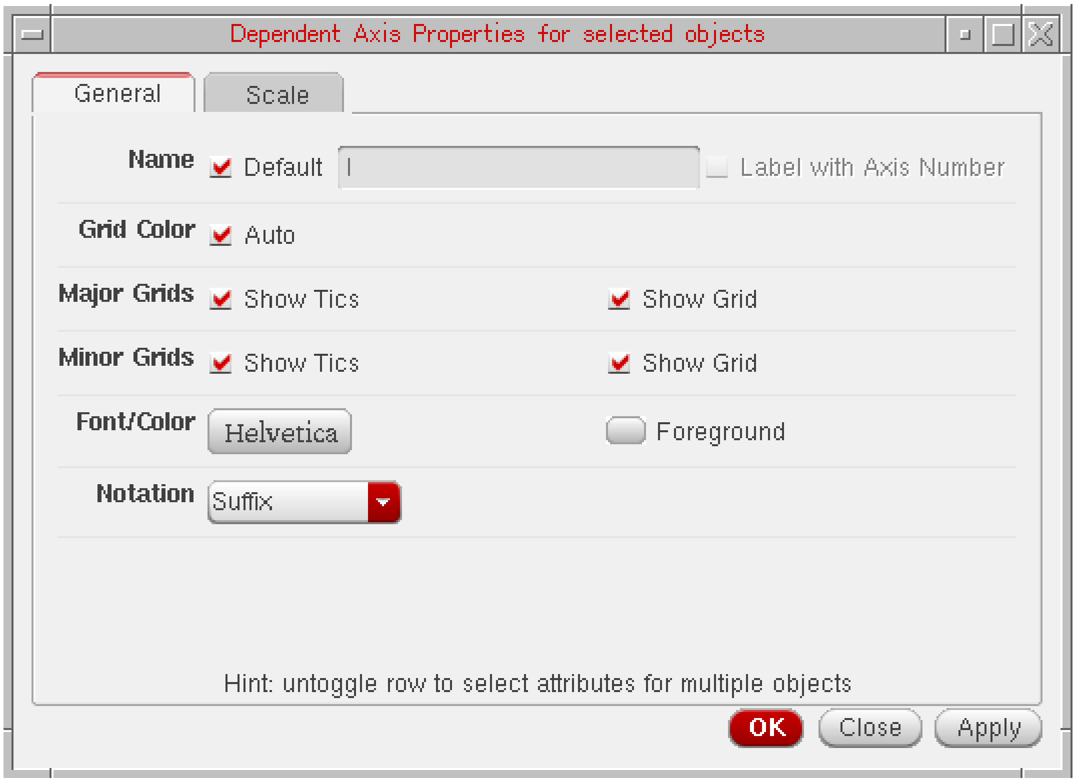


To find which traces are assigned to a particular axis, right-click the axis and choose *Select Attached Traces*. The traces attached to the selected axis are highlighted in the graph. Also, when you select a trace, only the axis for the selected trace is highlighted and all the other axes are dimmed.

To change the properties of all the dependent axis at the same time, do the following:

- Press the `Ctrl` key and click the axes for which you want to change the axis properties.
- Choose *Axis – Properties*.

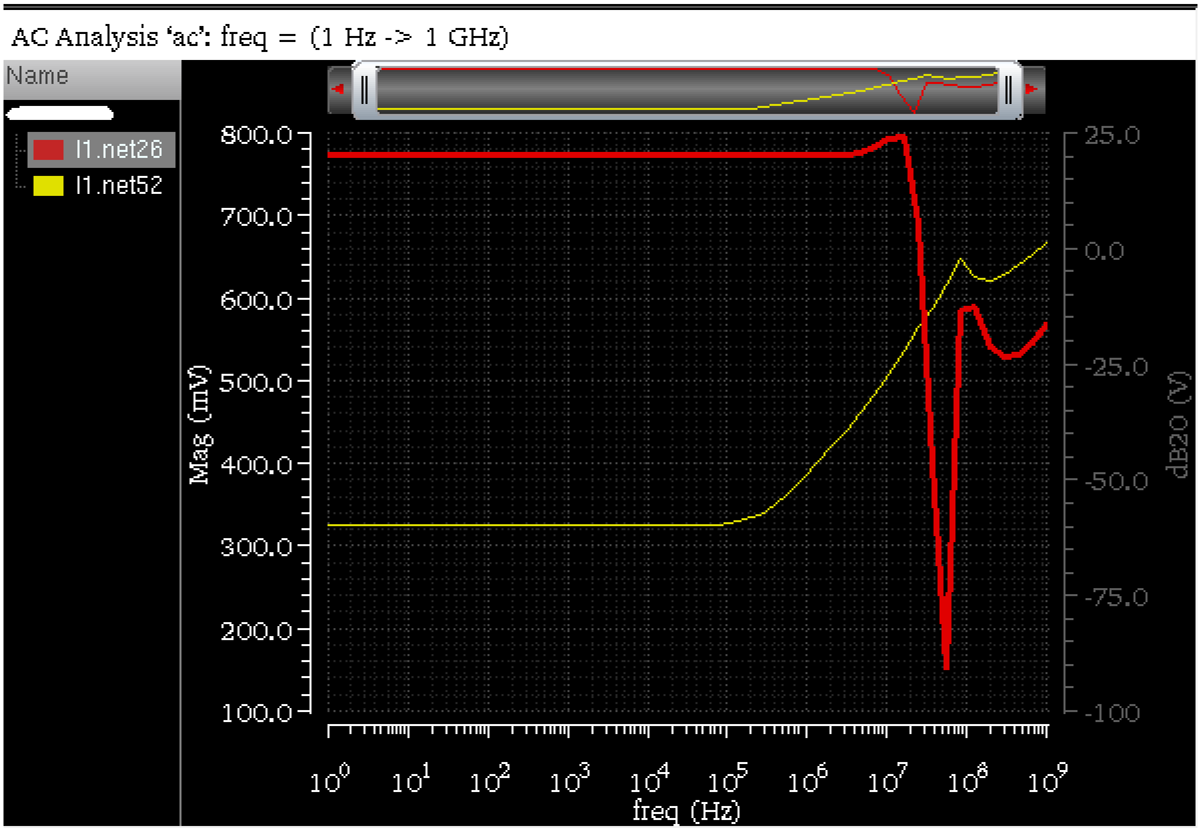
The *Dependent Axis Properties for selected objects* form appears.



For more information about the properties form fields, see Editing Graph Axis Attributes on page 111.

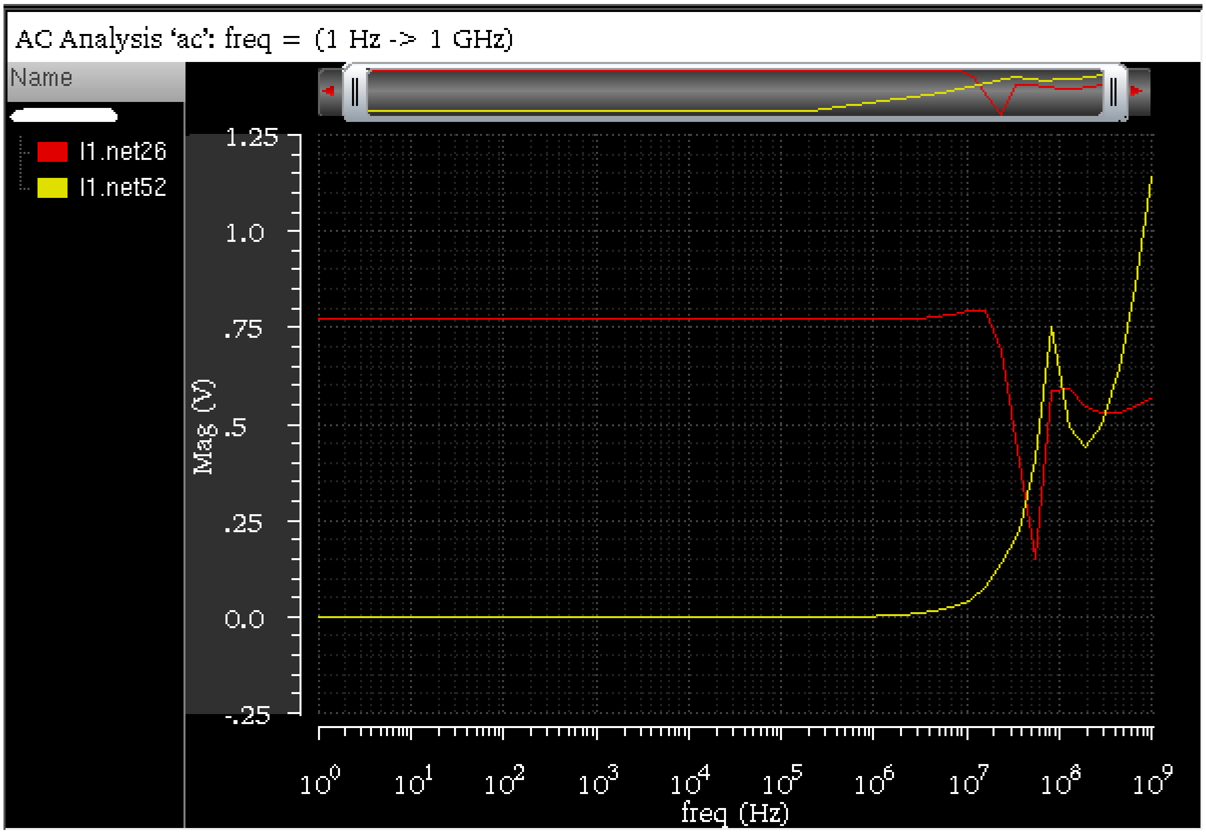
Plotting Traces Using Different Modifiers

You can plot two signals with different modifiers in the same graph along different Y-axes. In the example below the `I1.net26` signal is plotted with `Magnitude` as dependent modifier and `I1.net52` signal is plotted with `dB20` as the dependent modifier, the traces are plotted along different Y-axes in the same window.

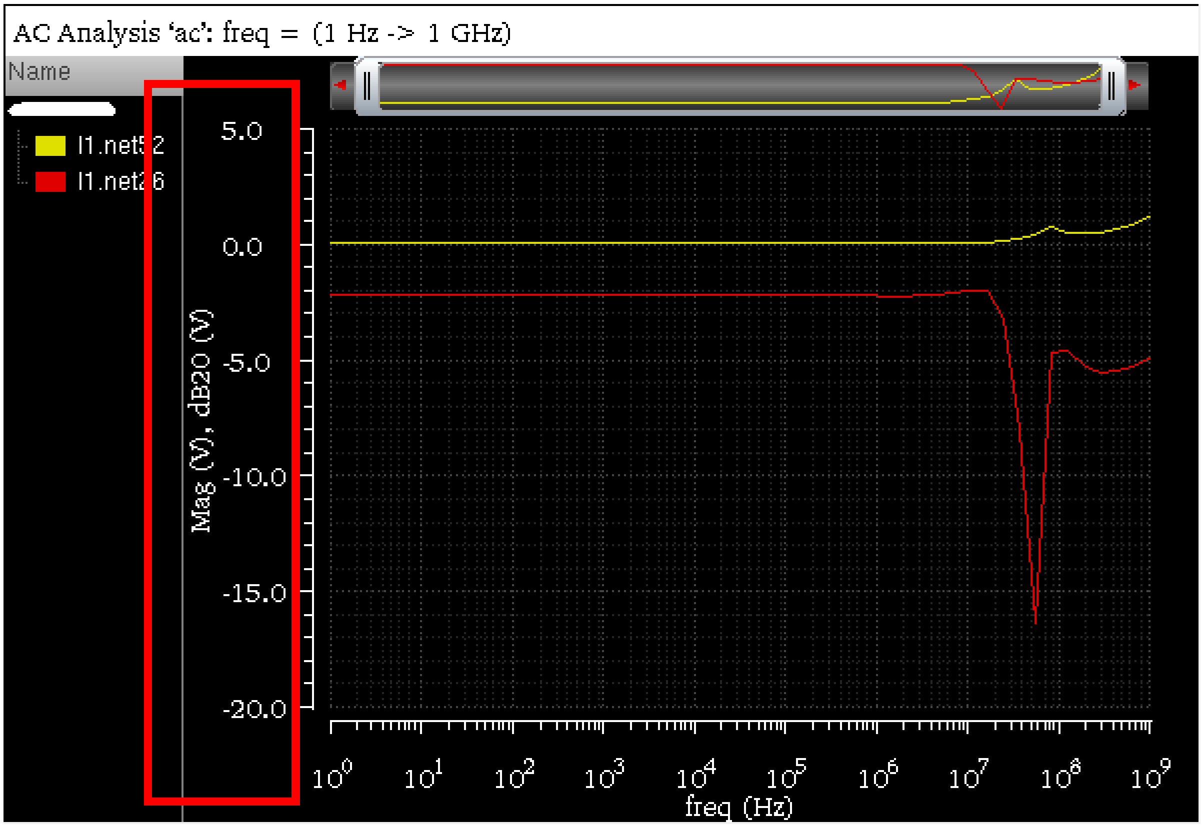


By default, the Y-axis label displays the name of the modifier, such as `Mag (mV)`. You can change the Y-axis label by using the Axis Properties form.

Now, if you change the dependent modifier for trace `I1.net52` to `Magnitude`, the trace is moved to the existing Y-axis `Mag (V)`.



When you plot these two signals in the same graph with `Magnitude` as dependent modifier and then change the dependent modifier value of signal `I1.net26` to `dB20`. In this case, both the signals remain plotted on the same Y-axis with label, `Mag (V), dB20 (V)` (as shown in figure below).



Plotting Multiple Signals on a Common Axis

When you plot signals with different Y-axis values, the signals are plotted on different Y-axes in the same graph. Similarly, when you plot signals with different X-axis values, the signals are plotted in different windows. However, there are some situations where you may need to plot signals with different Y- or X-axis data on a common Y- or X-axis. Following are a few examples where you require plotting signals on the same Y-or X-axis:

- Plotting phase noise and AM component of noise on the same Y-axis
- Generating Stability plots—Plotting dB and phase of the same signal on a common Y-axis for easier measurement
- Plotting measurement expressions with different, but equivalent units on the same Y-axis
- Plotting results from two different simulators on the same X-axis
- Plotting simulated and measurement data on the same Y- and/or X-axis.

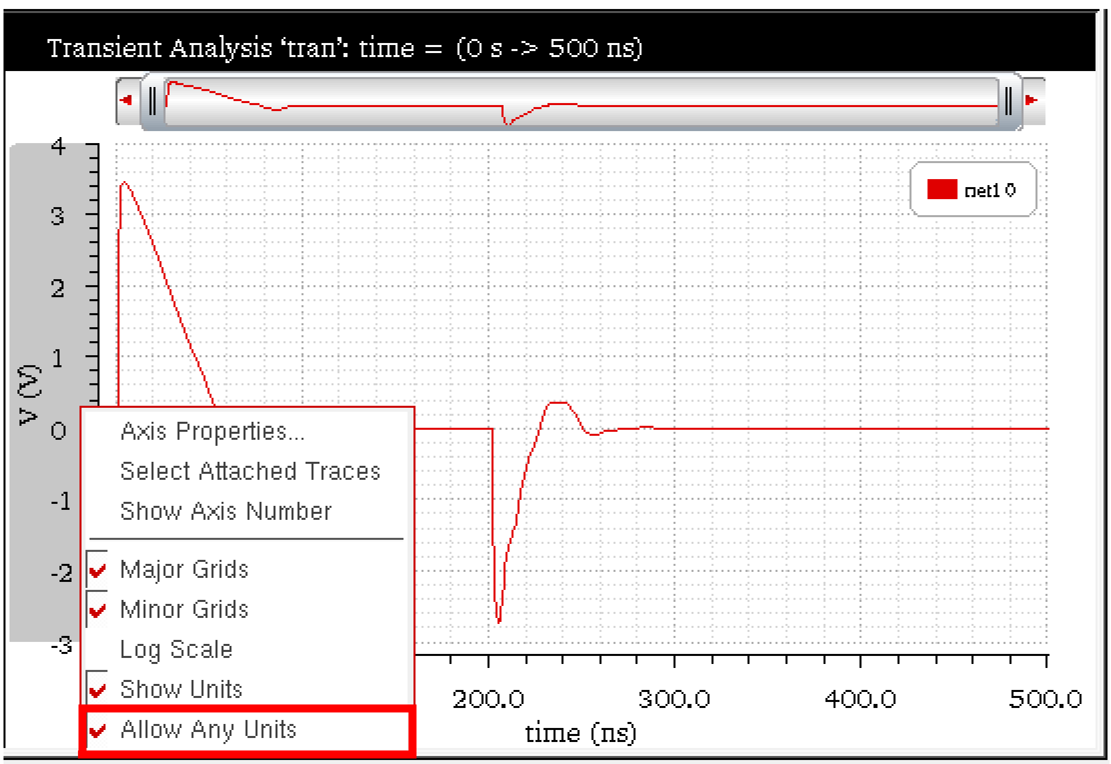
To plot multiple signals with different Y- or X-axis values on the same Y- or X-axis, you can do one of the following:

- Right-click Y- or X-axis and choose *Allow Any Units*.
- On the *Scale* tab of the Y- or X-axis Properties form, select *Units – Allow Any Units* check box.

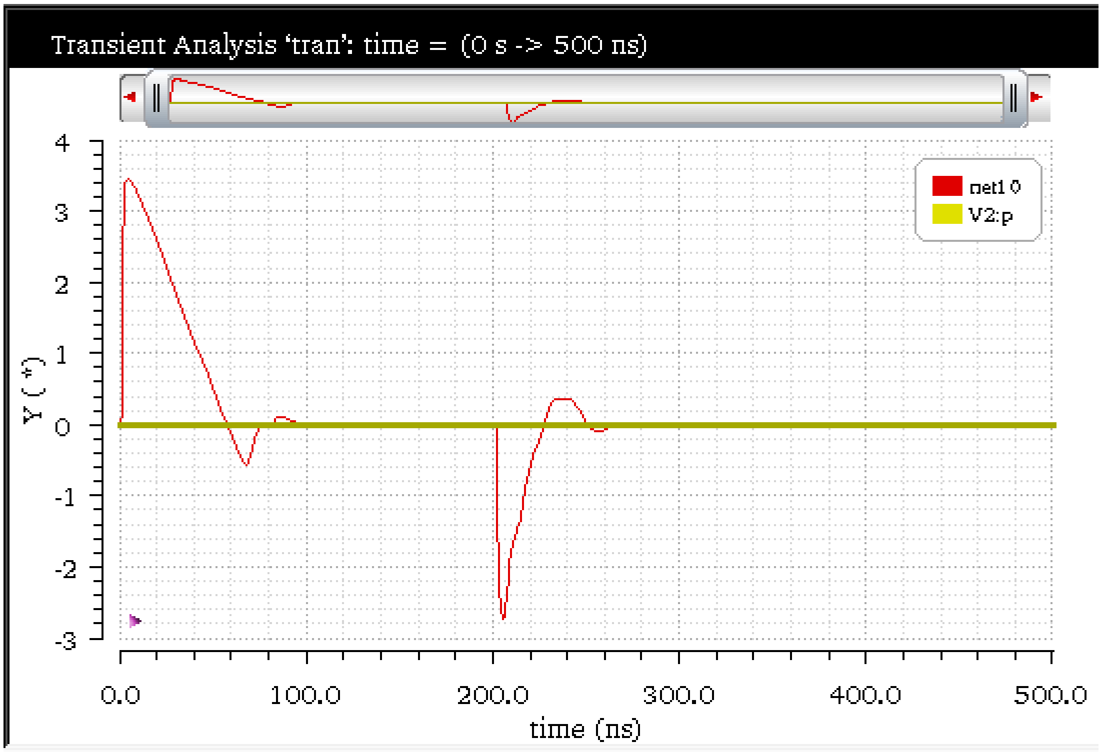
  Now, when you plot signals with different Y- or X-axis values in the same graph, they are plotted on the same Y- or X-axis. After the signals are plotted, the Y-axis title changes to `Y(*)` and the X-axis title changes to `X(*)`.

Example

The following figure displays a voltage signal, `net10`, plotted on a graph. When you right-click the Y-axis, `V(V)`, on this graph and choose *Allow Any Units*, the Y-axis becomes flexible to plot any signal.



Now, in append mode, if you plot a current signal, `V2:p`, in the same window, the signal is plotted based on the existing Y-axis values, as shown in the figure below. Note that in this figure, the Y-axis label has changed to `Y(*)`, which shows that you can plot signals of different Y-axis values on this graph.



Merging Two Y-Axes

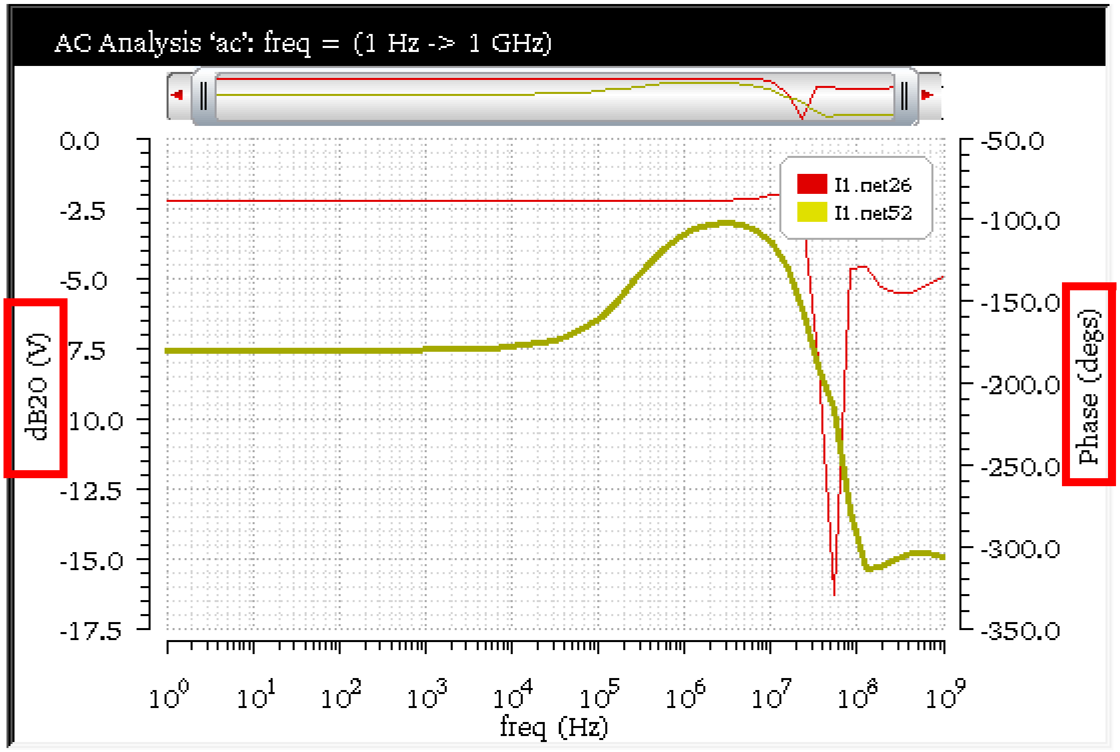
If you have two or more Y-axes present in a graph to display signals from different Y-axis data, you can merge these two Y-axes and plot the signals along one Y-axis. To do so, perform the following step:

➡ Right-click the trace for which you want to merge the axis and choose the target axis-name with which you want to merge.

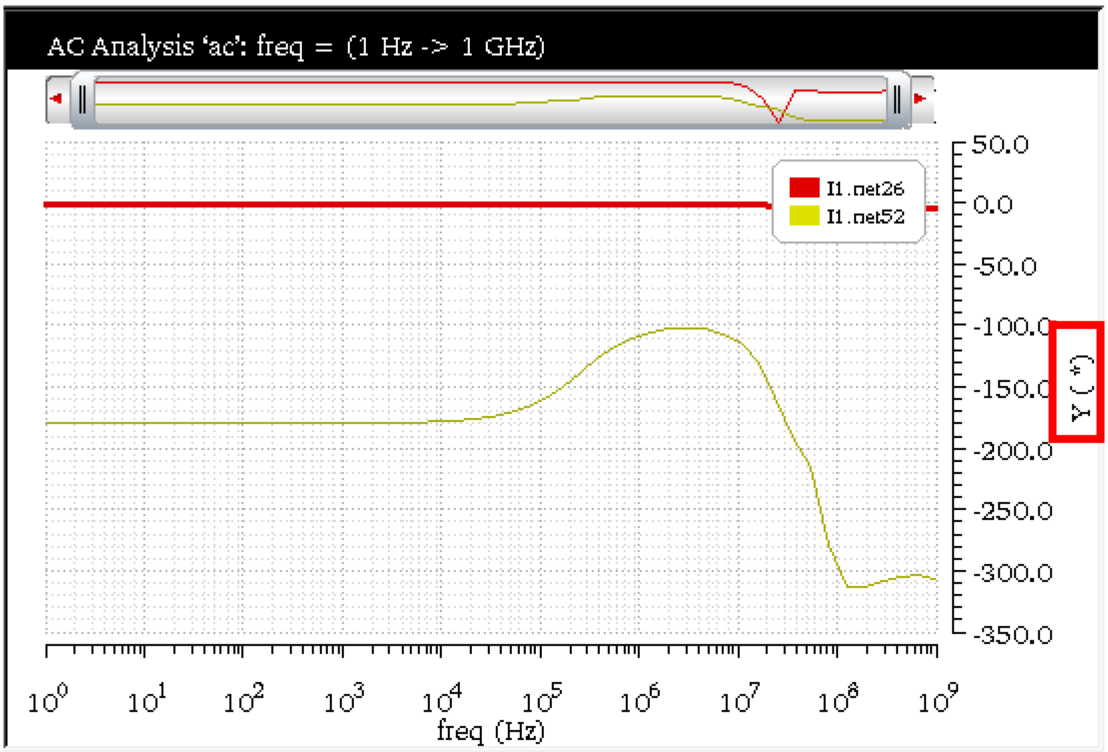
Now, when you plot a signal from Results Browser, Calculator, or ADE, the signals are plotted on the merged Y-axis.

Example

The following figure displays the `net26` and `net52` signals plotted on two Y-axes, `dB20` and `phase`, respectively.



Now, if you want to change the axis of the `net26` signal, right-click the trace for the `net26` signal and choose *Change Y-axis – Move to phase(deg)*. The `net26` signal is now plotted based on the units on the selected axis. Note that the Y-axis title has been changed to `Y(*)`, which indicates that you can plot any signal on this axis. Also, note that the *Allow Any Units* check box for this axis is always selected when the two axes are merged.



Locking Graphs

You can lock a graph to ensure that it does not change even when the simulation results in the data directory are reloaded. No signal can be added or removed from a locked graph. If you try to append a signal to a locked graph, it is plotted in a new subwindow. The graph operations like zooming and adding or moving markers are supported in a locked graph.

To lock a graph:

- In the window, choose *Graph – Lock*.

  A lock icon appears on the top-right corner of the window indicating that the graph is locked

To lock the size of a strip:

- Right-click anywhere in the strip and choose *Lock Strip Size*. When you view the menu again, you see a red check mark displayed next to the option, indicating that the strip size is locked.

A lock icon appears on the upper-left corner of the window tab indicating that the strip is locked.

Evaluating Graph Expressions

The signals plotted in the window can have expressions associated with them. How expressions are evaluated in the window depends on whether you are in the SKILL or MDL mode.

SKILL Calculator

When you open a saved graph that contains expressions, the expressions are evaluated by default within the context of the current results directory.

For example, if you plot an expression from a results directory and save the graph and later select a different results directory. When you open the saved graph again, the expression in the graph is evaluated in the context of the new results directory.

In the ADE mode, you can set the `ignoreTokenContext` variable to `false` so that expressions from the saved graphs are evaluated in the context of the results directory in which the graph was saved.

MDL Calculator

When you open a saved graph that contains expressions, the expressions are evaluated within the context of the results directory in which the graph was saved.

Working With Assistants

The Virtuoso Visualization and Analysis XL includes the following assistants:

- Spectrum on page 134
- Browser on page 142
- Marker Toolbox on page 144
- Eye Diagram on page 144
- Horiz Marker Table on page 154
- Trace Info on page 154

Note: You can hide or show assistants by using the `F11` key. Alternatively, click *the Toggle Assistants Visibility* button on the Workspace toolbar.

Spectrum

The Spectrum assistant is used to plot and calculate the Fast Fourier Transform (FFT) of a periodic waveform and its different measurements—Signal-to-Noise-and-Distortion Ratio (SINAD), Spurious Free Dynamic Range (SFDR), Effective Number of Bits (ENOB), and Signal-to-Noise Ratio (SNR without distortion) ENOB, SINAD, SNR, SFDR, THD, sigpower, thddb, totalharmpower, peakharmpower, snb, snrh, dcpower

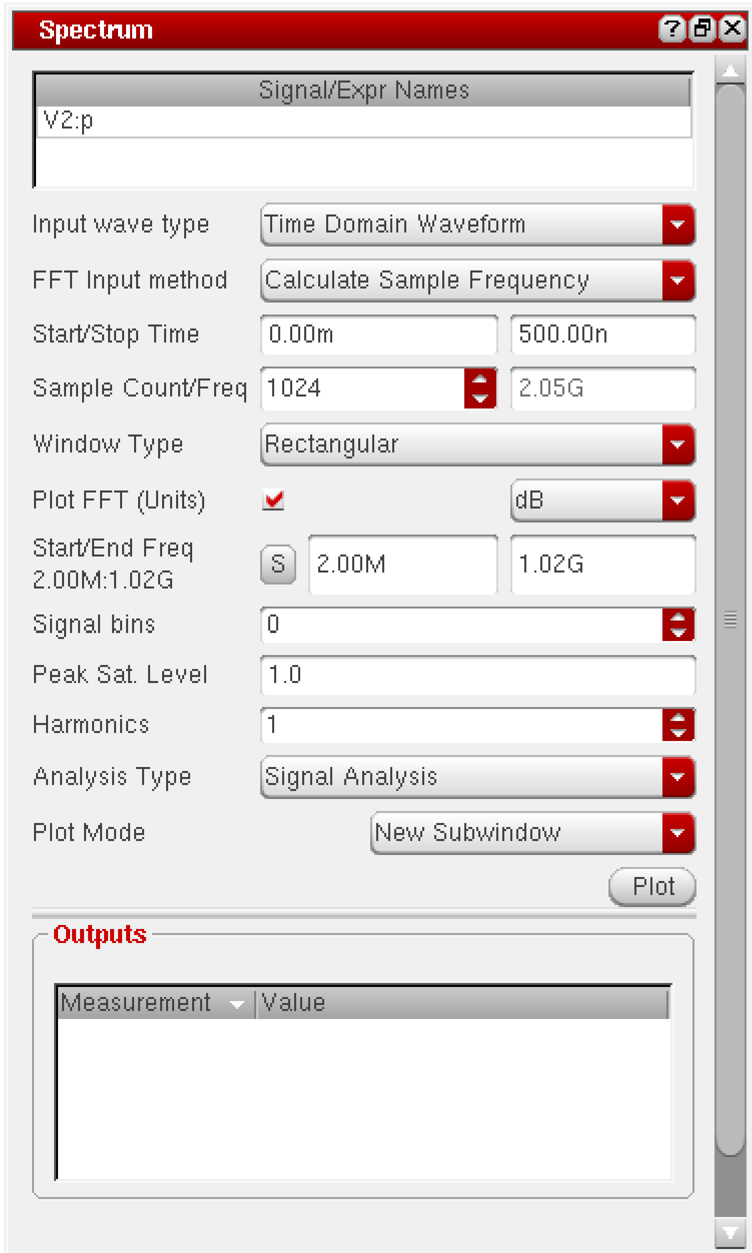
—for a given input signal. The spectrum measure is used for characterizing A-to-D converters and is typically supported for transient simulation data.

Note: In Virtuoso Visualization and Analysis XL, Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) are the same.

To open the Spectrum assistant, select a signal in the window and do one of the following:

- Choose *Window – Assistants – Spectrum*.
- Choose *Measurements – Spectrum.*

The Spectrum assistant appears on the left in the window by default.



The *Signal/Expr Names* field in the Spectrum assistant displays the name of the selected traces. The traces are displayed on the basis of their selection order; however, you can rearrange the trace order either by clicking the column header or by using the drag operation. The assistant has the following fields:

- *Input Wave Type*—Select the input wave type as `Time Domain Waveform` or `Frequency Domain Waveform`.

You can calculate the FFT for the time domain waveform. However, the frequency domain waveform is already an FFT waveform and is used only to calculate measurements. You can use the frequency domain waveform if you are running simulation with the Spectre Fourier component. The Spectre Fourier component performs a Fourier integral and outputs the results in the frequency domain.

- *FFT Input Method*—Select the Fast Fourier Transform (FFT) input method from the drop-down list box.

  If you select *Calculate Sample Frequency*, which is the default option, you need to specify the start and stop time. The Sample Count and Frequency fields display the values calculated based on the specified start and stop time.

  `Start Time` = Start time of the waveform
  `End Time` = End time of the waveform
  `Sample Frequency = SampleCount/(StopTime - StartTime)`

  If you select *Calculate Start Time*, you need to specify the stop time, sample count, and frequency. The *Start Time* field displays the value calculated based on the specified stop time, sample count, and frequency.

  `StartTime= StopTime - SampleCount/SampleFreq`

  If you select *Calculate Stop Time*, you need to specify the start time, sample count, and frequency. The *Stop Time* field displays the value that is calculated based on start time, sample count, and frequency that you specify.

- *Start/Stop Time*—Specify the start and stop time of the input time domain periodic waveform.

- *Sample Count/Freq*—Specify the sample count that is used to determine the number of frequency bins in the FFT waveform. Also, in the *Freq* field, specify the sampling frequency that determines the size of the frequency bin of the FFT waveform.

  The *Sample Count* value must be greater than zero and can include any integer that is a power of two. For a value that is not a power of two, the function rounds it up to the next closest power of two. By default, this field displays the number of data points in the selected signal.

  `SampleFrequency=SampleCount/(StopTime-StartTime)`

  `StartTime=StopTime-(SampleCount/SampleFreq)`

  `StopTime=Startime+(SampleCount/SampleFreq)`

The number of input periods in the period of the Fourier transform needs to be an integer and a prime number, where the period of the Fourier transform is 1/ the fundamental frequency of the Fourier transform.

- *Window Type*—Specify the window function that you can apply to the waveform, such as `Hanning` or `Kaiser`. By default, this field displays `Rectangular`. Some of these window types increase the amplitude of the signal. This helps in distinguishing the noise in the signal.

  You can use `Rectangular` window type when you have an integer number of cycles. However, if you are simulating a sigma-delta modulator, you can use `Hanning` instead of `Rectangular` even when an integer number of cycles occur between `StartTime` and `EndTime`.

  For Nyquist Rate converters, use `Rectangular` window. For Delta-Sigma converters, use `Cosine2 (Hanning)` window instead of Rectangular even when an integer number of cycles occur between `startTime` and `endTime`.

  In D/A converters, the fourier component is from `analogLib`. The output of a D/A converter is an analog signal, which is continuous. When Fourier Transform samples the input, some information can be lost; therefore, the Integral is used.

- *Plot FFT (units)*—Select this check box if you want to plot the FFT waveform. In the drop-down list to the right, select the values you want to plot in the FFT waveform. The values can be `dB`, `Imaginary`, `Magnitude`, and `Real`.

- *Start/End Freq*—Specify the lower limit and the upper limit of the frequency range for spectrum measures. By default, the lower limit field displays the first frequency point of the FFT and the upper limit field displays the last frequency point of the FFT.

  To synchronize the start and end frequency values with the specified start and stop time, click the `S` button. You can also change the start and end frequency values, if required.

- *Signal Bins*—Specify the number of signal bins. When you select a window type, this field displays the default number of bins for the selected window type. For example, if you select the *Window Type* as `Kaiser` that has two signal bins, this field displays `2`. You can increase the number of signal bins to up to half the value of the sample count. For example, if the sample count is `16` for the window type `Kaiser`, you can increase the signal bin count in the *Signal Bins* field up to `8`. You cannot decrease the displayed signal bin value.

  By default, this field displays zero to indicate the rectangular window type. This specifies the number of bins on each side of the signal bin or harmonic bin of the FFT waveform that are to be considered as part of the signal or harmonic.

  Signal bin for `Hanning`, `Hamming`, `Cosine2` windows is 1.

Signal bins for `Blackman`, `ExtCosBell`, `Kaiser`, `Cosine4` signal bin are 2.

Signal bins for `HalfCycleSine`, `Half3CycleSine`, `HalfCycleSine3`, `Half6CycleSine`, `HalfCycleSine6`, `Parzen` signal bin are 3.

Bins are used to calculate the total signal power, `P_Noise`, and `P_distortion`. The `startBin` and `endBin` values are obtained from the start and end frequencies.

If you change the number of the signal bins, the measurements in the Outputs section are changed. For example, if you change the value of signal bins, the signal power and noise power are also changed. This also changes the SINAD and ENOB values.

- *Peak Sat. Level*—Specify the peak saturation level of the FFT waveform. Magnitude of the FFT wave is divided by the Peak Sat Level before using it in calculations. Peak sat level is the full-scale span ignoring any DC offsets and used in ENOB calculation.
  Valid values: Any floating point number.
  Default value: If the peak sat level is not specified or `nil`, it is assumed to be 0 and is taken to be the peak-to-peak value of the fundamental.

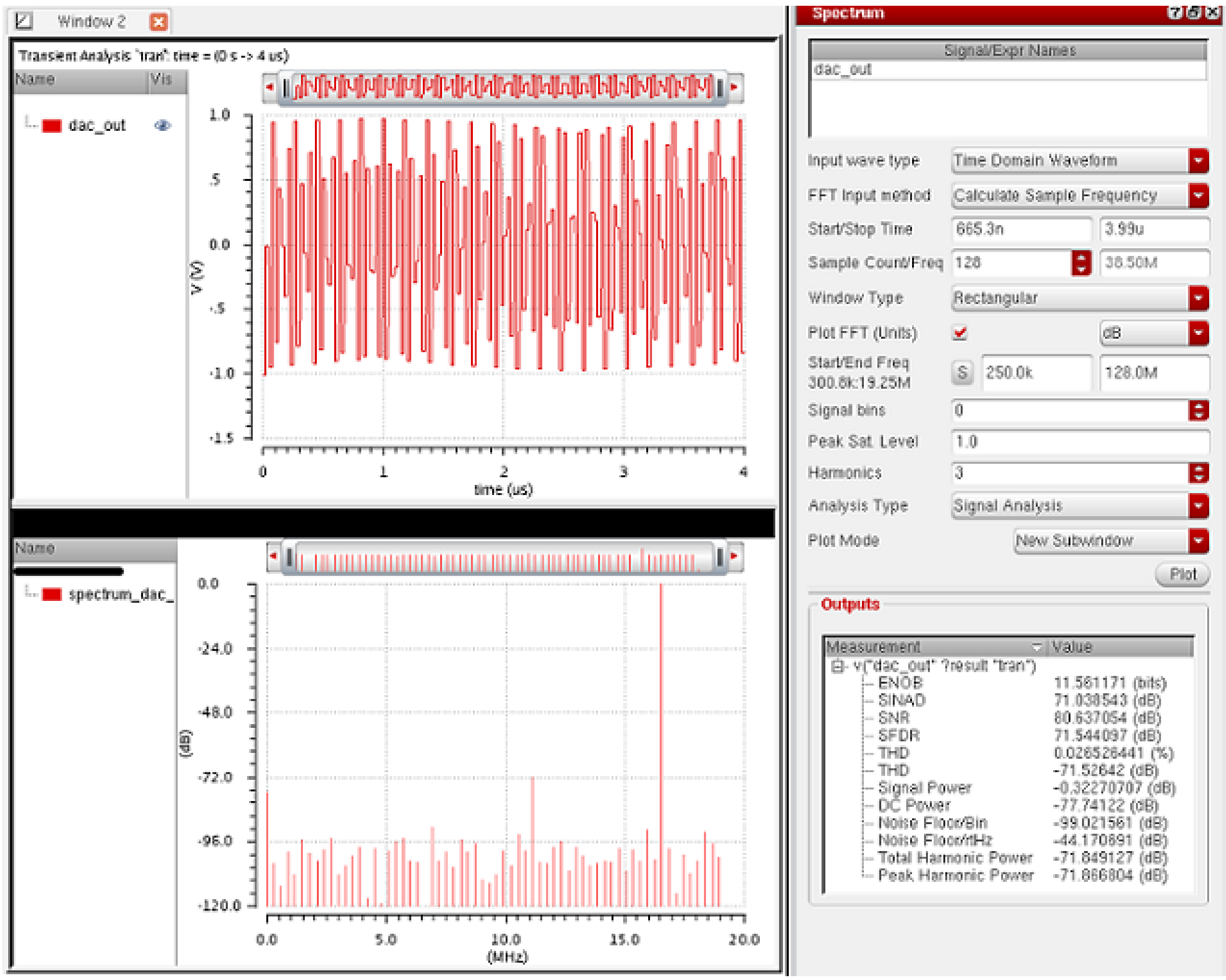
- *Harmonics*—Specify the number of harmonics for the waveform that you want to plot. For example, If this variable is n, where n should be greater than 1 and the fundamental frequency is harmonic 1, the n harmonics are considered for the harmonic power calculation. The signal bins are used for calculating the harmonic power.

  For example, to calculate the total harmonic distortion (THD), if you set the *Harmonics* value to `n`, where `n` is greater than `1`, and the fundamental frequency is harmonic 1, the number of harmonics used to calculate THD is `2`,...,`n`. If `n=3`, the `2`nd and `3`rd harmonics are used to calculate THD.

- *Analysis Type*—Specify the analysis type as `Signal Analysis` or `Noise Analysis`. Based on the analysis type that you select, the measurement values are calculated and displayed in the *Outputs* section. If you want to use only the noise in the frequency domain waveform as input, select `Noise Analysis`. However, if you want to use both the signal and noise in the frequency domain waveform as input, select `Signal Analysis`.

- *Plot Mode*—Specify whether you want to append the FFT waveform to the existing graph, replace the existing graph, or plot the waveform in a new window or a subwindow.

- ❑ *Plot*—Click the *Plot* button to plot and evaluate the FFT waveform and its measurements.



Outputs

The *Outputs* section displays the following measurements and their values:

- *SINAD* (Signal to Noise and Distortion Ratio)
- *SNR* (Signal to Noise Ratio)
- *SFDR* (Spurious Free Dynamic Range)
- *ENOB* (Effective Number Of Bits)
- *Signal Power*

- *DC Power*
- *Noise Floor/Bin*
- *Noise Floor/rtHz*
- *Total Harmonic Power*
- *Peak Harmonic Power*

Calculations

- ENOB (Effective Number Of Bits)

  If Peak Sat. Level is not given, which means it is 0, ENOB is calculated as below:

  `ENOB = (SINAD - 1.763)/6.02`

  If Peak Sat Level is given, a correction factor is considered as below

  `ENOB = (SINAD - 1.763 - dB10(Max_Signal_Power))/6.02`

  `Correction factor = Power of fundamental/ Power of (1/2 *peakSatlevel)`

- SINAD (Signal to Noise and Distortion Ratio)—SINAD is the ratio of total signal power to the noise-plus-distortion power.

  `SINAD= P_Signal / (P_Noise + P_distortion)`

  - `P_Signal` is the total signal power and can be calculated as below:

    `P_Signal = arr[i] * arr[i] ;`

    `arr[i]` is an array and, `i` is from `(freq - signalBin)` to `(freq + signalBin)`

    `freq` = Frequency of the fundamental bin. The bin that has the maximum Y value is called the fundamental bin. If the input waveform is in the time domain, the FFT of the signal is considered.

  - `P_Noise` is the power of the noise excluding the power around the fundamental bin, the power around the harmonic bins, and the power at DC.

    `P_Noise = arr[i] * arr[i] ;`

    where, `i` is an array index that does not include values from `(freq - signalBin)` to `(freq + signalBin)`.

  - `P_distortion = noiseNonHarmPwr`

`P_distortion` is the power of the distortion spurs up to the number specified by Harmonics. The algorithm also calculates the location of the aliased harmonics and includes the aliased harmonic power.

- SNR (Signal to Noise Ratio)

  `SNR = dB10(P_Signal/TotalNonHarmNoise)`

- SFDR (Spurious Free Dynamic Range)

  SFDR is the strength ratio of the fundamental signal to the strongest spurious signal in the specified frequency band set by the Start and End frequency fields.

  `SFDR = dB10(Max_Signal_Power/max(maxNoise, maxHarmNoise))`

  `Max_Signal_Power` is the maximum value of y-vector in the total signal power calculation.

  `maxNoise = Max of arr[i];`

  where `i` is the index of array elements that were not considered while calculating total signal power, total harmonic power and power at DC. `MaxNoise` is the maximum noise component.

  `maxHarmNoise = Max(arr[i]) ;`

  where, `i` is the index of the harmonic.

- THD (Total Harmonic Distortion)

  `THD = 100 * sqrt(total_Harmonic_Pwr/P_Signal)`

  where, `total_Harmonic_Pwr` is the sum of all harmonic powers

- Signal Power

  `SignalPower = dB10(P_Signal)`

- DC Power

  `dc_power= arrI[0]*arrI[0]+arrR[0]*arrR[0]`

  `arrI` is Imaginary part and `arrR` is Real part

- Noise Floor/Bin

  `NoiseFloor/Bin = dB10(TotalNonHarmNoise/anaBins)`

  `noiseNonHarmPwr = Sum of arr[i] * arr[i];`

where, `i` is the index of array elements which were not considered in the calculation of total signal power (the fundamental), total harmonic power, and DC.

```
Average [noiseNonHarmPwr] = (noiseNonHarmPwr / n)
```

where, `n` is the total number of bins used

```
TotalNonHarmNoise = (noiseNonHarmPwr / n) * anaBins
```

`anaBins` is length of the array `arr[]`

- Noise Floor/rtHz

```
NoiseFloor/rtHz = dB10((TotalNonHarmNoise/(anaBins-1))/binSize)
```

`binSize` is difference of the two nearest X points `(frequency - previousFrequency)` in the FFT wave.

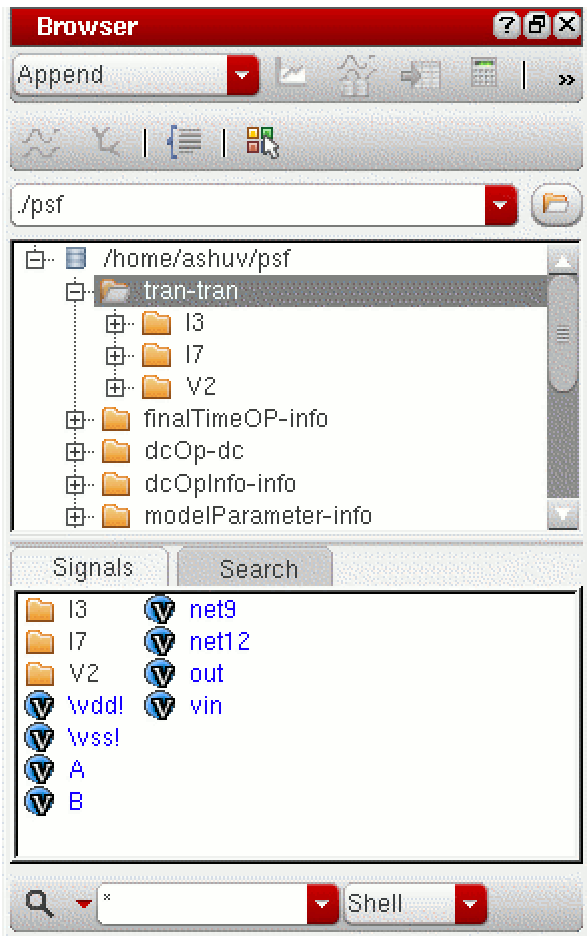
Important

When you send the computed measurement values from the Spectrum toolbox to ADE Outputs and create an expression for them using ADE, the `spectrumMeasurement` function is used in the expression. For more information about spectrumMeasurement functions, see spectrumMeasurement in the *OCEAN Reference*.

Browser

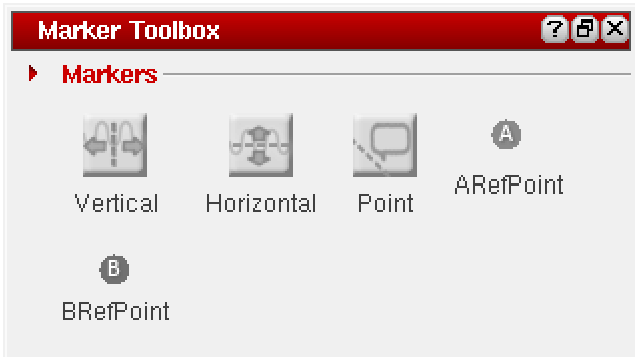
The Browser assistant displays the Results Browser that you can use to open simulation results saved earlier. To open the Results Browser, choose *Window – Assistants –*

Browser. For more information about the Results Browser, see Chapter 2, "Using the Results Browser,"



Marker Toolbox

Use this assistant to add point, vertical, horizontal, and reference point (ARefPoint or BRefPoint) markers to the trace. For more information about Marker Toolbox, see Adding Markers with Marker Toolbox on page 258.



Eye Diagram

You use the Eye Diagram assistant to create an eye. An eye diagram is a way of representing a digital data signal by repetitively sampling the signal and overlaying the repeated samples on the same X-axis.

The result is a plot that has many overlapping lines enclosing an empty space known as the eye. The quality of the receiver circuit is characterized by the dimension of the eye. An open eye means that the detector can distinguish between `1's` and `0's` in its input, while a closed eye means that a detector placed on `Vout` is likely to give errors for certain input bit sequences.

Example Circuit

The example described below is generated by using a Pseudo-Random Binary Sequence (PRBS) source in the Spectre and Verilog-A model that can add random amplitude variation and random delays to the edges.

Spectre Netlist prbs.scs

```
// prbs.scs
v1 (high 0) vsource type=prbs period=40n val0=0 val1=1 rise=0.3n \
   fall=0.3n delay=3n
vclk (clk 0) vsource type=pulse period=80n val0=0 val1=1 rise=0.3n \
   fall=0.3n delay=3n
```

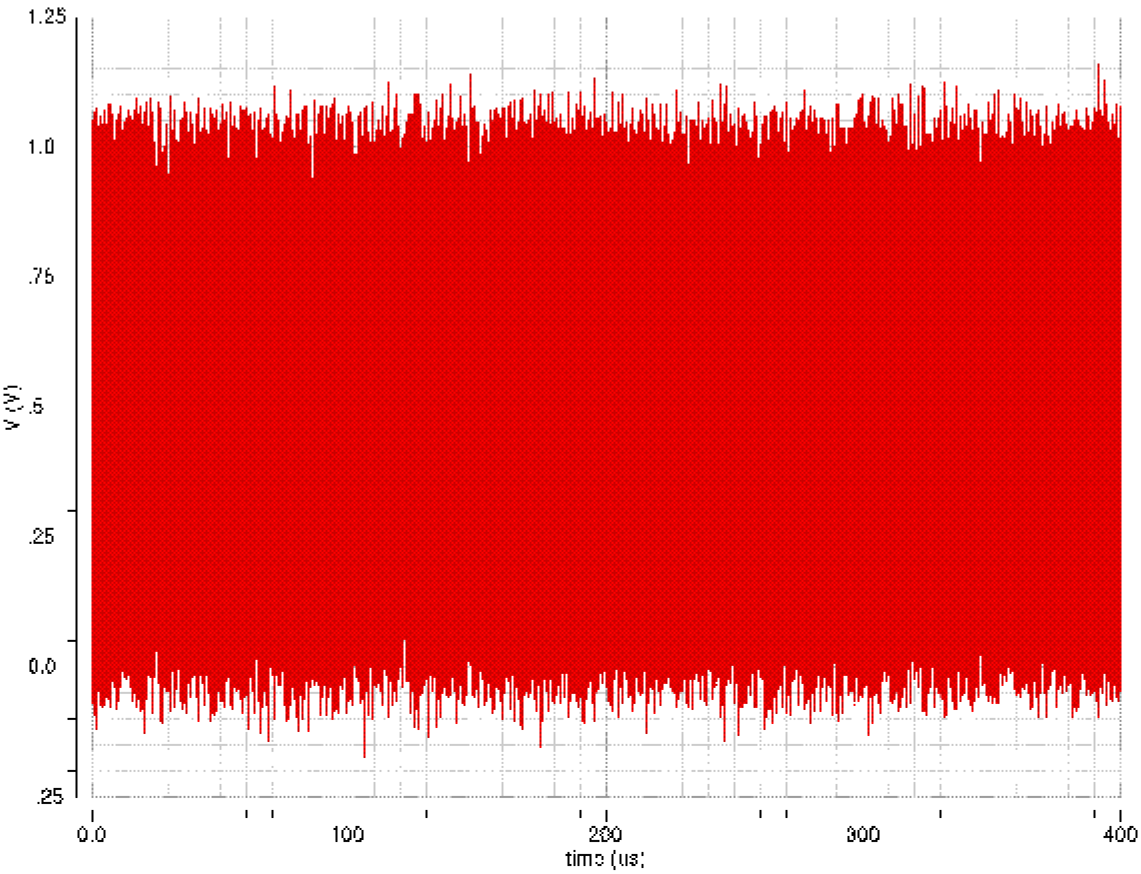
```
del (withdelay high) randdelay sd=1n rise=0.3n
rl (withdelay jitter) resistor r=100
cl (jitter 0) capacitor c=50p
r1 (high 0) resistor r=1k
ahdl_include "randdelay.va"
tran tran stop=400u
```

Verilog-A model randdelay.va

```
// VerilogA for randdelay
`include "constants.h"
`include "discipline.h"
module randdelay (op,ip);
output op;
input ip;
electrical op,ip;
parameter real sd=2.0n;
parameter real gainsd=0.1;
parameter real del=5.0n;
parameter real rise=0.1n;
parameter real thresh=0.5;
parameter integer seed = 23133;
integer vseed;
real randnum,gain,delayed;
analog begin
@(cross(V(ip)-thresh)) begin
    randnum=$rdist_normal(vseed,0,1);
    randnum=randnum*sd+del;
    gain=$rdist_normal(vseed,0,1)*gainsd+1;
    if(randnum<0) randnum=0.0;
  end
delayed= transition(V(ip),randnum,rise,rise);
V(op) <+ gain*(delayed-thresh)+thresh;
end
endmodule
```

Note: To run the simulation in the example shown above, type `spectre prbs.scs`.

When you plot the `jitter` signal as shown in the previous example, which is the output of the PRBS source that has passed through the Verilog-A model to add random delay and amplitude variation and then passed through a simple RC filter, you get the following plot:



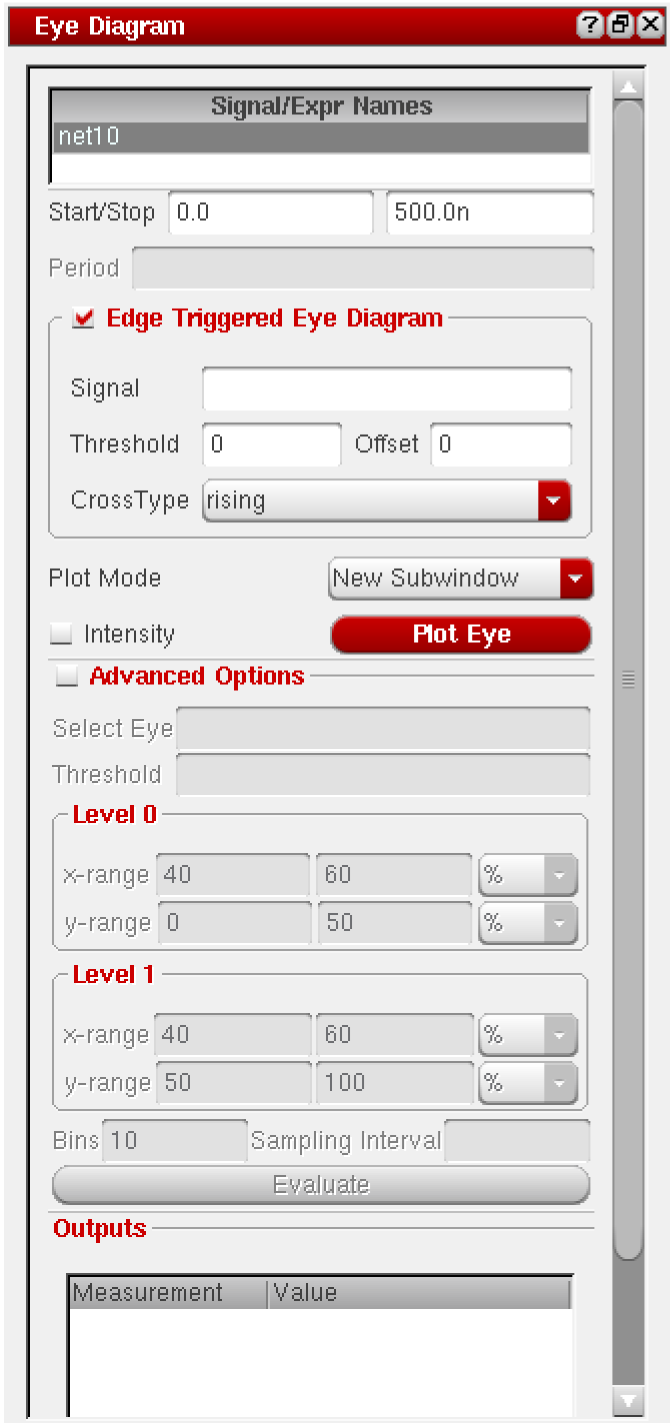
This plot does not show how much the original signal is distorted. The zoomed in waveform also does not help visualize the timing and amplitude variation in the data. These issues can be resolved by using the Eye Diagram assistant.

Opening the Eye Diagram Assistant

Select a signal in the window and do one of the following to open the Eye Diagram assistant:

- Choose *Measurements – Eye Diagram*.
- Choose *Window – Assistants – Eye Diagram.*

The *Eye Diagram* assistant appears on the right in the window.



The Eye Diagram form has the following fields:

- ❑ The *Signal/Expr Names* field displays the name of the traces that you select in an active window. The traces are displayed on the basis of their selection order; however, you can rearrange the trace order either by clicking the column header or by using the drag operation. To delete the traces that are not required, select the traces and press the `Delete` key.

  Note: You can also drag signals from the Results Browser and directly send them to the Eye Diagram *Signal/Expr Names* field.

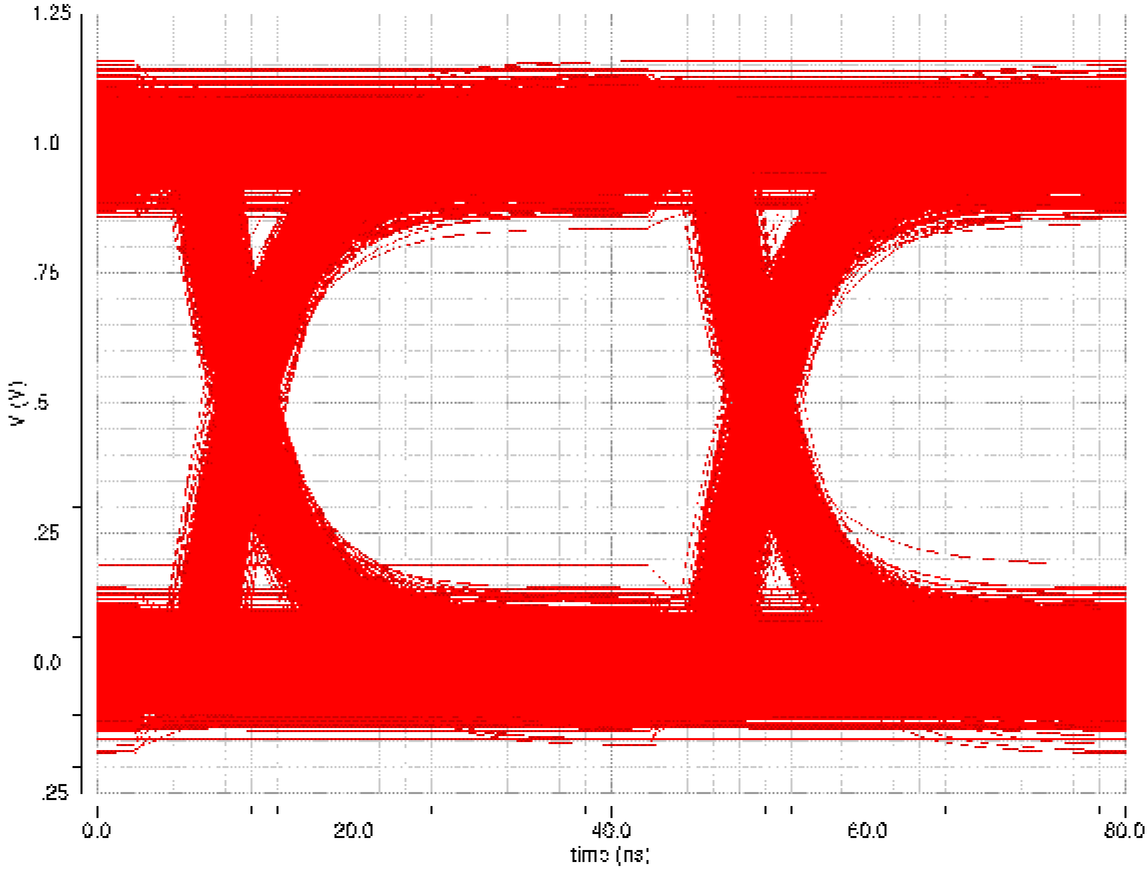
  Note: You cannot add signals to the *Signal/Expr Names* field that do not result in generating an eye diagram.

- ❑ In the *Start* field, specify the X-axis value from where the eye diagram plot needs to be plotted.
- ❑ In the *Stop* field, specify the X-axis value where the eye-diagram plot must end.
- ❑ In the *Period* field, specify the time period for the eye diagram. This is the period after that the X-axis starts repeating.
- ❑ Select the *Edge Triggered Eye Diagram* check box if you want another signal to be triggered at the beginning of the eye diagram instead of a fixed period.

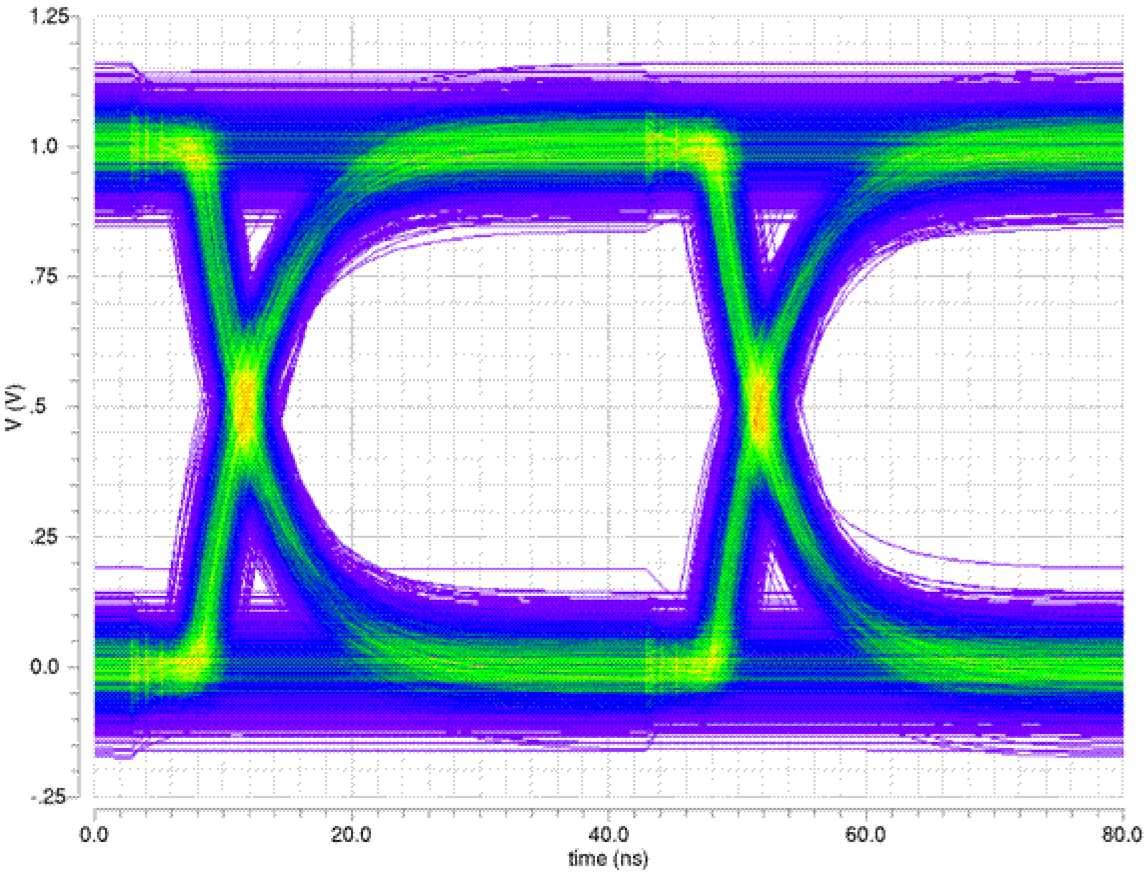
  For more information about how to plot the edge triggered eye diagram, see Edge Triggered Eye Diagram on page 150.

- ❑ In the *Plot Mode* list, specify whether you want to *append* the eye diagram to an existing graph, *replace* an existing graph with the eye diagram, or add the eye diagram to a *new subwindow* or a *new window*.
- ❑ Select the *Intensity* check box to highlight the intersection points of the eye diagram.
- ❑ Click the *Plot Eye* button to plot the eye diagram for the selected signal. By default, the name of the eye diagram is *eye_<signal-name>.*

If you do not select the *Intensity* check box and click *Plot Eye* to plot the graph, the following graph is opened in a new subwindow:



If you select the *Intensity* check box, the graph is displayed as shown below:



This graph helps find the extent of the amplitude variation and timing variation in the transitions. Colors displayed in the graph are used to show regions of greater density, which helps visualize the distribution of amplitude and timing variation.

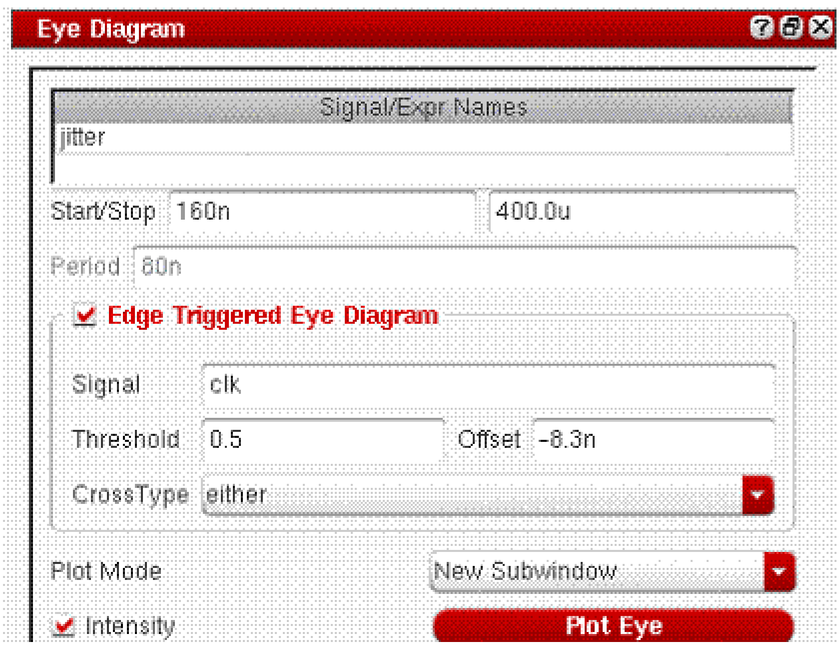
The hole in the middle of the graph is known as eye. The circuit or the system is considered bad if the hole is small.

Edge Triggered Eye Diagram

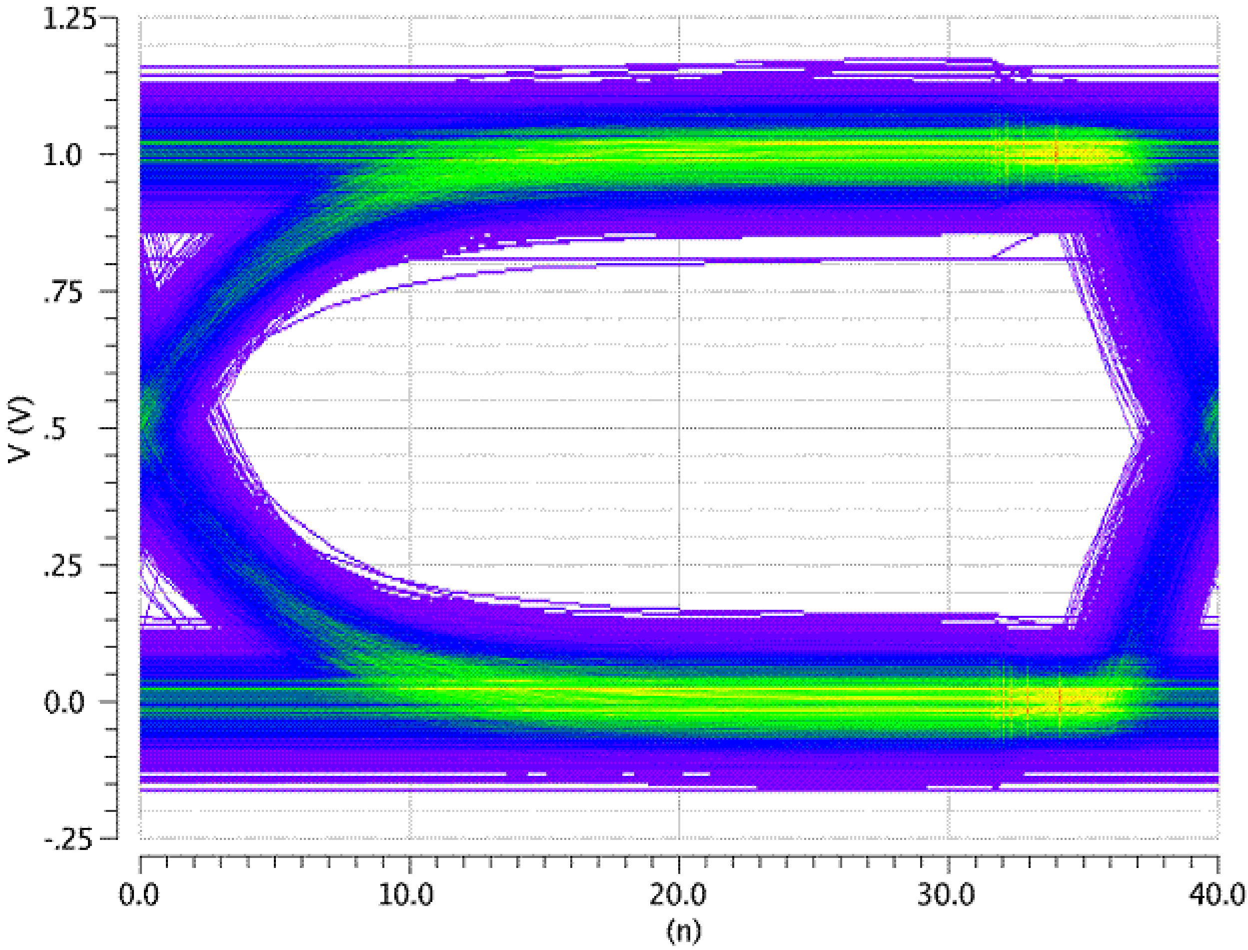
You can select the *Edge Triggered Eye Diagram* check box in the Eye Diagram assistant if you want another signal to be triggered at the beginning of the eye diagram instead of a fixed period. When you select this check box, the *Period* field becomes inactive. You also need to plot the signal that you want to use as a triggering signal.

When you select this check box, the *Signal*, *Threshold*, *Offset*, and *CrossType* fields are enabled.

- Click in the *Signal* field and select a signal that you want to use as a triggering signal. The selected signal name appears in this field.
- In the *Threshold* field, specify a threshold value. When a signal to be triggered crosses the specified threshold value in a given direction, the eye diagram starts a new period. You can specify direction in the *CrossType* field.
- In the *Offset* field, specify an offset value that is used to shift the phase of the eye. If you specify a positive value, the eye right is rotated by the specified amount of time and if you specify a negative value, the eye left is rotated by the specified amount of time.
- Select the *CrossType* as `rising`, `falling`, or `either`.



When you click the *Plot Eye* button, the edge triggered eye diagram for the above specified values is plotted in a new subwindow as shown in the figure below:



Advanced Measurements

After you plot an eye diagram for the selected signal, you can specify the advanced options for the eye diagram. To specify the advanced options, select the eye diagram plot and select the *Advanced Option*s check box. This evaluates the vertical (`Max Vertical Opening Level 0`) or horizontal opening (`Max Horizontal Opening Level 1`) of the eye diagram.

The advanced measurements that can be calculated are based on performing statistical analysis of the eye diagram. Consider the signal you want to analyze as a sequence of 1's and 0's, with transitions between the two logic levels. As a result, it is required to analyze the distribution of the timing of transitions between 0 and 1 levels, and also to analyze the distribution of signal levels of the parts of the eye diagram representing 0 and 1 values.

The *Advance Options* section includes the following fields:

- The *Selected Eye* field displays the name of the selected eye diagram.
- In the *Threshold* field, specify the threshold value. The Y-axis level, such as voltage, represents the switching threshold of the signal is generally half of the signal range. This is used to compute the statistical information about the times (relative to the beginning of the eye diagram) at which the signal crosses the threshold level.
- In the *Level 0* field, specify X-range and Y-range to plot their vertical histograms for level 0 measurements.
- In the *Level 1* field, specify X-range and Y-range to plot their horizontal histograms for level 1 measurements.
- Click the *Evaluate* button to evaluate the vertical and horizontal opening of the eye diagram.
- In the *No of bins*, specify the signal bins you want to display in the eye diagram plot. These signals bins are used to form the horizontal (threshold crossing times) and vertical (amplitude variation) histograms.
- In the *Sampling Interval*, specify the time interval after which the signals are divided in the eye diagram plot. If this field is left blank, the data within the level 1 and level 0 regions are used to analyze the amplitude variation of the signal. This means there is some sensitivity to the actual spacing between the data points in the signal, which is caused by the variable time steps in the simulator. If the points are clustered in the curve portion, the distribution can be skewed. To perform the analysis, the sampling interval you specify in this field is divided into even time points.

Outputs

After the evaluation is complete, the following additional outputs are displayed in the *Output*s section of the Eye Diagram assistant:

- *Eye width*
- *Eye height*
- *Eye amplitude*
- *Eye SNR*
- *Eye Rise Time*
- *Eye Fall Time*
- *Level 1 standard deviation and mean*

- *Level 0 standard deviation and mean*
- *Threshold crossing stddev*
- *Threshold crossing average*

Setting Eye Diagram Properties

To change the trace properties of an eye diagram, right-click the eye diagram plot and choose *Trace Properties*. The *Eye Trace Properties for <eye_diagram_name>* form appears. It includes the following fields:

- *Name*—Specify the name of the eye diagram trace. If you select the *Default* check box, it displays the default trace name.
- *Expression*—Specify an expression that you can associate with the selected eye diagram.
- *Color*—Specify the foreground color of the trace.

Horiz Marker Table

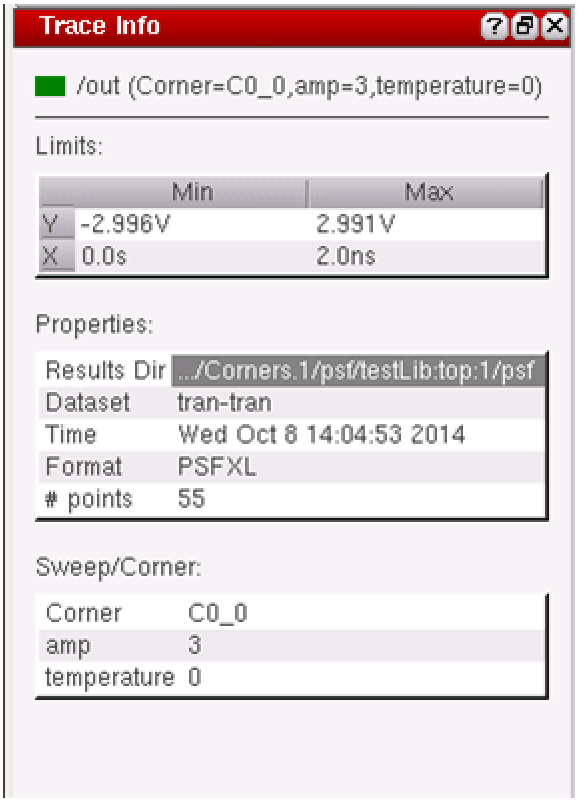
You use this assistant to view the interception data for horizontal markers in a table. For more information, see Displaying Intercept Data for Markers in Marker Tables on page 265.

Trace Info

You use this assistant to view information about the selected trace. To open the Trace Info, select a trace and choose *Window – Assistants – Trace Info*. The *Trace Info* assistant appears, displaying the information, such as *trace name and color, Y Min, Y Max, X Min, X Max, Time, Results dir, Dataset, Time, Data Format,* and *Number of Data points,* about the selected signal. It also displays information about the sweep and corner conditions for the selected trace.

To copy the trace properties, right-click the property and choose *Copy*.

For more information about the data displayed in the trace panel, see Working with Traces on page 176.



Vert Marker Table

You use this assistant to view the interception data for vertical markers in a table. For more information, see Displaying Intercept Data for Markers in Marker Tables on page 265.

Transient Measurement

The Transient Measurement assistant displays the calculated measurements for the transient markers on specific edges. The measurements can be falltime, risetime, overshoot, undershoot, and slewrate values. You can also generate the derived plots for rising and falling edges by using this assistant.

To enable transient measurement, set the following environment variable in CIW:

```
envSetVal(viva.rectGraph "enableEdgeMeasurement" 'string "true")
```

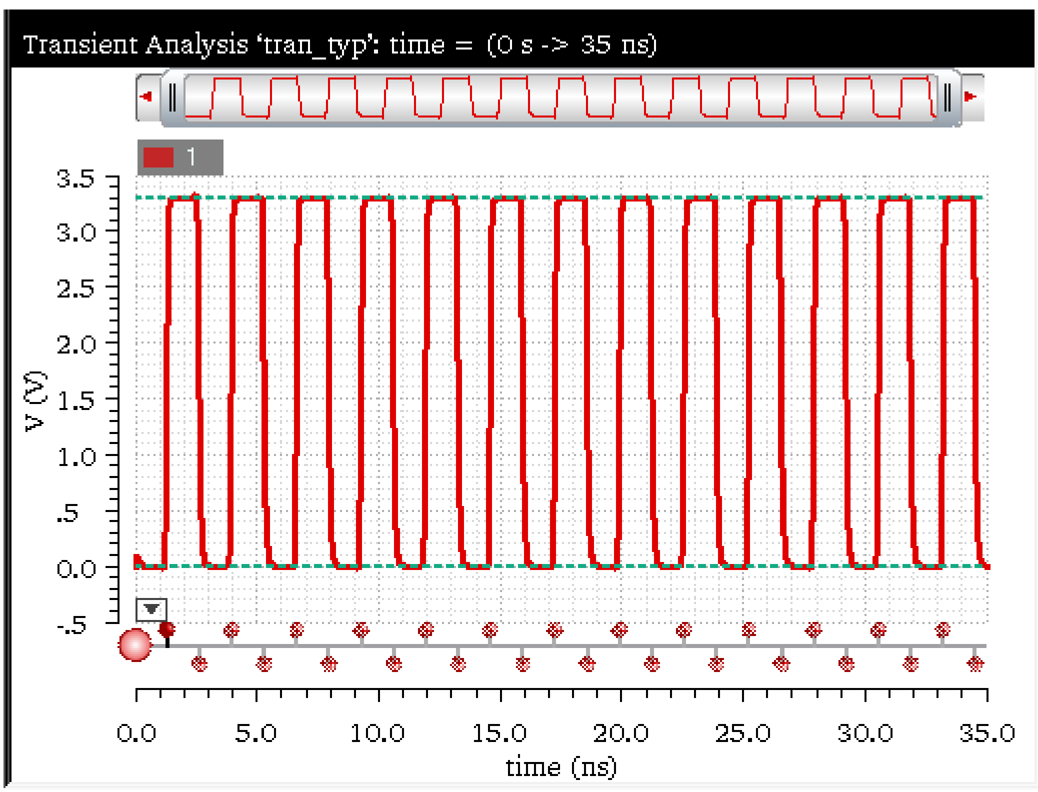
Note that this variable is by default set to `false`.

Before you start using the Transient Measurement assistant, refer to the following topics to understand some basic concepts:

- Overview on page 156
- Specifying Edge Settings on page 157
- Using Edge Browser on page 159
- Context-Sensitive Menus on page 161
- Using the Transient Measurement Assistant on page 163
- Using Markers on page 166

Overview

When you plot a trace and load the waveform, Virtuoso Visualization and Analysis XL analyses the waveform and generates a set of edges, which are referred as threshold (crossing) points. An edge traverses from a low threshold point to a high threshold point (rising edge) or from a high threshold point to a low threshold point (falling edge). Refer to the next section to know more about how to specify the properties of an edge.



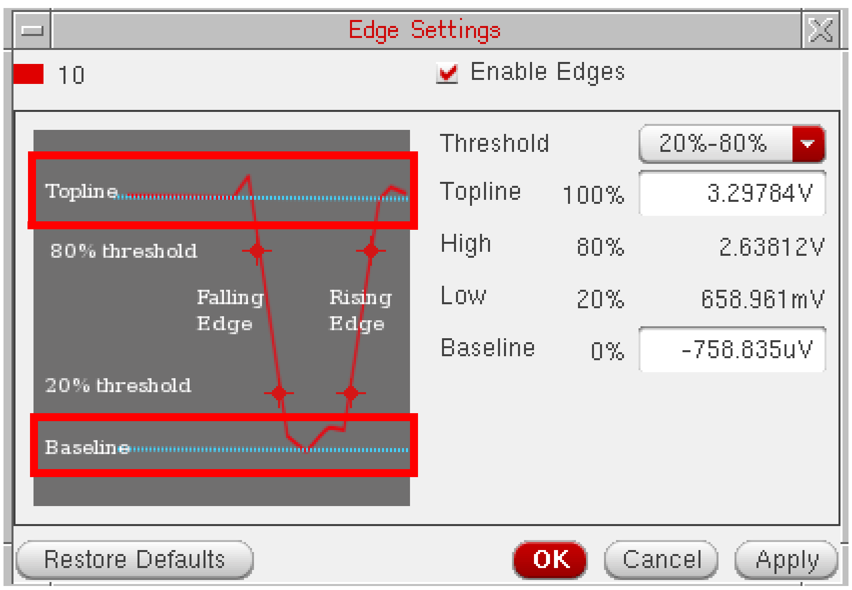
Specifying Edge Settings

The *Edge Settings* form lets you change the edge properties and recalculate the edges.The measurements are displayed in the Transient Measurements assistant.

Do one of the following to open the Edge Settings form:

- Click the *Settings* button on the Transient Measurement assistant.
- Right-click the Edge Browser and choose *Edge Settings*.

The *Edge Settings* form displays a model depicting how the tool calculates edges of the selected trace. The blue reference lines in the model, as highlighted in the figure below, indicate the topline and baseline reference values. You can drag these blue reference lines on the graph to set the topline and baseline values interactively.



This form includes the following fields that you can set to modify settings of the edges for a selected trace:

- *Enable Edges*—Disable this check box if you want to disable the edge calculation for the selected graph. By default, this check box is selected. When this check box is not selected, all fields in Transient Measurement assistant are disabled and the following message is displayed:

  ```
  Edge calculation for current graph disabled.
  To enable, check the Enable Edges checkbox in the Edge Settings dialog.
  ```

- *Threshold*—Select the threshold value from the drop-down. The default threshold value is `20%-80%`.

- *Topline*—Specify the topline (maximum) value. For more information about how to calculate the topline value, see Calculating Topline and Baseline Values on page 158.
- *High*—Displays the maximum threshold value.
- *Low*—Displays the minimum threshold value.
- *Baseline*—Specify the baseline (minimum) value. For more information about how to calculate the baseline value, see Calculating Topline and Baseline Values on page 158.
- *Restore Defaults*—Click this button to fill default values in the fields in the Edge Settings form.
- Click `OK`.

  The specified settings are applied to the selected trace and the trace edges are recalculated. Measurements and edge markers are updated.

Important

The edge settings that you define by using this form are applied only to the selected traces. Edge settings for other traces in the graph follow the settings defined under the following environment variables:

- `viva.trace threshold string "20_80"`
- `viva.trace baseAndToplineReferenceHint string "0.0,3.5"`
- `viva.trace autoReferenceLines string "true"`

Calculating Topline and Baseline Values

A threshold point is where the waveform intersects the Y threshold value. A threshold Y value is defined relative to the distance between the baseline and topline values. For example, for the threshold level of 20% and 80%, low and high threshold Y values are calculated as below:

```
Low threshold Y value == (0.20*(topline-baseline))+baseline
```

```
High threshold Y value == (0.80*(topline-baseline))+baseline
```

The topline (maximum) value is the most frequent value in the upper half of the entire range of trace Y values. Similarly, the baseline (minimum) value will be the most frequent value in the lower half of the entire range of Y values.

By default, the baseline and topline values are calculated automatically using a simple statistical analysis. Note that these values are not actual values and only indicate an estimate. Therefore, there might be cases where these values do not result in generating edges. In such cases, you need to specify these values manually. The following environment variables control the topline and baseline reference values:

```
viva.trace autoReferenceLines string "true"
```

By default this variable is set to `true`, which means Virtuoso Visualization and Analysis XL calculates the topline and baseline reference values automatically. If set to `false`, the values defined by the following environment variable is returned:

```
viva.trace baseAndToplineReferenceHint string "0.0,3.5"
```

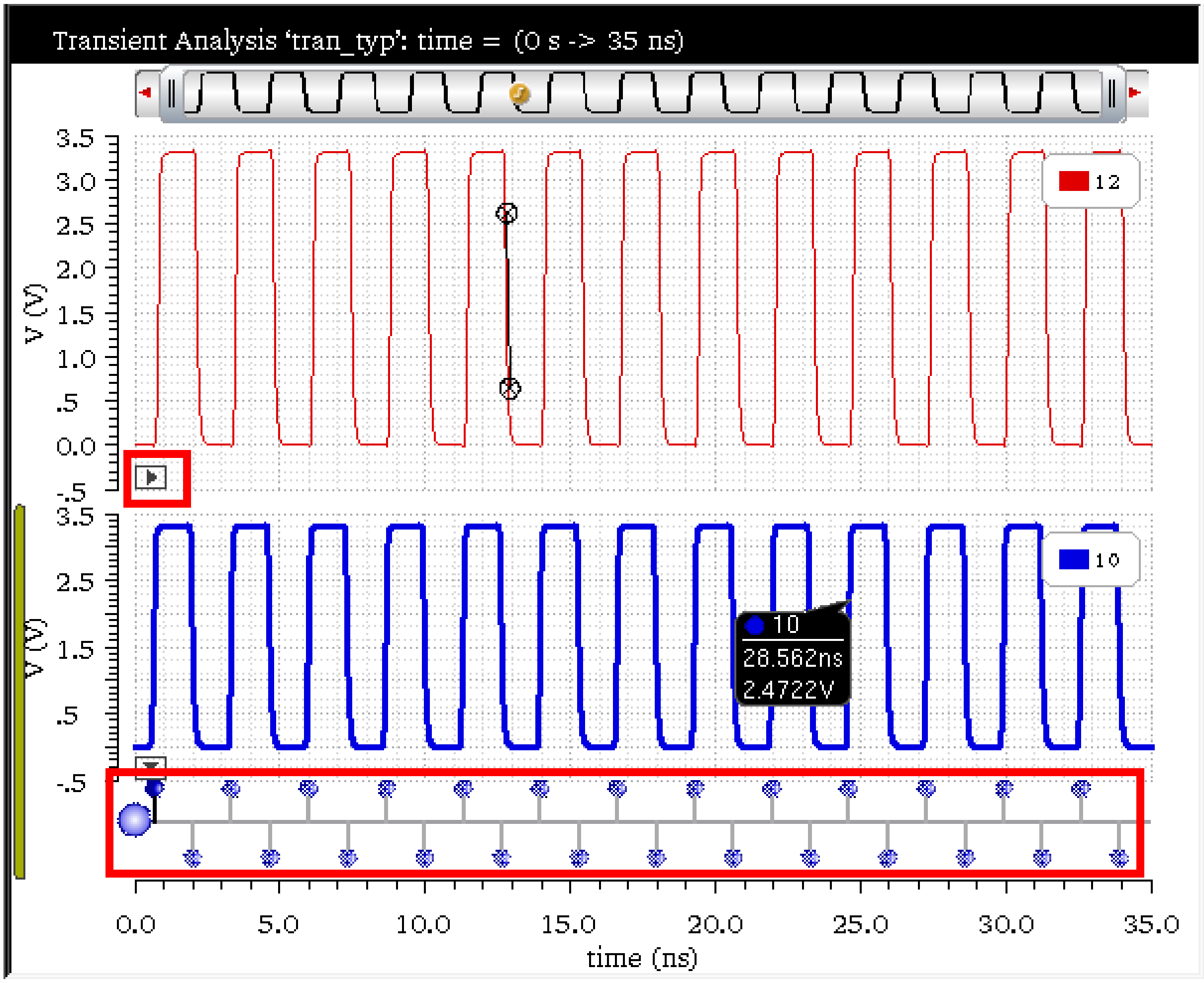
Note: The edge analysis is performed on transient time waveforms only.

Using Edge Browser

You can use the edge browser to view and analyze the entire set of trace edges, zoom into a smaller range of edges, or zoom into a specific edge. You can use the edge context menu options to analyze a specific edge and annotate the edge with different marker types, such as edge, dx/dy, or period markers. Edge Browser also includes an individual context-menu options that you can use to generate risetime or falltime waveforms derived from the entire set of edges.

Each trace is associated with an individual edge browser, which can be displayed and hidden by clicking the Edge Browser button, as shown in the figure below. The edge browser is hidden in the first strip and displayed in the second strip.

Note: The edges in the graph are calculated only when the Edge Browser is open.



Do one of the following to display the edge browser on the graph:

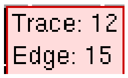
- Click the *Edge Browser* button as displayed in the above figure.
- Choose *Graph – Edge Browser*. This toggles the display of edge browsers for all strips in the graph.
- Right-click the *Edge Browser* button on the strip and choose *Edge Browser.*
- Set the following environment variable to `true`:

  ```
  envSetVal("viva.rectGraph showEdgeBrowser" 'string "true")
  ```

  If this variable is set to `true`, the edge browsers are displayed in all the strips. By default, this variable is set to `false`, which means all the edge browsers are hidden in all the strips.

Note the following:

- When you hover pointer on an edge in the Edge Browser, the tooltip displays the trace and edge number as shown in the figure below:

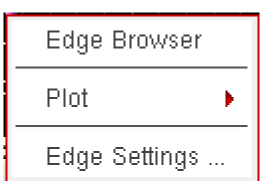
- When the mouse pointer focus is on edge browser, you cannot create horizontal and point markers because these markers require a Y value to be specified. As a result, the bindkeys `m` and `h` do not work when mouse pointer is on edge browser. However, you can create vertical markers by selecting an edge in the edge browser.
- All the edge controls are disabled and the measurement values are cleared in the Transient Measurement assistant to indicate that the edge calculation is in progress.

Context-Sensitive Menus

The two context-sensitive menus are available for edge and edge browser respectively.

Edge Context-Sensitive Menu

The following context-sensitive (shortcut) menu appears when you right-click the edge browser button:

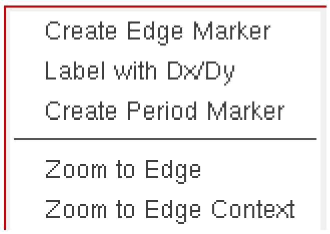


This menu includes the following options:

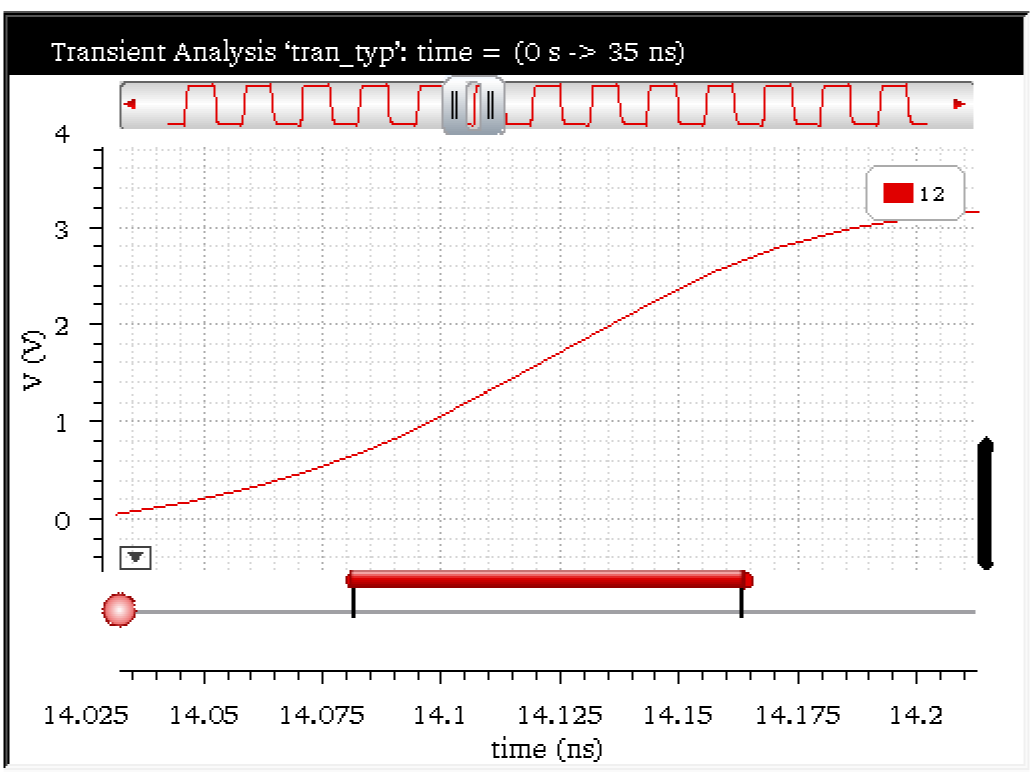
- *Edge Browser*—Displays the Edge Browser for the selected trace.
- *Plot*—Plots the risetime or falltime waveforms.
- *Edge Settings*—Opens the Edge Settings form that you can use to modify the settings of the selected edge. For more information, see Specifying Edge Settings on page 157.

Edge Browser Context-Sensitive Menu

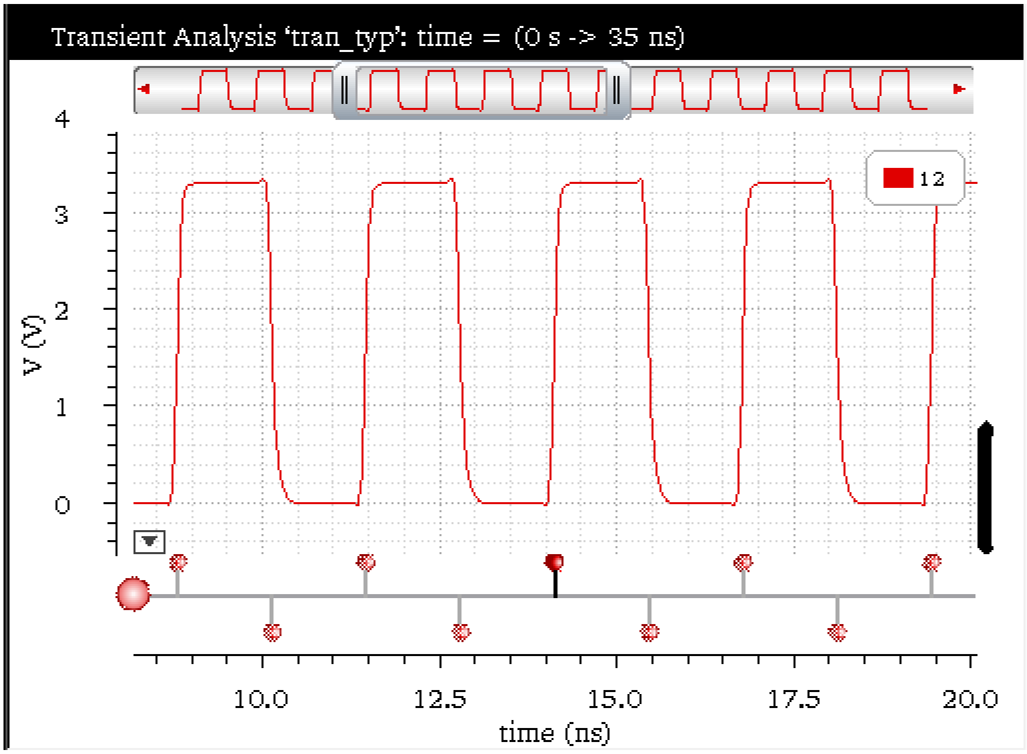
When you right-click an individual edge in the Edge Browser, the following context-sensitive (shortcut) menu appears:



- *Create Edge Marker*—Creates an edge marker on the selected edge. For more information about edge markers, see Working With Edge Markers on page 278
- *Label with Dx/Dy*—Creates a delta marker by adding two point markers at the lower and upper thresholds and a delta line between them.
- *Create Period Marker*—Creates a period marker on the selected edge.
- *Zoom to Edge*—Zooms in the selected edge to view the details of the edge on which it is placed. The following figure illustrates the zoomed-in graph.



- *Zoom to Edge Context*—Zooms in the selected edge to the context of the edge on which it is placed. The following figure illustrates the zoomed-in graph.



Send To—

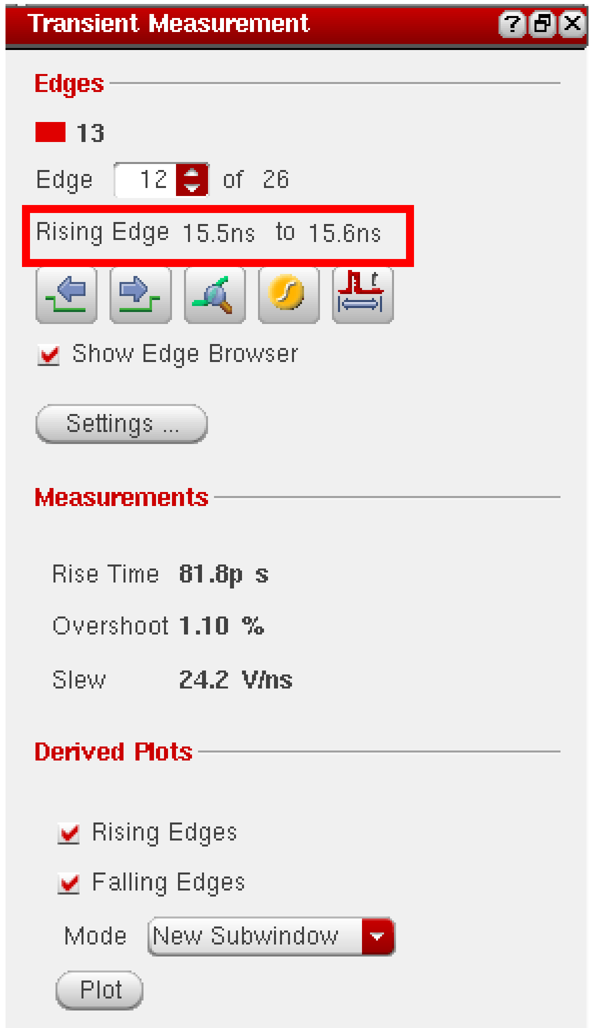
Using the Transient Measurement Assistant

Perform one of the following steps to open the Transient Measurement assistant:

- Choose *Measurements — Transient Measurement*.
- Choose *Window – Assistants – Transient Measurement*.

  Alternatively, you can open the tool in the `TM` workspace to display the Transient Measurement assistant. The workspace can be set using the Workspace toolbar.

The *Transient Measurement* assistant is shown in the figure below:



This assistant includes the following fields:

- *Edges*—This panel lets you navigate between edges and displays the information for the current edge. It also helps you to perform certain operations on the selected edge, such as placing an edge or a period marker, zoom into the edge and so on. This includes the following fields:
    - *Edge*—Select the edge from the combo box for which you want to view the measurements. This also displays the number of total edges in the selected trace. The edge calculation is displayed below the *Edge* combo box, as highlighted in the figure above.

- ❑ The table below lists the icons that you can click to perform certain tasks on the graph:

Button	Description
	Click this icon to move to the pervious edge on the selected trace.
	Click this icon to move to the next edge on the selected trace.
	Click this icon to zoom the graph to view the area of the edge.
	Click this icon to place an edge marker on the selected edge in the graph
	Click this icon to place a period marker on the current edge to the next edge in the same direction. For example, rising-to rising and falling-to-falling.

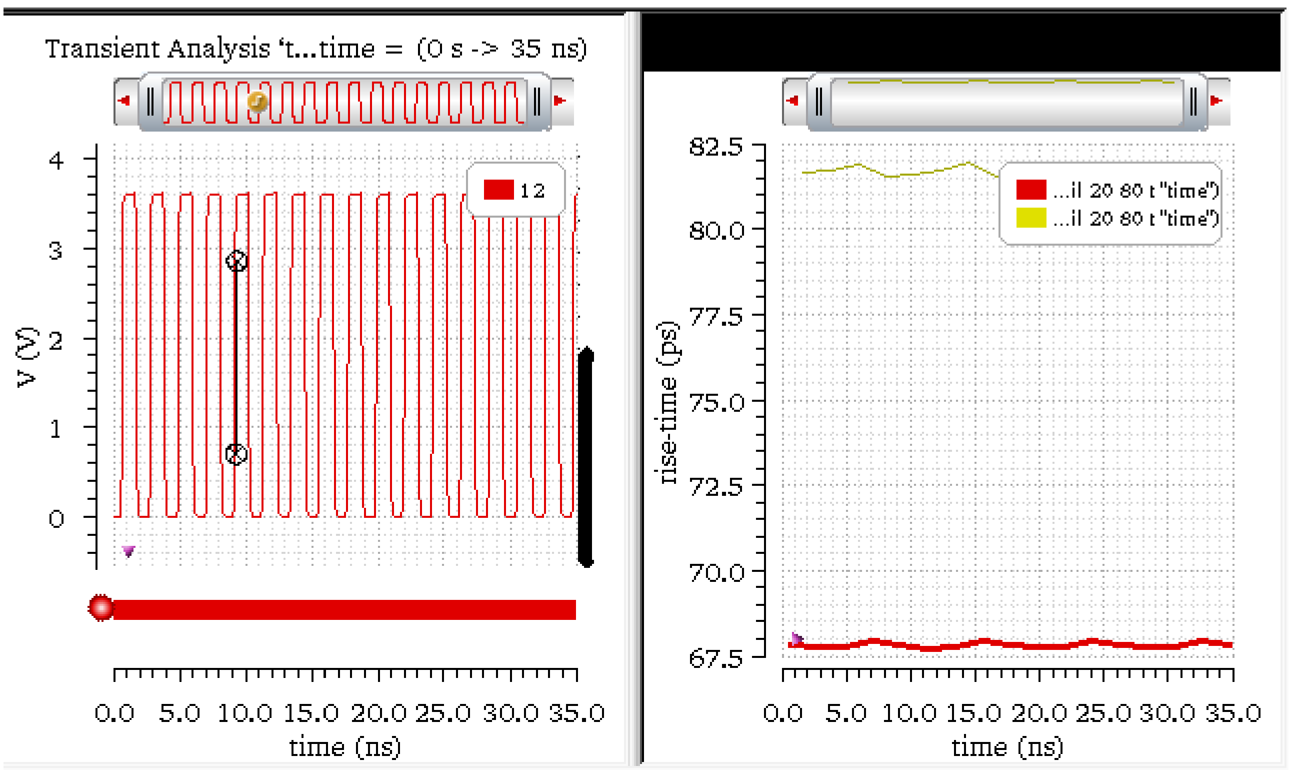
- ❑ *Show Edge Browser*—Select this check box to show or hide the edge browser for the selected trace, which is displayed at the bottom of the graph or strip. For more information about edge browser, see Using Edge Browser on page 159.

- ❑ *Settings*—Click this button to view and modify the edge settings of the selected trace. For more information, see Specifying Edge Settings on page 157.

- ■ *Measurements*—Calculates and displays the following measurement values for the selected edge:
  - ❑ *Rise Time*
  - ❑ *Fall Time*
  - ❑ *Overshoot*
  - ❑ *Undershoot*
  - ❑ *Slew*

- ■ *Derived Plots*—Generates the derived plots for the selected edges in the graph. Derived plots are the risetime or falltime waveforms derived from the entire set of edges and plotted against time.

- *Rising Edges*—Click this check box if you want to generate derived plots for all the rising edges in the graph.
- *Falling Edges*—Click this check box if you want to generate derived plots for all the falling edges in the graph.
- *Mode*—Specify the plotting mode that you want to use for the derived plots. The available plotting modes are—*Append*, *Replace*, *New Window*, and *New SubWindow.*
- *Plot*—Click this button to plot the specified derived plots. The figure below displays the derived plots for falling as well as rising edges in a new subwindow:



Using Markers

You can use the following markers to view and examine a specific edge:

Edge Marker

An Edge Marker annotates an edge graphically by using the current edge information. If you update the edges by using the *Edge Settings* form, the edge markers are also updated to reflect the new edge data. In general, an edge marker shows the top and bottom threshold points of the edge and the risetime or falltime values of the selected edge. When you zoom into the marker, the additional information is displayed to show the baseline and topline values and other details. You can change the threshold value that is displayed by an edge marker. Note that this does not change the edge and only changes the annotation points of the edge marker. If you lock the marker, the intercepts points are not updated when the edges are changed.

For more information about edge markers, see Working With Edge Markers on page 278.

Period Marker

The Period Marker measures the distance between the selected edge and its adjacent rising or falling edge. By default, this measures the threshold mid-point distance between the two edges. You can use the marker context menu options to change the start and end measurement points for each edge. For example, you can measure from the high threshold of the first edge to the low threshold of the end edge. When you create a period marker, if required, the edge markers are added and connected with a standard delta marker.

Dx/Dy Marker

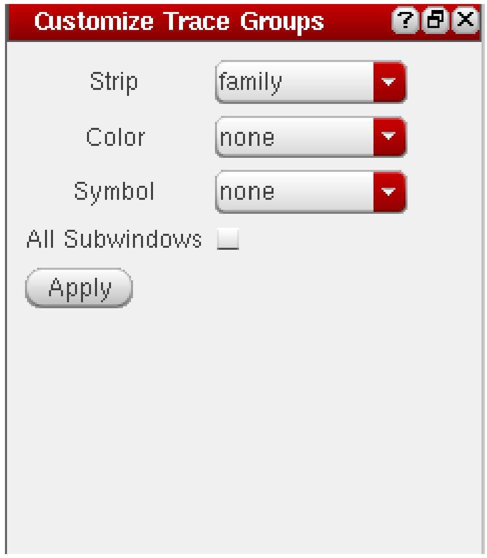
The Dx/Dy marker is a standard delta marker with end points initially placed at the low and high threshold intercept points of a single edge. After this marker is created, you can perform all the delta marker operations on it. Note that the dx/dy marker does not change with the change in the edges.

Customize Trace Groups

This assistant is used to customize the trace settings for the traces that belong to a common family.

Perform the following step to open the Customize Trace Group assistant:

➡ Choose *Window – Assistants – Customize Trace Groups.*

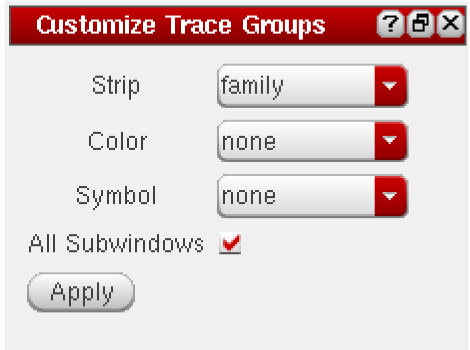


The traces in the graphs are updated based on the filter value you select in each field. The filter values can be: `none`, `trace`, `leaf`, `family`, `Corner`, and sweep variables.

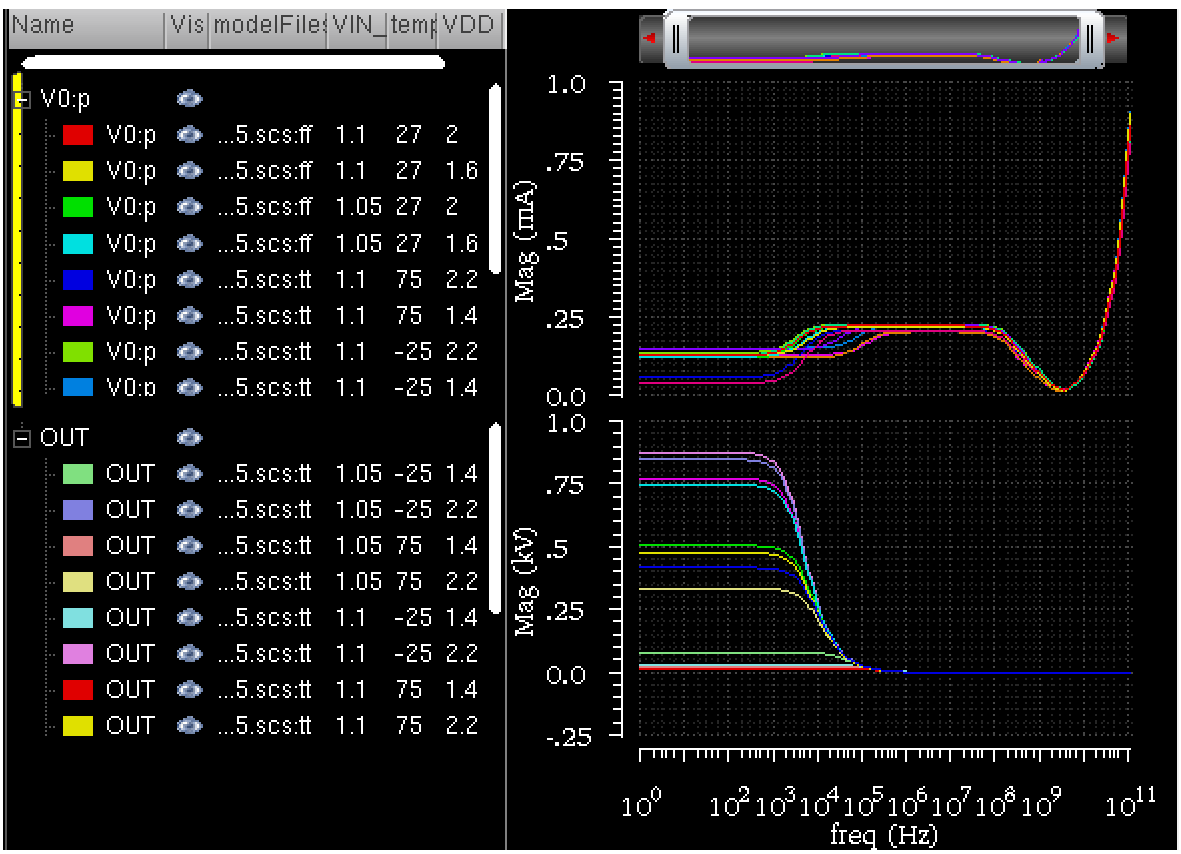
- *Strip*—Specify how you want to display the traces in a strip. Traces that belong to the filter value you select are displayed in the same strip. For example, if you select `family`, all the traces that belong to a common family are displayed in the same strip.
- *Color*—Specify the color of the traces. The traces that belong to the specified filter value you select are displayed in the same color. Traces can be displayed in 18 different colors and after that the colors are repeated.
- *Symbol*—Specify how symbols are to be shown on traces. The traces that belong to the specified filter value you select display the same type of symbols. By default, the symbol type is plus. To change the symbol type, right-click the trace and select the symbol type from the *Symbol* menu.
- *All Subwindows*—Select this check box to update the traces in all subwindows based on the options you select in this form.
- Click *Apply*.

Example1: Displaying traces belonging to a common family in one strip:

Set the fields in the Customize Trace Settings assistant as shown in the figure below:

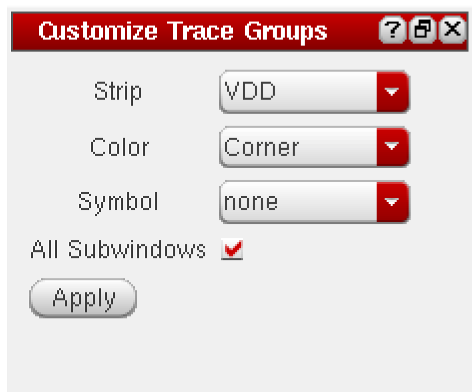


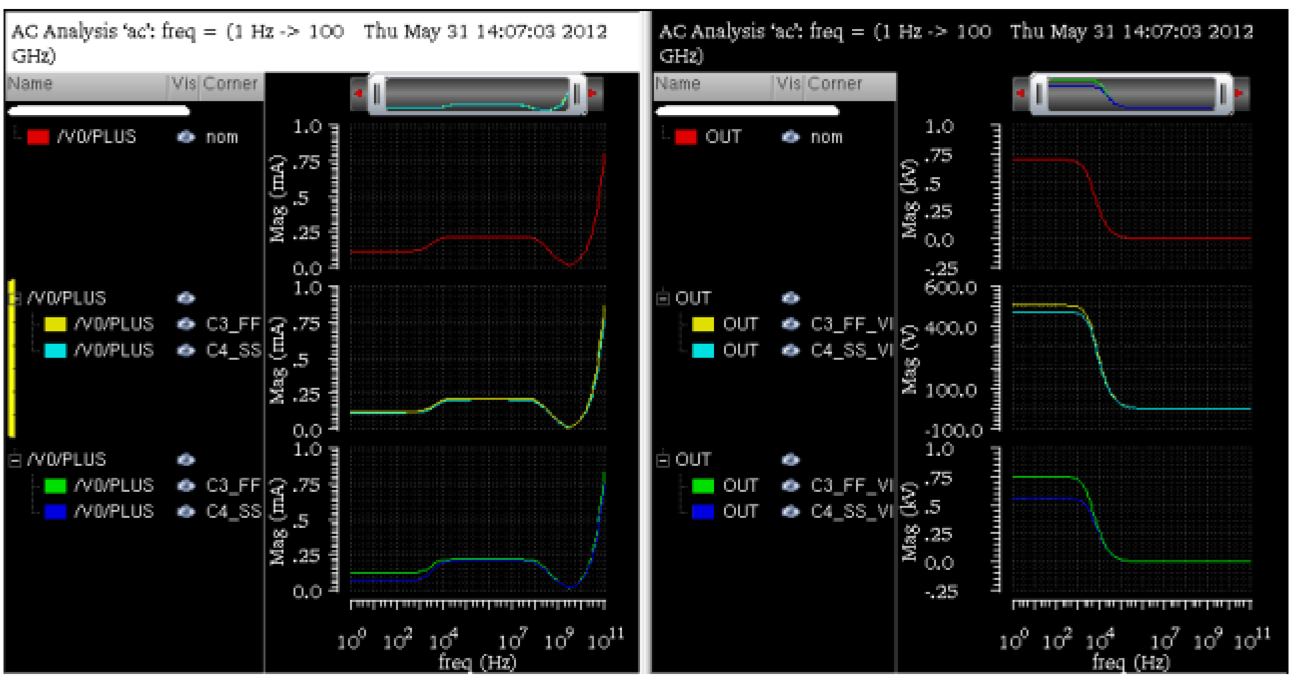
When you click the *Apply* button, the traces in the graph are updated as shown in the figure below. Notice that traces belonging to a common family (`V0:p` and `OUT`) are displayed in the same strip.



Example2*: Displaying traces that belong to a common VDD value in one strip and common corners in same color:*

Set the fields in the Customize Trace Settings assistant as shown in the figure below:

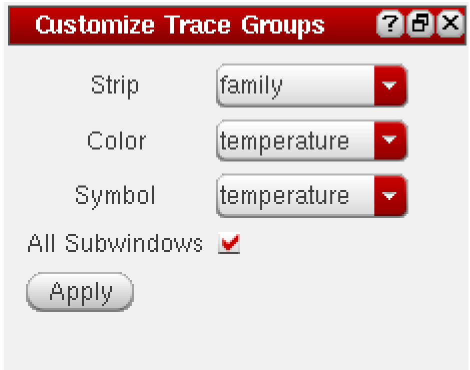


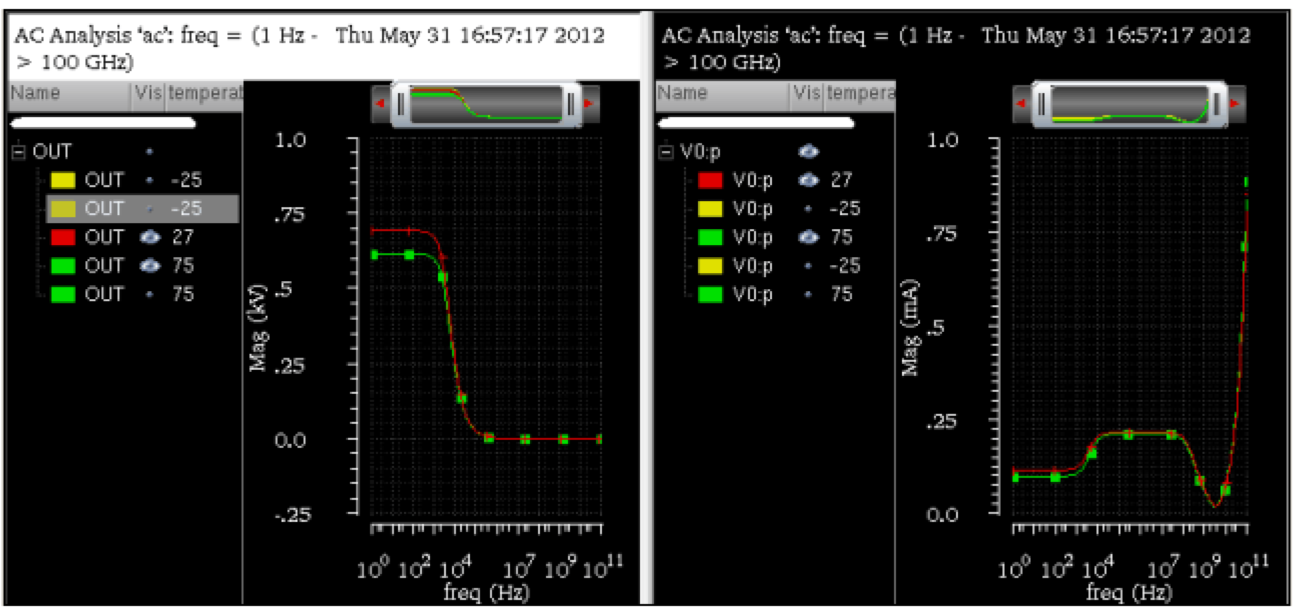


In this example, notice that the traces with similar VDD value are displayed in the same strip and traces that belong to similar corners are displayed in same color.

Example3: Displaying traces that belong to a common temperature value with same type of symbols displayed on them

Set the fields in the Customize Trace Settings assistant as shown in the figure below:





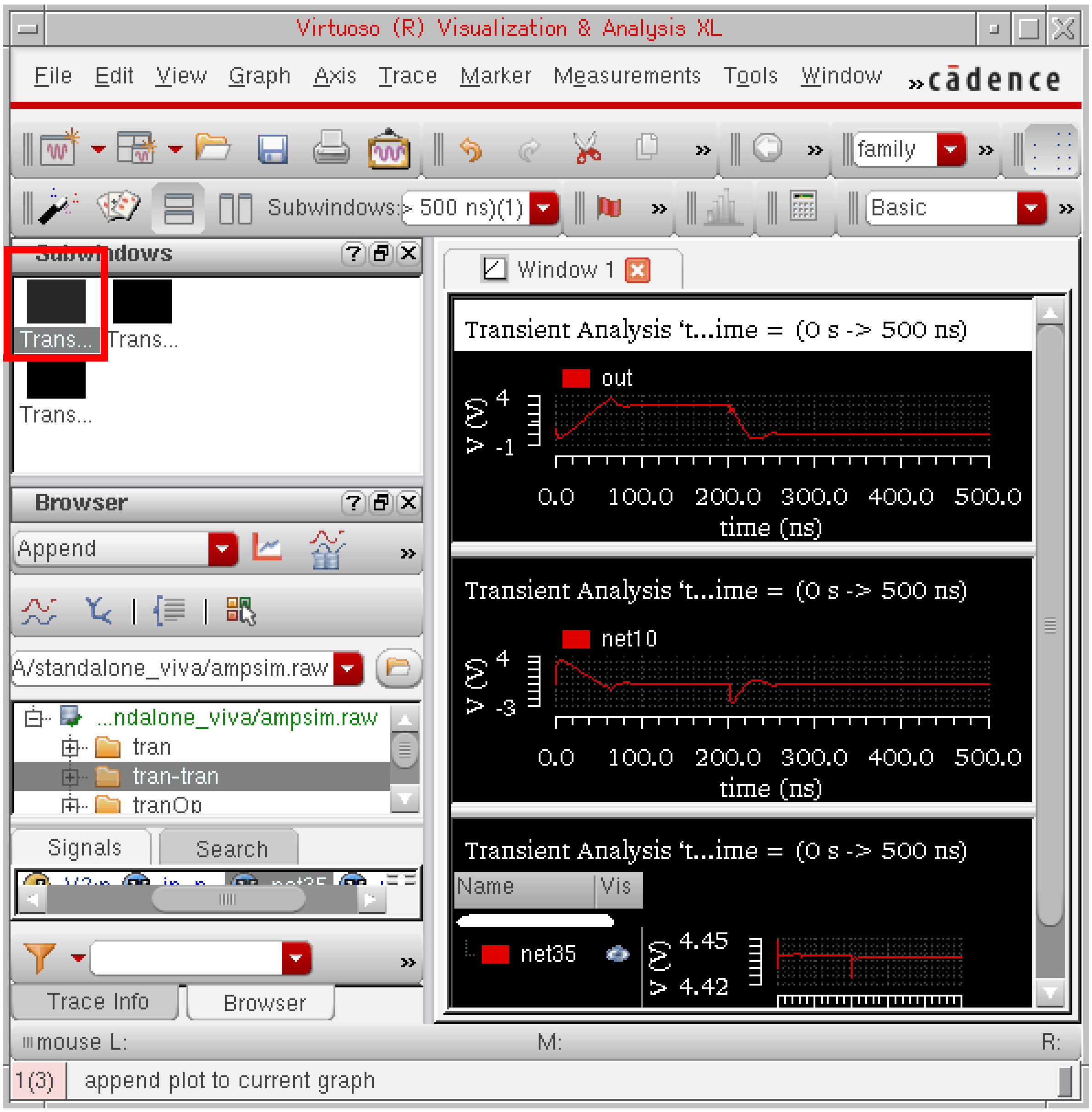
In this example, notice that the traces with similar temperature value are displayed in the same color and also display the similar types of symbols on them.

Subwindows

A subwindow is a graph that you can open within a window. The Subwindows assistant displays icons for all the subwindows that are open in a window. When you open a new window, the Subwindows assistant creates a default subwindow named `Subwin(1)`.

The title of a subwindow includes the analysis name. If a graph includes an expression, the expression name is displayed in the subwindow title. The subwindow name is displayed in the Subwindow toolbar.

You can specify the layouts to display subwindows in the window. For more information about graph layouts, see Specifying the Graph Layout on page 98.



To determine the active subwindow in a graph, see Determining the Active Subwindow on page 96.

Deleting a Subwindow

To delete a subwindow, do one of the following:

- Right-click the graph in the subwindow and choose *Delete*.
- Select the subwindow you want to delete and choose *Edit – Delete*.
- Select the subwindow you want to delete and press the `Delete` key.
- Select the subwindow icon in the Subwindows menu and press the `Delete` key.

  The selected subwindow is deleted.

Copying a Subwindow

To copy a subwindow to a new window or a subwindow:

➡ Right-click a subwindow icon (highlighted in the above figure) in the Subwindows assistant and choose one of the given options in the *Copy To – New Window* or *New Subwindow* menu. The options are *Rectangular*, *Polar*, *Impedance*, or *Admittance*.

  The selected subwindow is copied to a new window or subwindow.

Hiding a Subwindow

➡ Right-click a subwindow icon (highlighted in the above figure) in the Subwindows assistant and deselect the *Visible* option. This option is selected by default.

  The selected subwindow icon turns blank identifying that the subwindow is hidden.

Working with Workspaces

A workspace is the arrangement of various assistants and the window settings that you specify while working with a graph. You can either use the available workspaces or create your own workspace while working in the window.

Workspace Types

The available workspaces are of four types:

- **Basic**—This workspace displays the following dockable assistants:
  - ❑ Subwindows

- Results Browser
- Graph

■ **Browser**—This workspace displays the following dockable assistants:

- Results Browser
- Graph

■ **Classic**—This workspace displays only the graph window.

■ **MarkerTable**—This workspace displays the following dockable assistants:

- Subwindows
- Results Browser
- Graph
- Marker Table

■ **TM**—This workspace displays the following dockable assistants:

- Subwindows
- Results Browser
- Graph
- Transient Measurement Assiatant

If you open the tool in stand-alone mode, it is opened in the `Basic` workspace because in stand-alone mode you work on the saved simulation results.

If you open the tool from within ADE, it is opened in the `Classic` workspace because in ADE you work on the simulation results for the current run.

When you open a new Virtuoso Visualization and Analysis XL session from the same Virtuoso window, the workspace that you specified in the previous session is available.

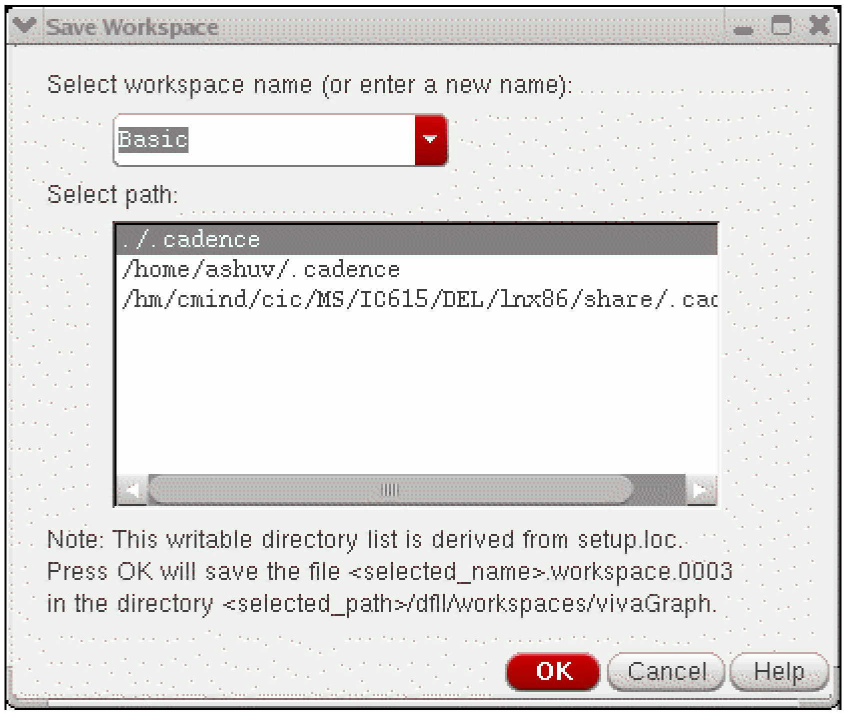
Saving a Workspace

You can customize a workspace by selecting the assistants that you want to display from the *Window – Assistants* menu. You can then save the customized workspace by doing one of the following:

■ Choose *Window – Workspaces – Save As.*

- On the Workspace toolbar, select the  option.

  The *Save Workspace* form appears.



  In this form, specify the name with which you want to save the workspace and select the path where you want to save the workspace. You can specify a new name or can make changes to an existing workspace.

If you do not want to save tWorking With Assistants on page 133he changes you made to the existing workspace, choose *Windows – Workspaces – Revert to Saved* to revert to the factory settings.

Loading a Workspace

To load a workspace, do one of the following:

- Choose *Windows – Workspaces – Load.*

  The *Load Workspace* form appears. In this form, select the workspace you want to load.

- On the Workspace toolbar, select the required workspace from the *Workspace* drop-down list box.

Deleting a Workspace

To delete a workspace, choose *Windows – Workspaces – Delete.*

The *Delete Workspace* form appears. Select the name of the workspace that you want to delete.

Setting the Default Workspace

To set a workspace as the default workspace, choose *Windows – Workspaces – Set Default.*

The *Default Workspace* form appears. Select the name of the workspace that you want to set as the default.

Showing and Hiding Assistants

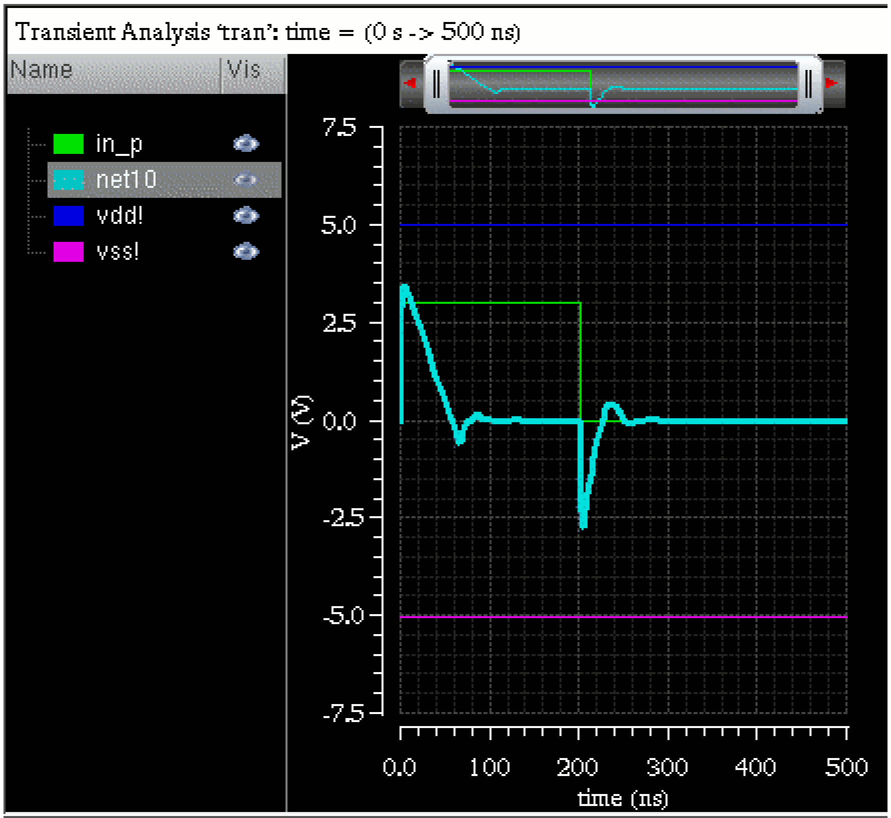
To show or hide the assistants in the workspace, do one of the following:

- Choose *Windows – Workspaces – Show/Hide Assistants*
- Press the `F11` key to hide or show the assistants.
- On the Workspace toolbar, select the  option.

Working with Traces

A signal when plotted in the window is called a trace. Each trace in the window is displayed in a different color. The graph supports 18 unique colors. The information displayed in the area to the left of the trace is called the trace legend. A splitter line separates the trace and the trace legend area.

To highlight a trace in the window, select the legend corresponding to the trace or select the trace in the graph.



The trace legend area displays the following information:

- The name and color of all the traces plotted in a graph.
- The corner names and sweep parameters from ADE L/XL in separate columns that can be sorted.
- The trace families, based on sweep parameters, in a hierarchical order. To view the analog traces in a family or the digital traces in a bus, click the + symbol in the trace legend area.

Each trace in the trace legend area displays a Visibility button  that you can use to show or hide traces.

When you plot traces belonging to a sweep data, do the following to show or hide sweep variables in the trace legend:

➡ Right-click the trace legend header and choose the sweep variable you want to hide or show.

A red tick mark appears with the selected sweep variable, which indicates that the variable is displayed in the trace legend.

You can resize the trace legend area by dragging the dynamic splitter on the right. You can also adjust the width of each column in the trace legend area.

By default, the trace legend is displayed on the left of the graph. You can also move the trace legend inside the graph or to the top of each strip by using the *Graph Properties* form. For more information, see Editing Graph Properties on page 108.

You can select one or more traces in a graph by doing the following:

- To select a trace, click the trace in the graph or click the trace name in the trace legend area.
- To select all traces in a graph or a subwindow, press `Ctrl+Shift+A`.
- To select all traces in a strip, press `Ctrl+A`.

Note: The active strip is determined by a yellow bar displayed to the left of the trace legend area.

You can select a trace and choose one of the following *Trace* menu commands to manipulate traces in a window:

- *Symbols On*—Displays the symbols on the individual data points for the selected trace.
- *Select By Family*—Selects the traces from the parametric sweeps by family, instead of selecting an individual leaf.
- *Strip By Family*—Adds all the traces belonging to a family to a strip.
- *Select All*—Selects all the traces in a graph or in a strip in the active window or subwindow.
- *Delete All*—Deletes all the traces in the active window or subwindow. You can also use bind key `Ctrl+E` to delete all the traces in a window.
- *Move to*—Moves the selected trace to the following locations:
  - *New Window*—Moves the selected trace in a graph to a new window.
  - *New Subwindow*—Moves the selected trace in a graph to a new subwindow.
  - *New Strip*—Moves the selected trace in a graph to a new strip.
- *Copy to*—Copies the selected trace to the following locations:
  - *New Window*—Copies the selected trace to a new window.

- *New Subwindow*—Copies the selected trace in a graph to a new subwindow.
- *New Strip*—Copies the selected trace in a graph to a new graph strip. This is especially useful when you want to study a single trace separate from a set of parametric leaf waveforms. You can alter trace selection to select signals by family, rather than by selecting individual traces, by choosing *Trace - Select by Family*.

To show or hide traces, right-click the trace and choose *Visible*. The red check mark is displayed with this option indicating that the trace is visible.

Note: When you move a trace to a new window, the moved trace is plotted with the same x-axis scale in the new window.

Dragging Traces

Traces support the following drag-and-drop operations:

- You can drag traces from one window to another.
- You can drag traces from one subwindow to another. The window to which you drag a trace becomes the active window.
- You can drag traces from one strip to another.

Tracking Cursor

The tracking cursor displays the color and name of the trace and its X- and Y-axis values when you drag a trace. Do one of the following to display tracking information on the tracking cursor:

- Choose *Markers – Tracking Cursor*.
- Click the Tracking Cursor button on the Marker toolbar.

Hiding and Showing Traces

To show or hide a trace, do the following:

- Right-click a trace and select *Visible*.
- In the trace legend area, click the Visibility button ( ).

Deleting Traces

To delete a trace from a graph, subwindow, or strip, do one of the following:

- Select the trace and choose *Edit – Delete*, or press the `Delete` key.
- Right-click the trace and choose *Delete*.

To delete all traces in a subwindow, select a trace and do one the following:

- Choose *Trace – Delete All*
- Press `Shift+E`.

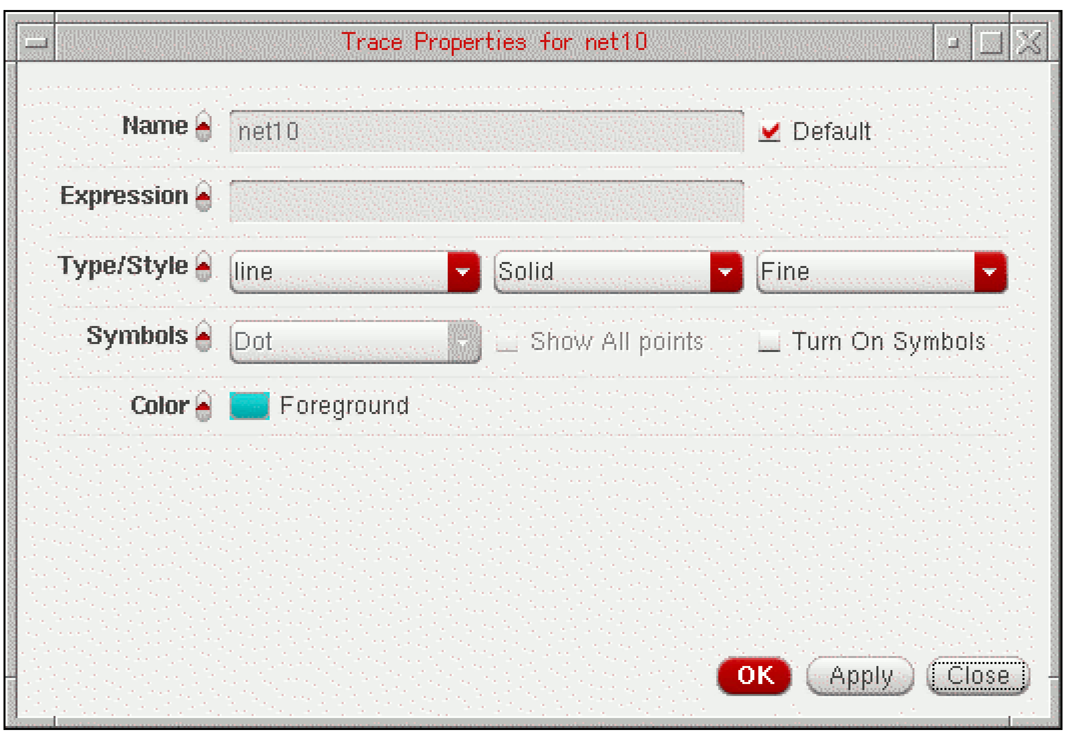
Setting Trace Properties

The default trace properties are controlled by the values assigned to variables in the `.cdsenv` file. For more information, see <u>Appendix A, "Virtuoso Visualization and Analysis XL Tool Environment Variables."</u>

Do one of the following to set the trace properties for a trace:

- Double-click a trace in the window.
- Select a trace and choose *Trace – Properties*.

  The *Trace Properties for <trace-name>* form appears.



In this form, set the trace properties:

- ❑ In the *Name* field, type the name for the trace or select the *Default* check box to display the default trace name. When you select the *Default* check box, the *Name* field becomes unavailable. The *Name* and *Default* fields are not available if you select more than one trace.
- ❑ In the *Expression* field*,* type the expression associated with the selected trace or get the expression from the Calculator Buffer.
- ❑ In the *Type/Style* fields, do the following:
  - ❍ Specify whether you want to represent the trace by a *line*, *points*, *histogram*, *bar*, *spectral,* or *sampleHold*.
  - ❍ Specify whether you want the trace style to be *Solid*, *Dashed*, *Dotted*, or *DotDashed,* or *DashDotDot*.

  Note: The trace style option does not work for *Bars* and *Spectrum*.
  - ❍ Specify whether you want the trace to be *Fine*, *Medium*, *Thick* or *ExtraThick*.

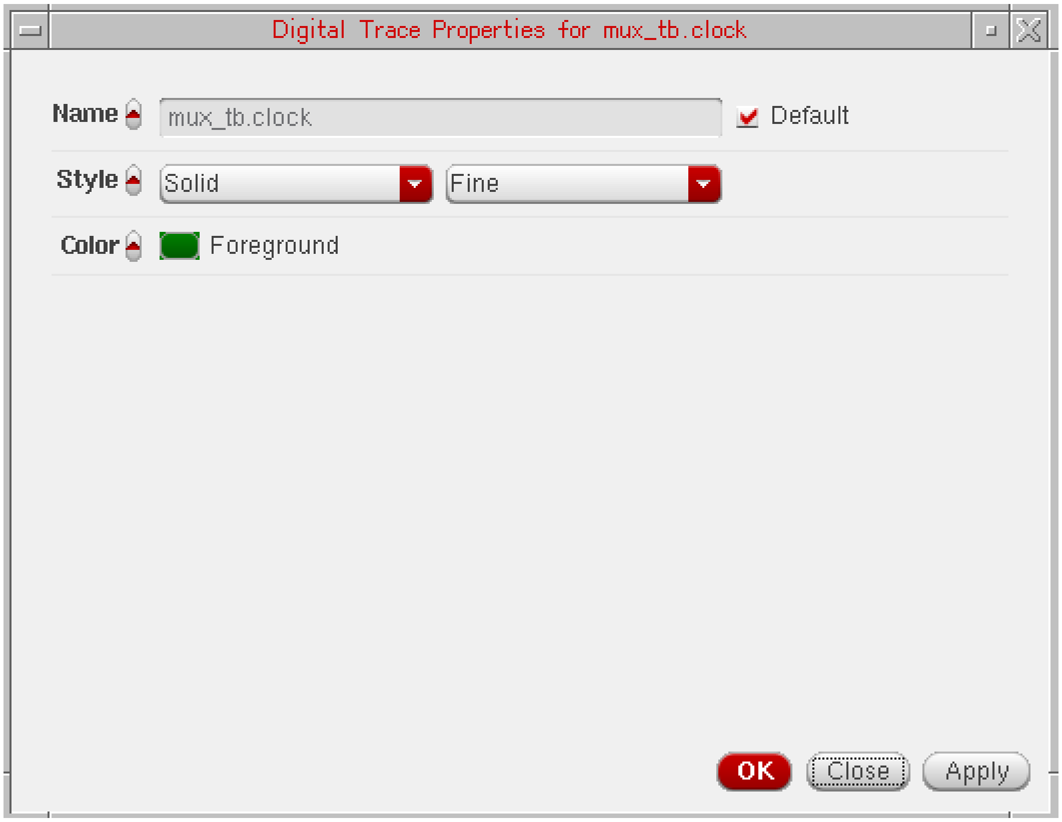
  Note: The trace thickness option does not work for *Bars* and *Spectrum*.
- ❑ In the *Symbols* field, select the *Show All Points* check box to display data points on the trace and specify the number of points to be displayed.
- ❑ In the *Points per Symbol* field, specify whether you want data points to be displayed on the trace as symbols of *Point*, *Dot*, *Plus*, *Square*, *Box*, *X*, *Circle,* and so on*.*
- ❑ In the *Color* field, select the foreground color for the trace. Alternatively, you can also set the trace color by right-clicking the trace and selecting *Color.*
- ❑ Click *OK*.

Note: You can also select these properties by right-clicking a trace.

For more information about the selected trace, choose *Window – Assistants – Trace Info Panel*. The Trace Info Panel assistant appears, displaying the *Name, Max value, Min value,* and *Data points* of the selected trace.

Setting Digital Trace Properties

To change the properties of a digital signal, right-click the digital signal and choose *Digital Trace Properties*.



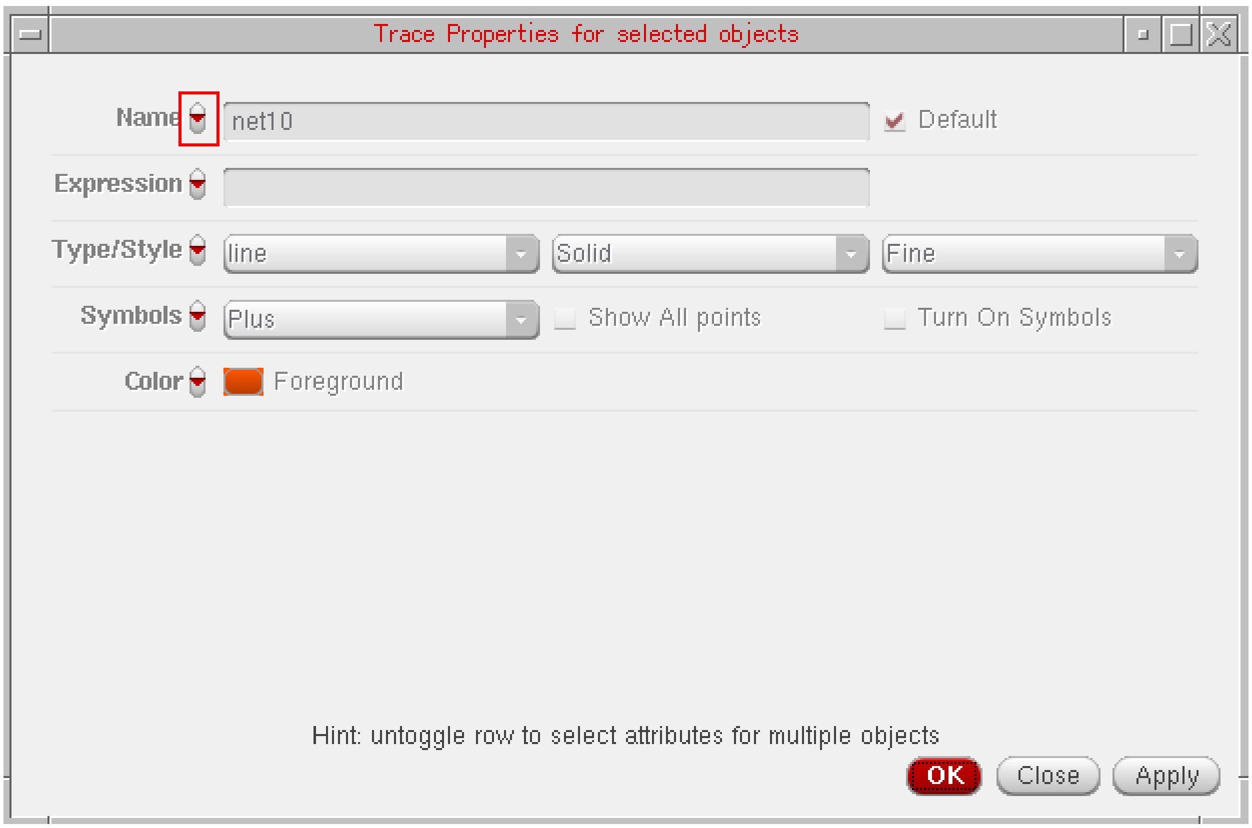
The *Digital Trace Properties for <signal name>* form appears. This form includes the following fields:

- *Name*—Specify the name for the trace or select the *Default* check box to display the default trace name. When you select the *Default* check box, the *Name* field becomes unavailable. The *Name* and *Default* fields are not available if you select more than one trace.
- *Style*—Specify whether you want the trace style to be *Solid*, *Dashed*, *Dotted*, or *DotDashed,* or *DashDotDot*. And, specify whether you want the trace width to be *Fine*, *Medium*, *Thick* or *ExtraThick*. Alternatively, you can set the trace style and width by right-clicking the trace and selecting *Style* and *Width* respectively.
- *Color*—Select the foreground color for the trace. Alternatively, you can set the trace color by right-clicking the trace and selecting *Color.* By default, the trace is displayed in green.
- Click *OK*.

Setting Properties for Multiple Traces

You can set the properties for more than one trace at a time. To perform this, select the traces by using the `Ctrl` key and choose *Trace – Properties*.

The *Trace Properties for selected objects* form appears.



You can change the form fields by clicking the arrow button adjacent to each field name (as highlighted in the figure above).

Note: You can also use this method to change the properties of markers of similar marker type. For example, select markers of similar marker type by using the `Ctrl` key, such as two or more vertical markers, and choose *Marker – Properties*. The *<Marker-name> Properties for selected objects* form appears. You can then use this form to change the properties for selected markers in a single step.

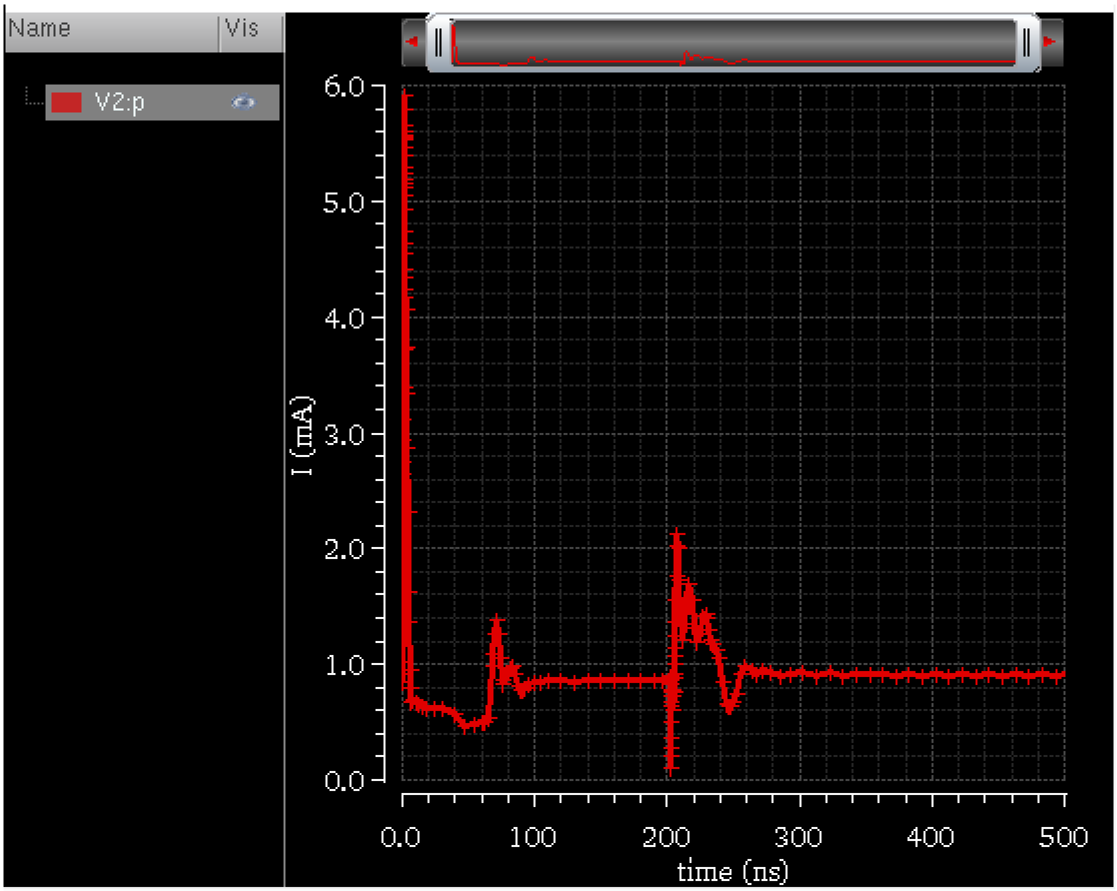
Displaying Symbols on a Trace

To display symbols for the data points on a trace, do one of the following:

➤ Select a trace and choose *Trace – Symbols On*.

You can control the symbol type and the number of data points that can be identified by the symbols. The symbol used for the trace is displayed next to the trace name.

The following figure illustrates how two traces can be distinguished by using symbols for the data points.



Exporting a Trace

To export a trace from the window in a variety of formats and later load it in the required application. You can also save a clipped part of the dataset by specifying the start and end values, or interpolate the data before saving it. By exporting a trace, you can also save the expressions associated with the trace.

To export the trace, do one of the following:

- Select the trace and choose *Trace — Export*.
- Right-click the trace and choose *Send To – Export*.

The *Export Waveforms* form appears. For detailed information about the fields in this form and how to save a trace using this form, see Exporting Signals on page 62.

Sending Trace Expressions to Calculator

To send the expression associated with a trace in the window to the Calculator, select the trace for which you want to send the expression to the Calculator and do one of the following:

- Choose *Tools – Calculator*.
- Right-click the trace and choose *Send To – Calculator*.

  The Calculator window appears with the expression for the selected signal displayed in the Buffer.

  You can select more than one trace by holding down the `Shift` or `Ctrl` key and clicking the traces you want to select. The most recently selected trace appears in the Buffer and the remaining traces are added to the Stack with the recently selected trace at the top of the stack.

Sending Traces to ADE

To include the expressions for the traces displayed in a graph directly to ADE L and XL output Setup tab, right-click a trace and choose *Send To – ADE*. The expression for the selected trace is added to ADE as a new output and evaluated when you run the simulation.

You use this option if you want to evaluate the expression for the trace in the current simulation run in ADE.

When you send to ADE the measurement values obtained from assistants, such as Eye Diagram and Spectrum toolbox, the alias name in the ADE displays the assistant name and the measurement name. For example, if you send the DC Power value from Spectrum toolbox to ADE, it is displayed with the `spectrum_dcpower` alias name. Similarly, the Level0 Mean value calculated from Eye Diagram is displayed with the `eye_level0Mean` alias name.

Note: If you are using the Refresh feature in ADE L or XL, the traces that you include into ADE L or XL from the graph are also updated with the new simulation data.

Working with Strips

You can append multiple traces to a graph. If you want to view the individual traces, you can split the graph into strips that are arranged vertically. Each strip has its own Y-axis and shares the X-axis with the other strips. The window displays the trace legend separately for each individual strip.

The active strip is determined by a yellow bar displayed on the left of the strip. If you want to change the order in which the strips are displayed, drag the strips. You can drag a strip to any of the following locations:

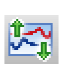
- To a strip—the trace is appended to the strip.
- To an area outside the strip—the trace is placed in a new strip below the strip closest to the point to where you drag the strip.

You can also resize the active analog strip by dragging the strip splitter.

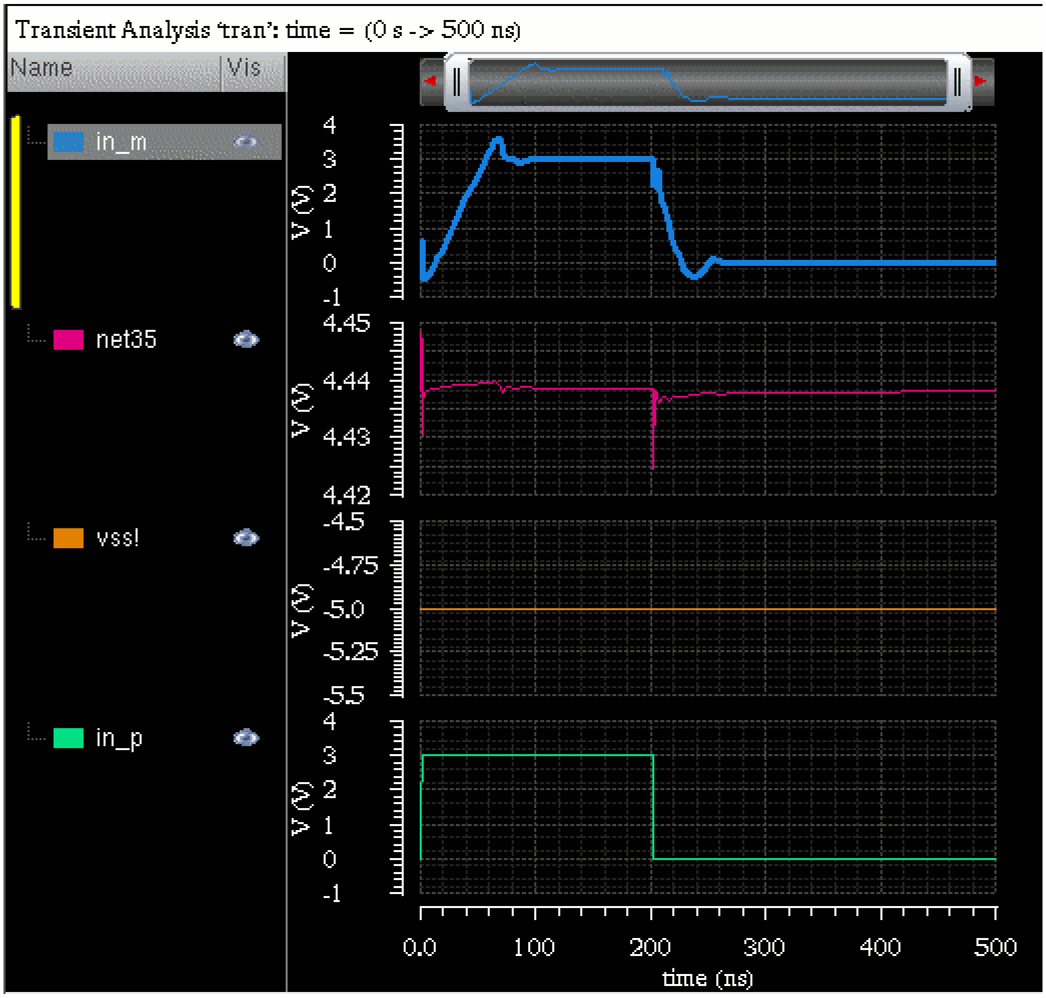
This section includes the following topics:

Splitting Current Strip

To split the traces in an active graph into individual strips, do one of the following:

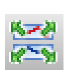
- Choose *Graph – Split Current Strip*.
- Right-click anywhere in the window and choose *Split Current Strip – trace/leaf/family.*
    - If you choose *trace*, all the traces in the active strip are displayed in individual strips.
    - If you choose *leaf*, all the leaf traces are displayed in individual strips.
    - If you choose *family*, the traces that belong to a common family are displayed in one strip.
- Click the  button on the Strip toolbar.

Each trace in the active strip is displayed in a separate strip.



Splitting All Strips

To split the traces from all the strips into individual strips, do one of the following:

- Choose *Graph – Split All Strips*.
- Click the  button on the Strip toolbar.

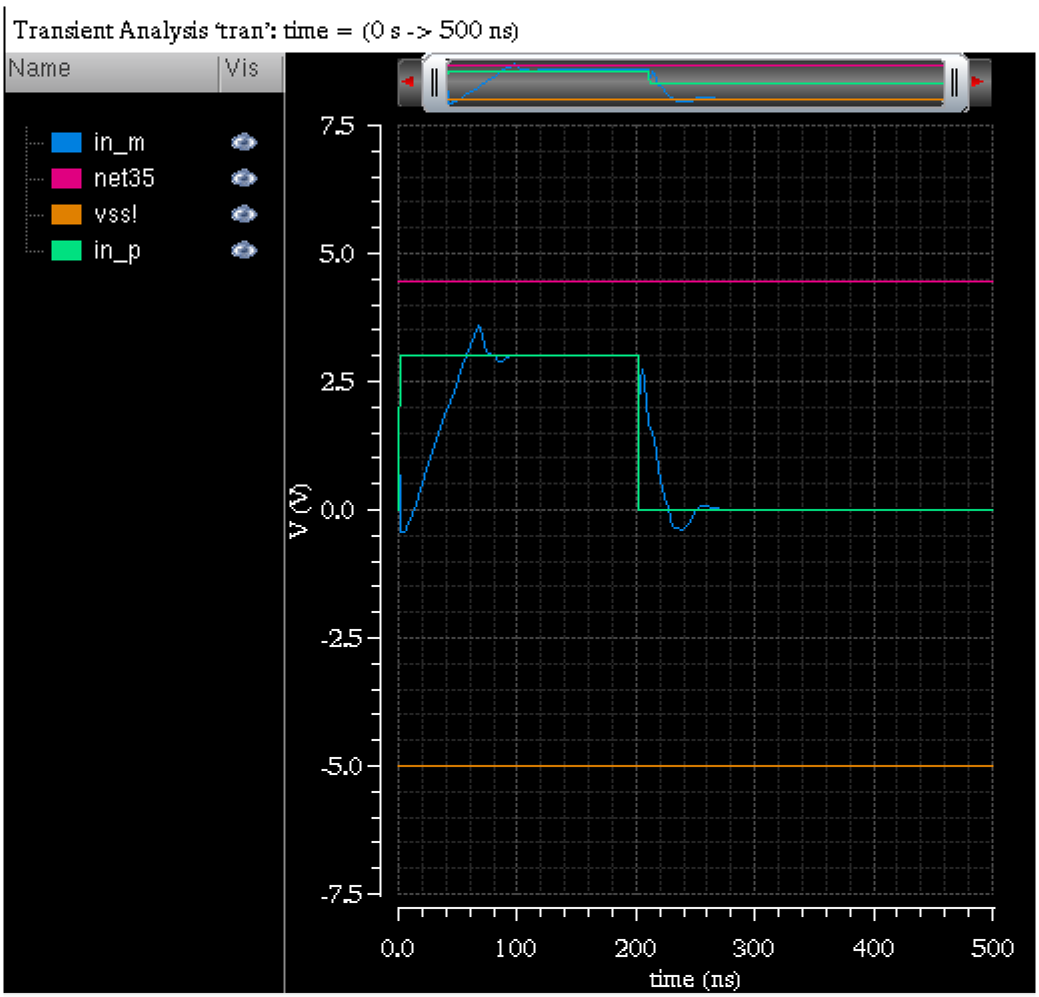
  Traces contained in each strip are displayed in individual strips.

Combining Graph Strips

You can combine one or more strips by dragging them to a single strip. To combine multiple graph strips, do one of the following:

- Choose *Graph – Combine All Analog Traces*.
- Right-click anywhere in the window and choose *Combine All Analog Traces*.
- Click the  button on the Strip toolbar.

  The traces displayed in the various strips are combined into a single graph.



Moving Traces

Do one of the following to move the selected traces in a graph to a new window, subwindow, or strip:

- Select a trace and choose *Move to – Move Selected Traces to a New Window/Move Selected Traces to a New Subwindow/Move Selected Traces to a New Strip*.
- Select the traces that you want to move. Right-click the selection and choose *Move to – New Window/New Subwindow/New Strip*.
- Select the traces and click the  button on the Strip toolbar.

Copying Traces

Do one of the following to copy the selected traces in a graph to a new window, subwindow, or strip:

- Select a trace and choose *Copy to – Move Selected Traces to a New Window/Move Selected Traces to a New Subwindow/Move Selected Traces to a New Strip*.
- Select the traces that you want to move. Right-click the selection and choose *Copy to – New Window/New Subwindow/New Strip*
- Select the traces and click the  button on the Strip toolbar.

Setting Strip Properties

To set the strip properties, do one of the following:

- Choose *Graph – Properties.*
- Right-click anywhere in the window and choose *Graph Properties*.

  The *Graph Properties* form appears. On the *Strips* tab, set the analog and digital height for the strips.

  In this form, you can set the minimum analog height, and the minimum and maximum digital heights of strips. If you click the *Default* button, you cannot change the values of these fields, and the values that you have entered become the default values.
- Click *Apply,* and then click *OK*.

Locking and Unlocking Strips

You can lock and unlock a strip while splitting the traces into different strips.

To lock a strip, right-click anywhere in the window and choose *Lock Strip Size*. A red check mark is displayed before this command and a lock icon appears at the top right corner of the strip, displaying the strip in the locked mode. If you change the size of any other strip in the window, it does not change the size of the locked strip.

To unlock the strip, right-click anywhere in the window and choose *Lock Strip Size*. The lock icon is no longer displayed and you can resize the strip now.

Video

The Using Strip Charts in Qt Graph video demonstrates how to split and combine analog traces into strips, resize strips, plot the trace to a new strip, and split and combine traces from sweep data.

Working with Sweeps

To display the sweep data for a family in the same strip, choose *Trace – Strip by Family* before plotting the sweep data. Now, if you select to display traces in strips, each trace family is displayed in a separate strip.The trace legend area displays the traces in the family. Click the + sign to view all traces.

- To select an individual trace in the family, click the trace in the trace legend area.
- To select all the traces in the family, choose *Trace – Select By Family*. Now, when you select a signal in the trace legend area, the entire family is selected. Alternatively, you can select all signals in the family by using the `Ctrl` key.

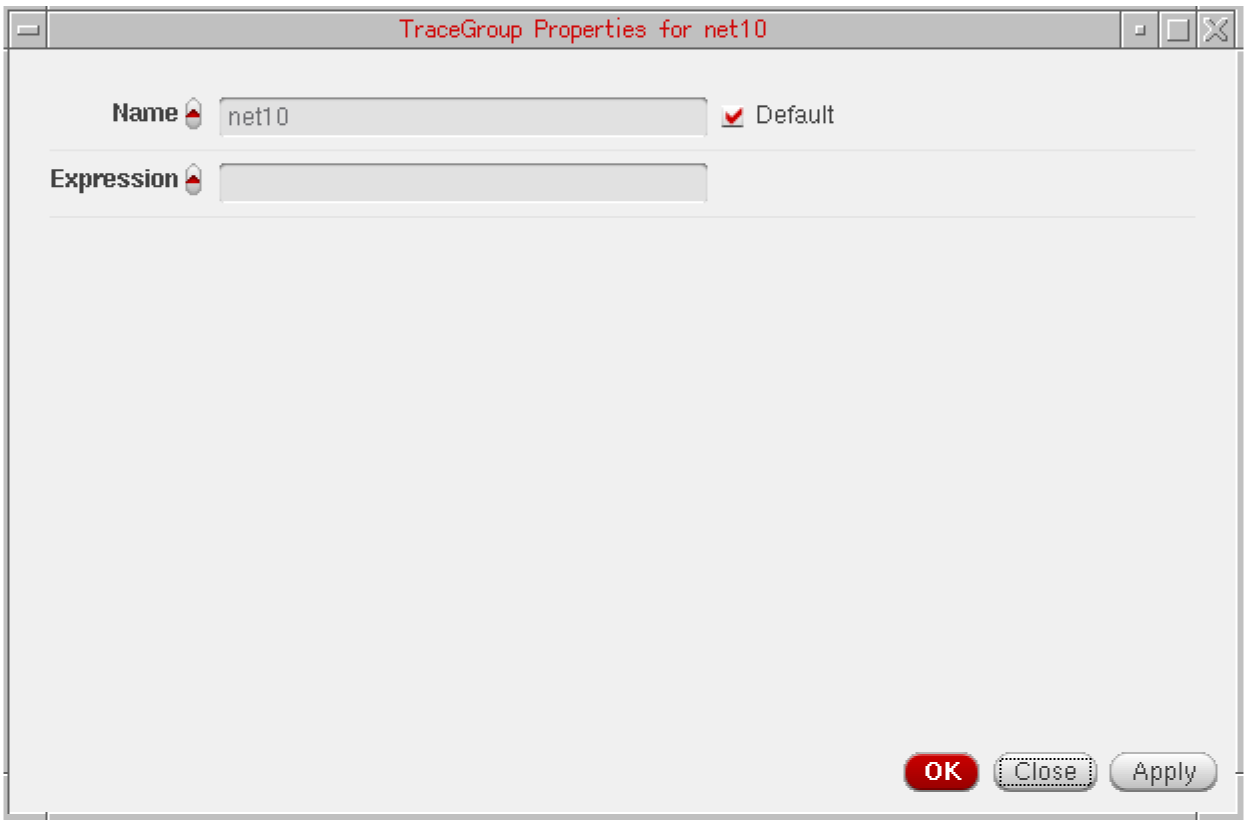
To display the traces from the sweep data in individual strips, ensure that the *Trace – Strip By Family* is not selected. Then, right-click anywhere in the graph and choose *Split Current Strip*.

Changing Trace Properties for Family

To change the properties of a trace in the family, right-click the trace in the trace legend and choose *Trace Properties*. For more information about how to use the Trace Properties form, see Setting Trace Properties on page 180.

To change the properties of all traces in the family, select the traces by using the `Ctrl` key and choose *Trace Properties*. The *Trace Properties for selected objects* form appears. For more information about this form, see Setting Properties for Multiple Traces on page 183.

To change the trace group properties for a family of traces, right-click the family header in the trace legend and choose *Trace Group Properties*. The *TraceGroup properties for family-name* form appears.



Swapping Sweep Variables

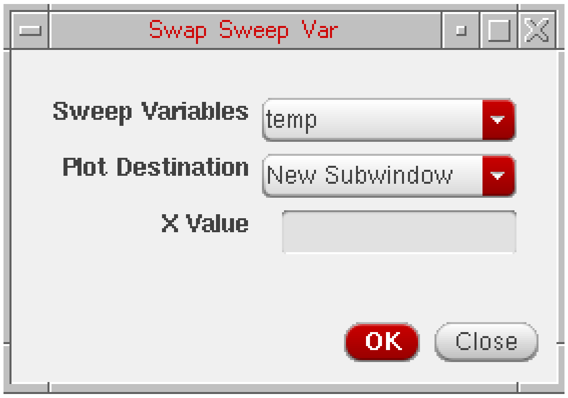
The sweep data can include multiple sweep variables; however, you can plot sweep data analysis results by using only two variables at a time. If you want to plot the sweep data results with another variable, you can swap the X-axis variable with this variable.

To swap the sweep variables in the graph:

➡ Right-click the X-axis and choose *Swap Sweep Var.*

  The *Swap Sweep Var* form appears. You cannot perform any action in the Virtuoso Visualization and Analysis XL window when the Swap Sweep Var form is open.

Note: The *Sweep Swap Var* option is enabled only if the simulation involves more than one sweep variable.



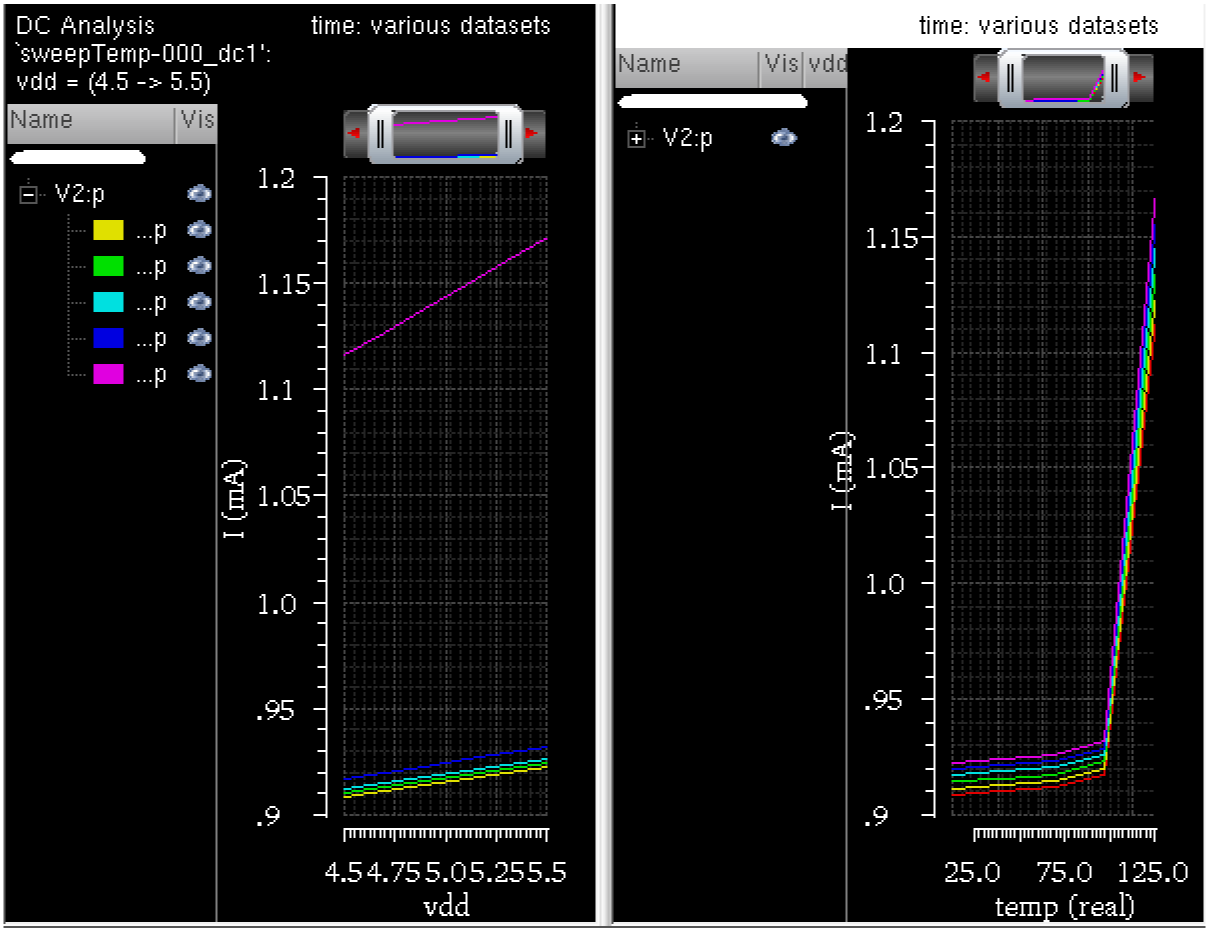
The *Sweep Variables* drop-down list box in this form displays all the sweep variables. Now, perform the following steps:

- ❑ Select from the drop-down list the sweep variable that you want to swap and specify the plot destination that can be a new window or a subwindow.

  Note that the *X Value* field is not available in this form. This field is used to plot data for signals for a particular X-axis value of frequency (AC response) and time (transient response). When the graph includes the plots against frequency or time, this field is enabled.

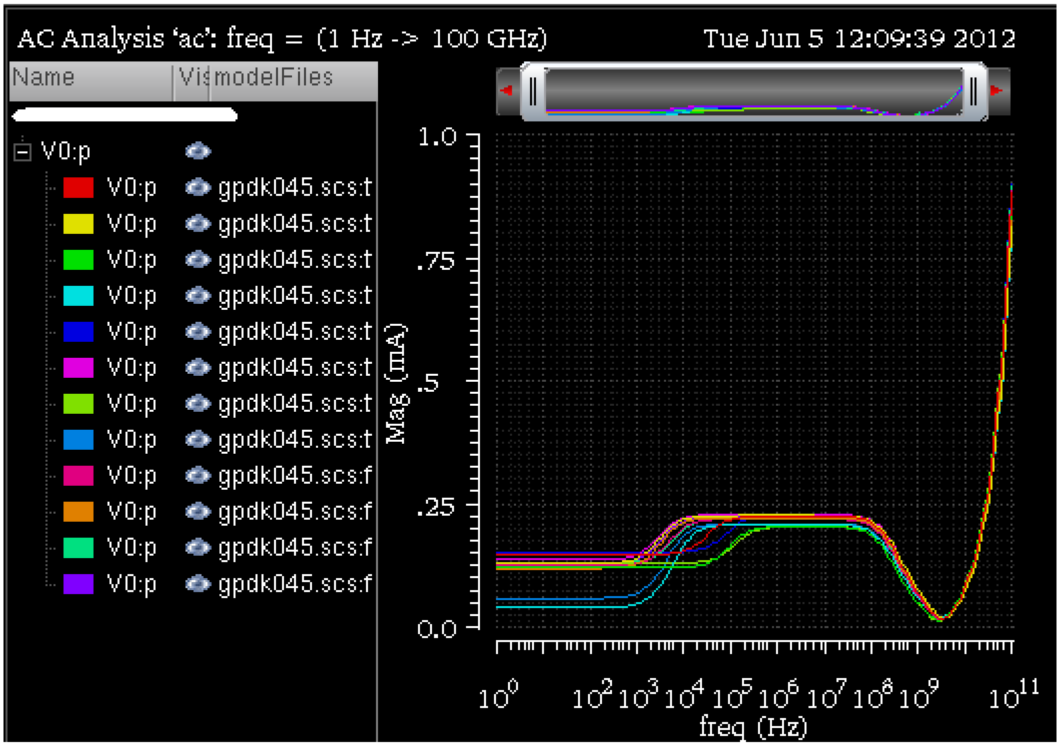
- ❑ Click *OK*.

A new graph is created according to the swapped variable for the X-axis.

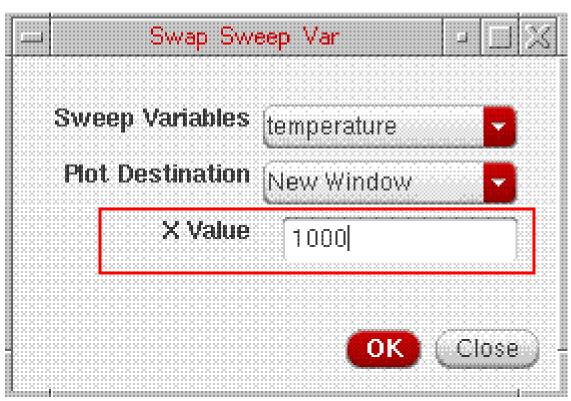


Plotting Graph across Fixed Frequency

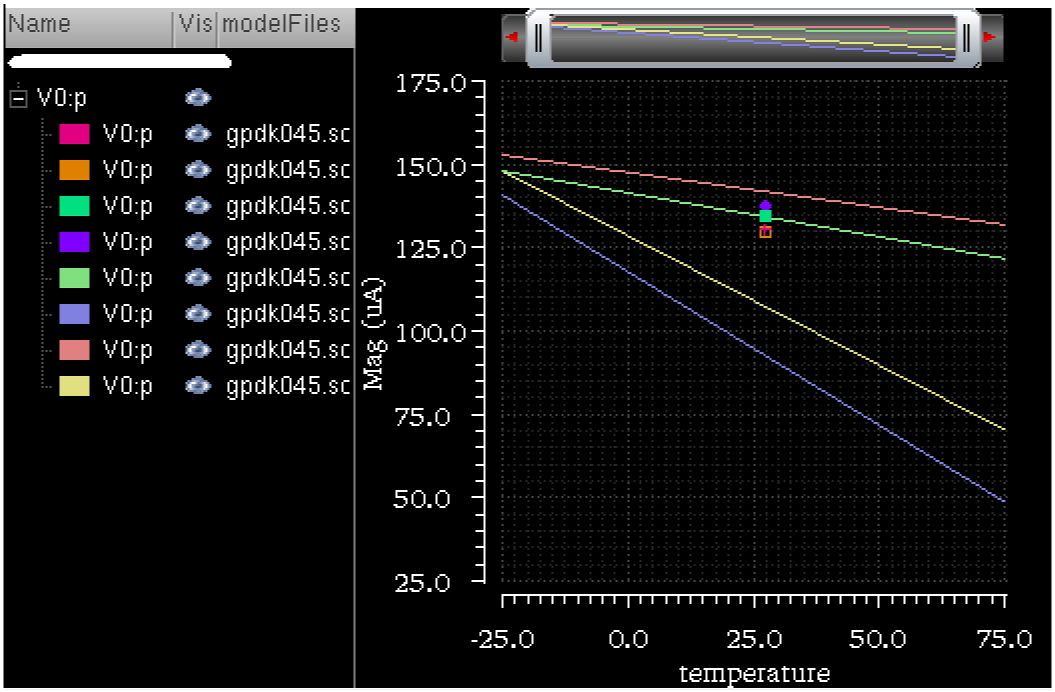
The graph shown in the figure below is obtained from an AC analysis run on sweep data. In this graph, the signal `V0:p` is plotted against `frequency` on X-axis.



If you want to plot this graph against another sweep variable, `temperature`, and analyze the plot at a particular frequency, specify the frequency value in the *X Value* field in the Swap Sweep Var form (as shown in the figure below).

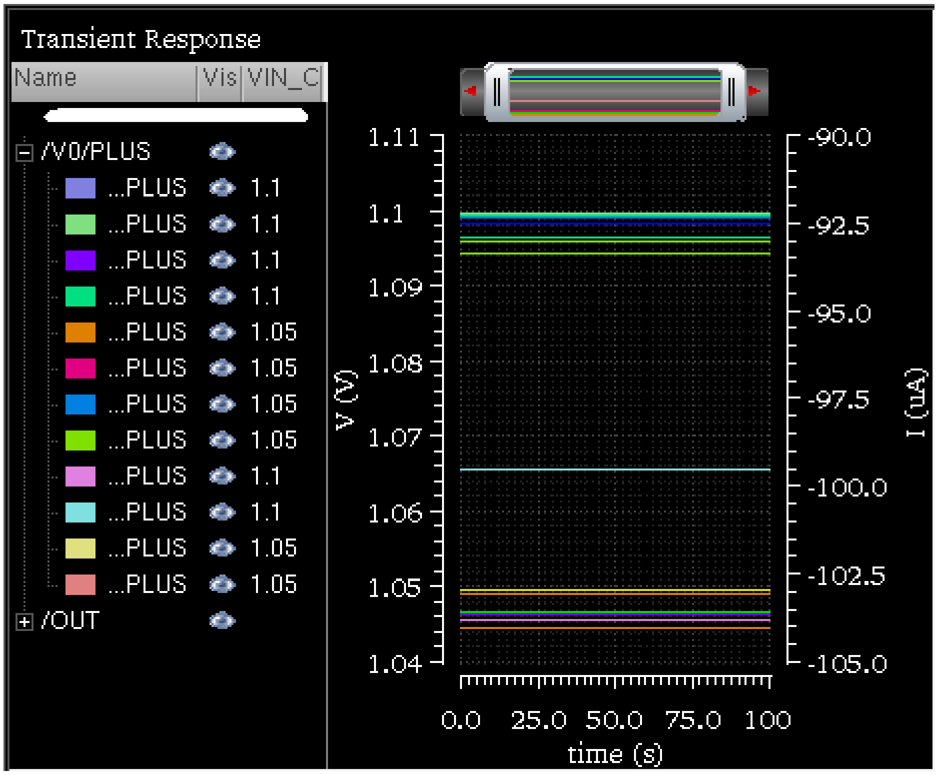


The graph is now plotted against temperature for frequency=1000 Hz (as shown in the figure below).

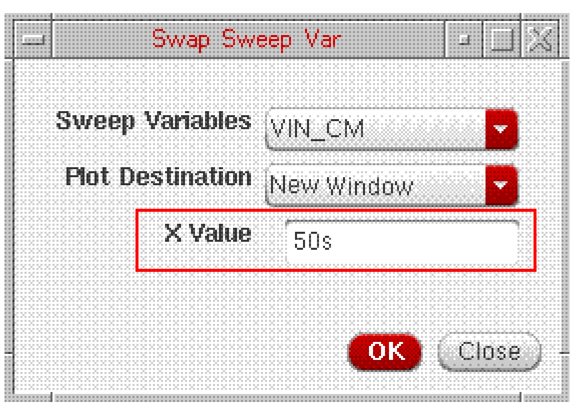


Plotting Graph across Fixed Time

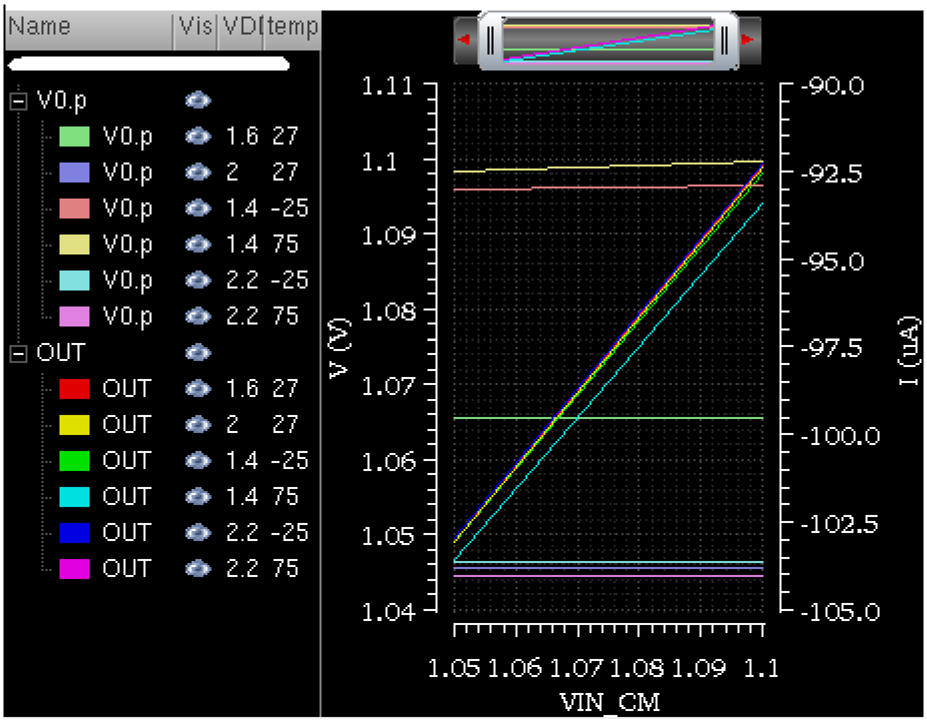
The graph shown in the figure below is obtained from a transient analysis run on sweep data. In this graph, the signal `V0/PLUS` and `OUT` are plotted against `time` on X-axis.



If you want to plot this graph against another sweep variable, `VIN_CM`, and analyze the plot at a particular time value, specify a time value in the *X Value* field in the Swap Sweep Var form (as shown in the figure below).



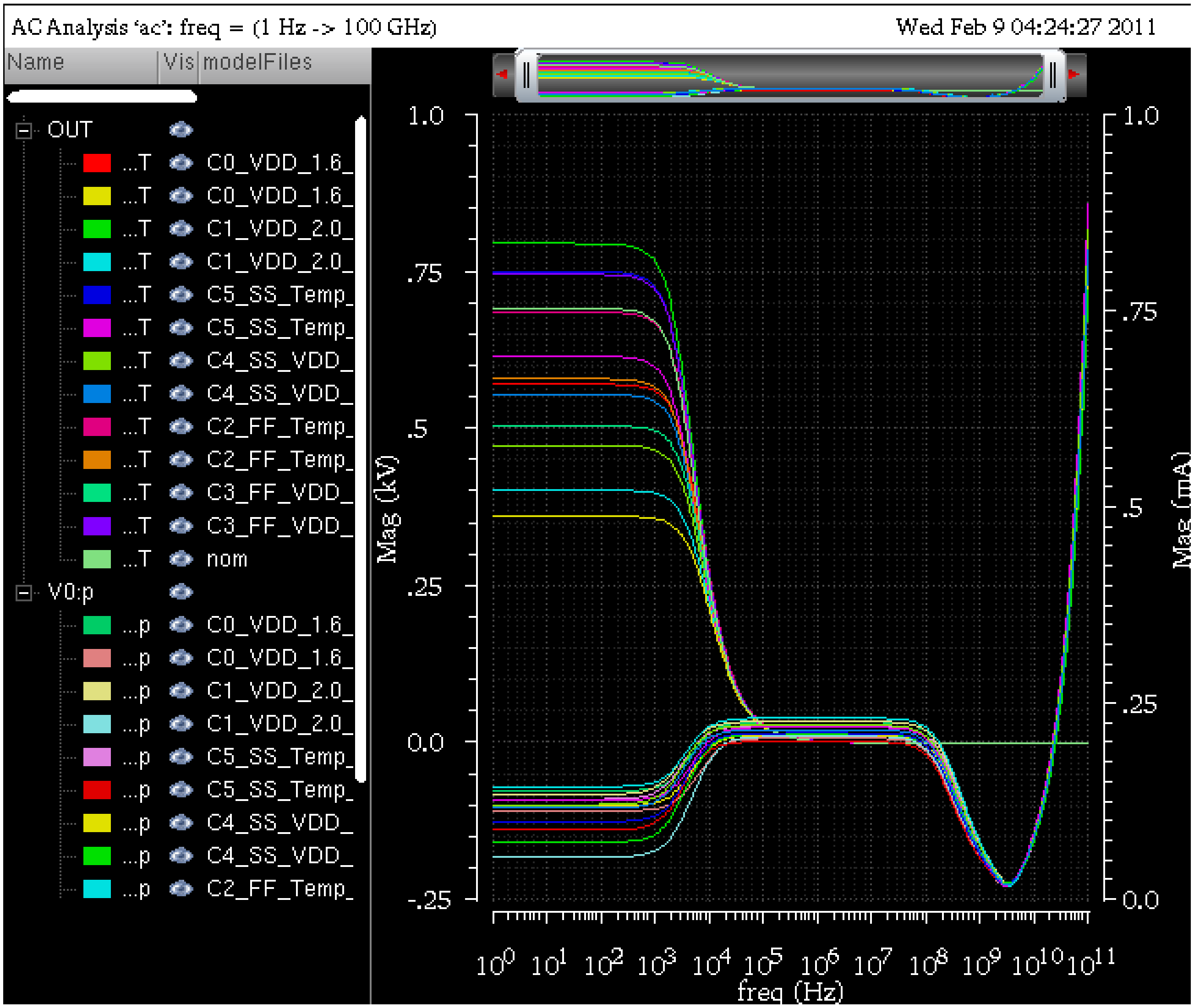
The graph is now plotted against `VIN_CM` for time=50s (as shown in the figure below).



Filtering Traces Using Sweep Visibility Filter

To display in a graph specific traces that belong to a selected sweep data range, you can filter the traces by using the sweep visibility filter option. Filtering helps you analyze the simulation data in a specific sweep range.

The figure below shows the `OUT` and `V0:P` traces that are plotted after running a simulation is run in ADE XL for sweep data. The X-axis sweep variables for this simulation are—*modelFiles*, *VDD*, *temperature*, *freq*, and *Corner*. The graph below is plotted with `freq` as the sweep variable on X-axis.

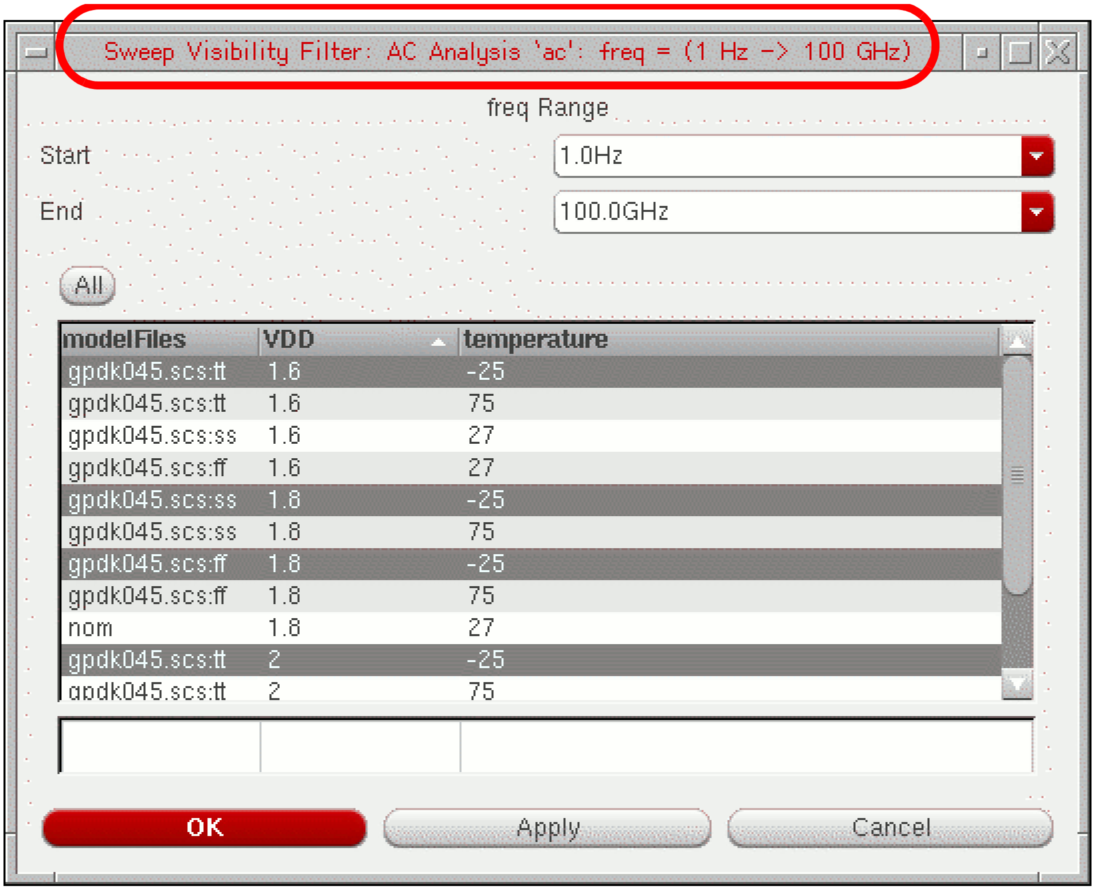


You can select different combinations of sweep variables to filter the visibility of traces that you want to display in the graph. For example, in the graph shown above, you can select a specific range of other sweep variables—*modelfiles*, *VDD*, and *temperature*—to filter traces. The traces that fall in the range you have selected are visible in the graph and the visibility of remaining traces is turned off.

To filter traces from sweep data, perform the following steps:

1. Choose *Graph – Filter By Sweep Var*.

The *Sweep Visibility Filter* form appears. The form name includes the subwindow name, as shown below.

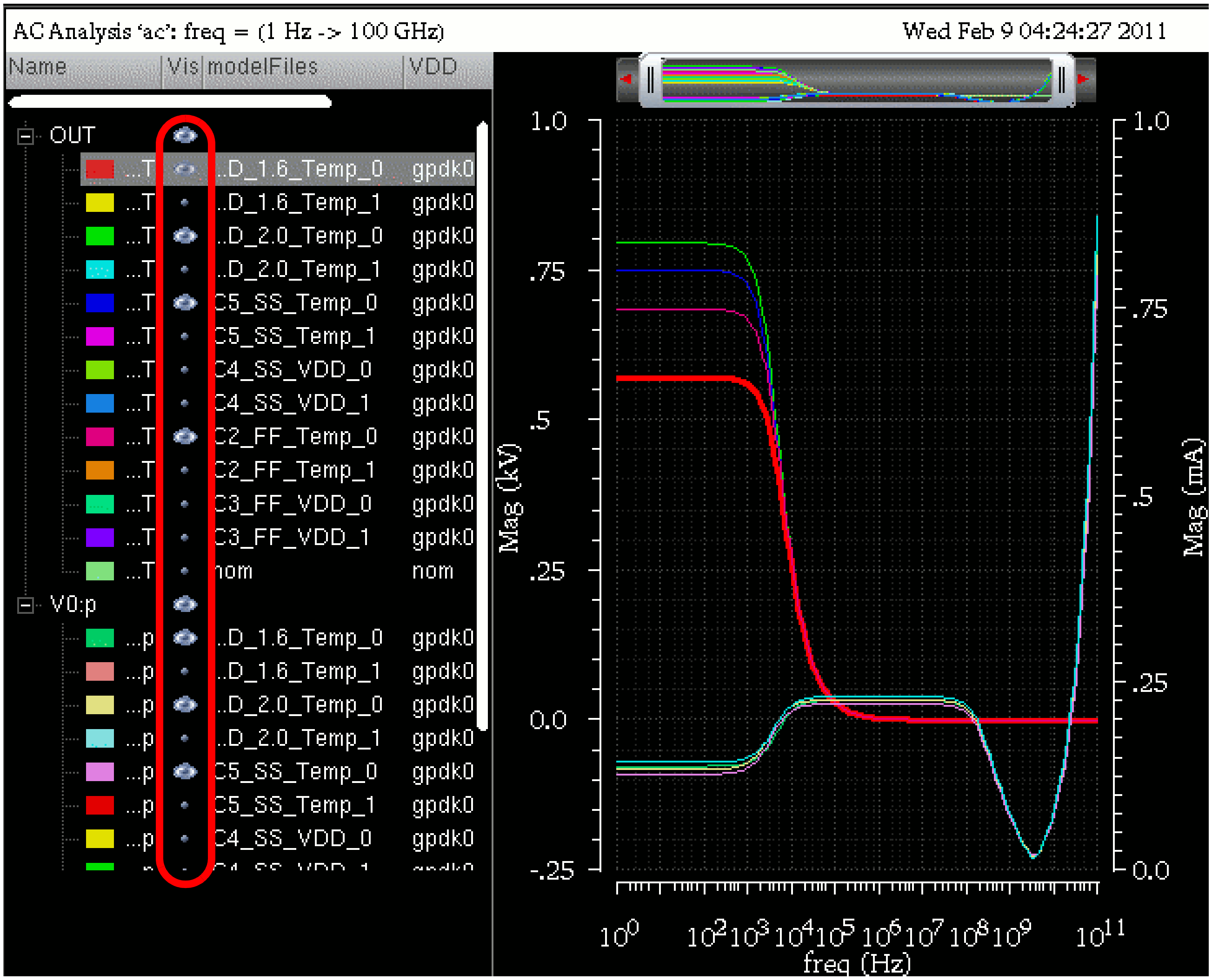


2. Select the variable values for which you want to display traces in the graph. For example, the form displayed in the figure above shows the following four traces selected for the *modelFiles*:*VDD*:*temperature* combination:

```
gpdk045.scs:tt, 1.6, -25
gpdk045.scs:ss,1.8, -25
gpdk045.scs:ff, 1.8, -25
gpdk045.scs.tt, 2, -25
```

3. Click *Apply*.

The traces for the selected combination of *modelFiles*, *VDD* and *temperature* values are displayed in the graph.



Note: The visibility icon in the trace legend area is ON only for the selected combination of sweep variables, as shown in the figure above.

You can also filter traces by selecting the *Set Sweep Ranges* option in the Results Browser. For more information, see Selecting and Plotting Signals in a Data Range on page 52.

Working with Graph Labels

You can add labels in a graph to display information about the graph or a trace. You can also attach labels with markers. For information about how to attach and edit the labels on marker, see Adding Markers on page 251.

To add a graph label in a graph, do one of the following:

- Select a graph and choose *Graph – Add Label.*
- Right-click anywhere in the window and choose *Create Graph Label*.

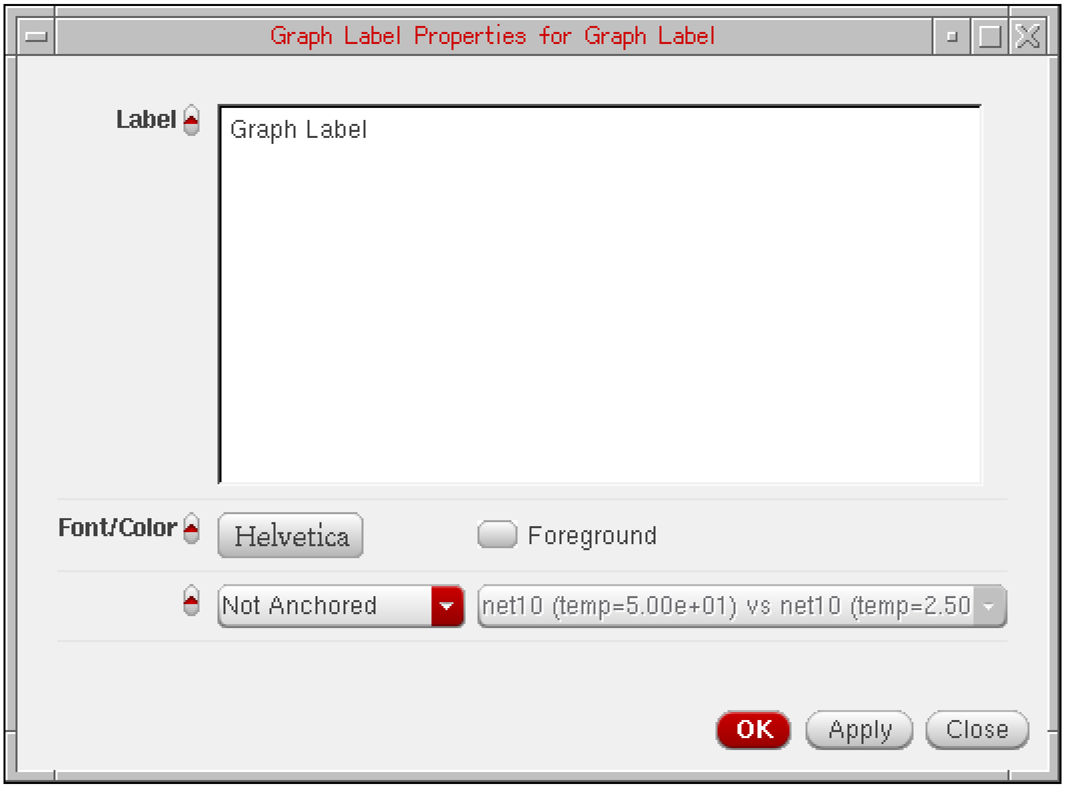
  A label is added to the graph you selected. By default, the label displays the string `Graph Label`. You can change the graph label by double-clicking the displayed string.

  Note: To add a new line in the graph label, press `Enter`. The new line and carriage return characters are also supported in the graph labels.

  To change the graph label properties, do one of the following:

  - Right-click the graph label and choose *Graph Label Properties*.
  - Select the label and press `Q`.

The *Graph Label properties for <Graph Label Name>* form appears.



This form includes the following fields:

- *Label*—Displays the default graph label name. You can provide a new name for the label.
- *Font*—Specifies the font type.
- *Foreground*—Specifies the foreground and background color of the graph label.
- Drop-down list box to select whether the label is an anchored frame or attached to a trace. You can select the trace name in the drop-down.

- Click *OK* to save the changes you made.

Note: You can change the position of a graph label by dragging the graph label to a new location.

Deleting Graph Labels

To delete a graph label, do one of the following:

- Select a label and choose *Edit – Delete*, or press the `Delete` key.
- Right-click the label and choose *Delete*.

Creating Multiple Graph labels

To create multiple graph labels, right-click a label in the graph and choose *Copy*. Then, right-click any where in the window and choose *Paste*. A copy of the graph label is created. Drag one of the labels to the required new position. Using this method, you can create as many labels as you want.

Note: You cannot move a label that is attached to a marker.

Plotting WREAL Signals

Virtuoso Visualization and Analysis XL supports the plotting of WREAL (wire-real) signals, where the WREAL signals are by default plotted in the sample and hold plot type. The WREAL signals are displayed with the symbol in the Results Browser.

Important

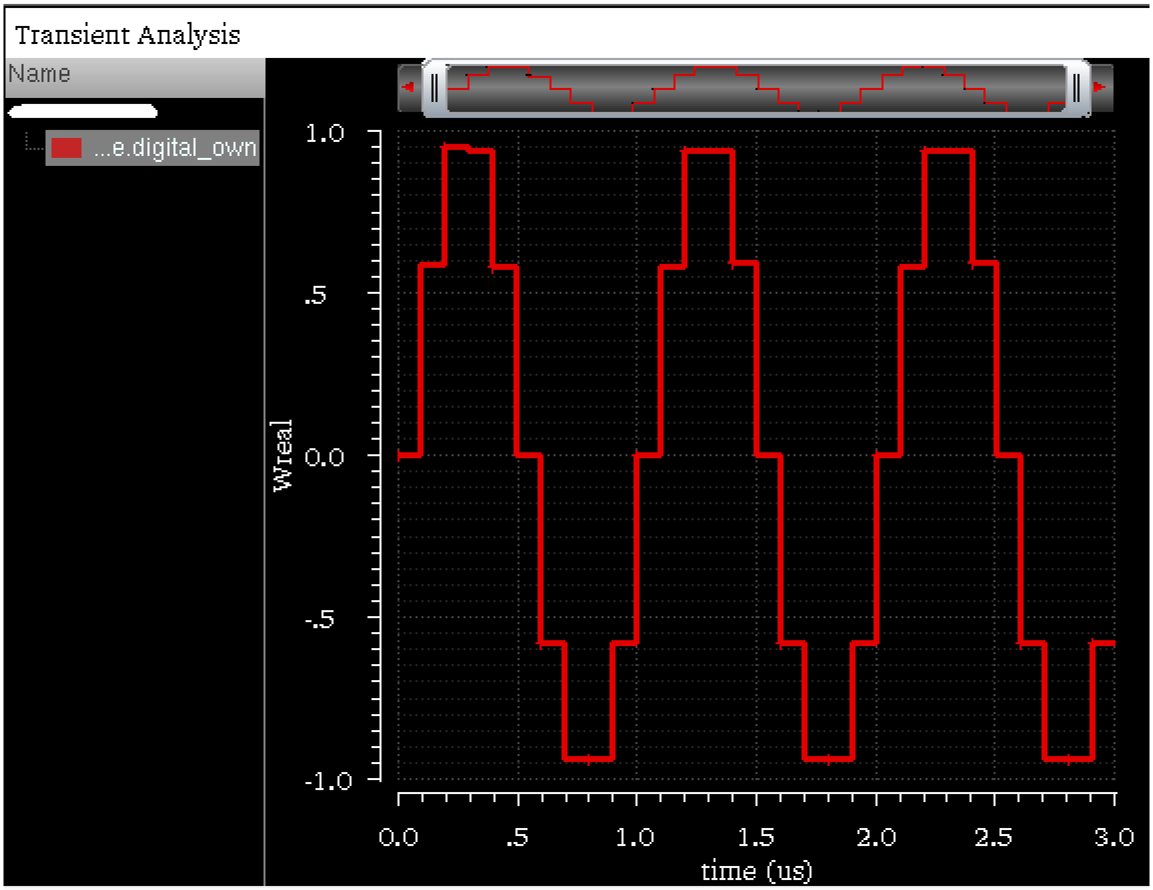
If you want to disable the WREAL plotting, set the `vivaWrealSupport` environment variable to `false`:

```
setenv vivaWrealSupport false
```

Perform the following step to set the plot type of a trace to sample and hold:

➡ Right-click the trace for a WREAL signal and choose *Type – Sample Hold*.

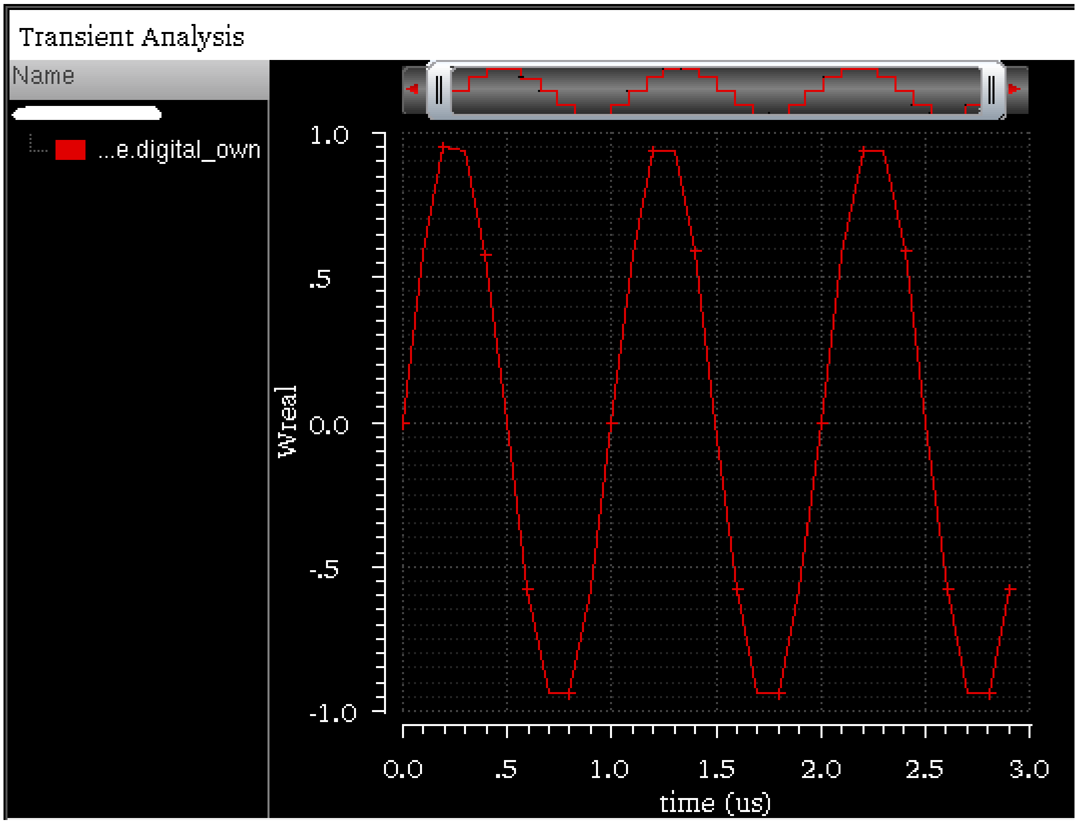
The following figure displays a WREAL signal plotted in the sample and hold plot type.



By default the data points are visible in the plots for the WREAL signals. To hide the data points, do one of the following:

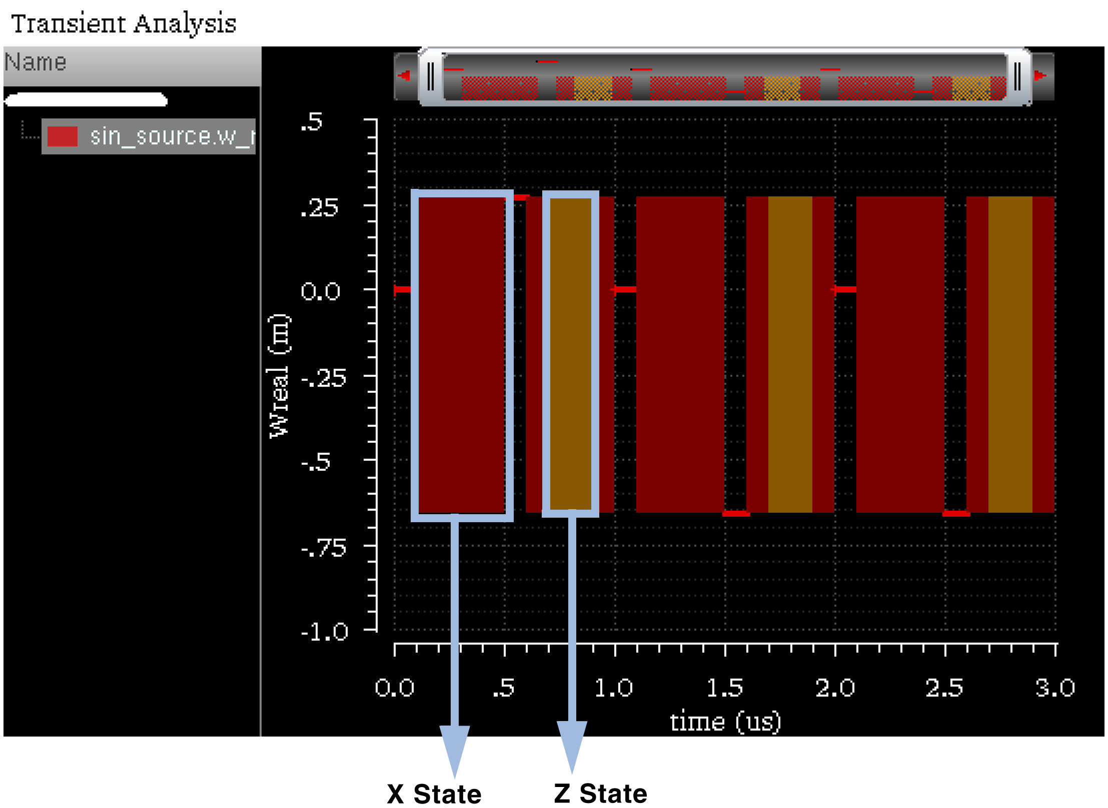
- Choose *Trace – Symbols On*.
- Right-click the trace and deselect *Symbols On*.
- Right-click the trace and choose *Symbol – Symbols Off*.

To change the plot type to a continuous line, right-click the trace and choose *Type – Continuous line*.



To change the properties of a WREAL trace, right-click the trace and choose *Trace Properties*. The *Trace Properties for <WREAL trace name>* form appears. This form includes the similar fields as that of the rectangular trace. For more infomation about the fom fields, see Setting Trace Properties on page 180.

WREAL plots includes a special depiction style to display the X and Z states. In the figure below, the red blocks indicate the X state and the yellow blocks indicate the Z state.



Limitations

Following are the limitations while plotting the WREAL signals:

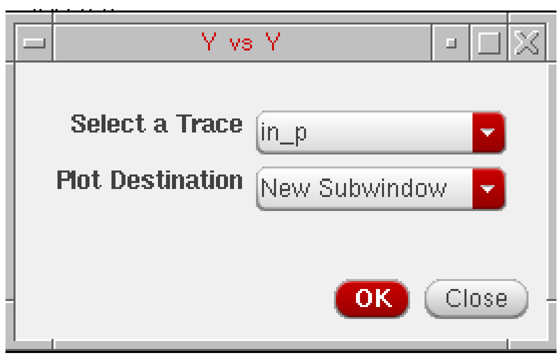
- The X and Z states are not visible when the trace is exported into a table.
- The X and Z states are not plotted when you plot the WREAL signal from ADE and Calculator.
- Single point WREAL signals are not displayed as a line.
- WREAL array data is not displayed as a group and each member of the array is displayed separately.
- Zooming a WREAL signal does not work properly.

Plotting YvsY Graph

To plot a YvsY graph for the sweep data, do the following:

- Right-click the X-axis and choose *YvsY.*

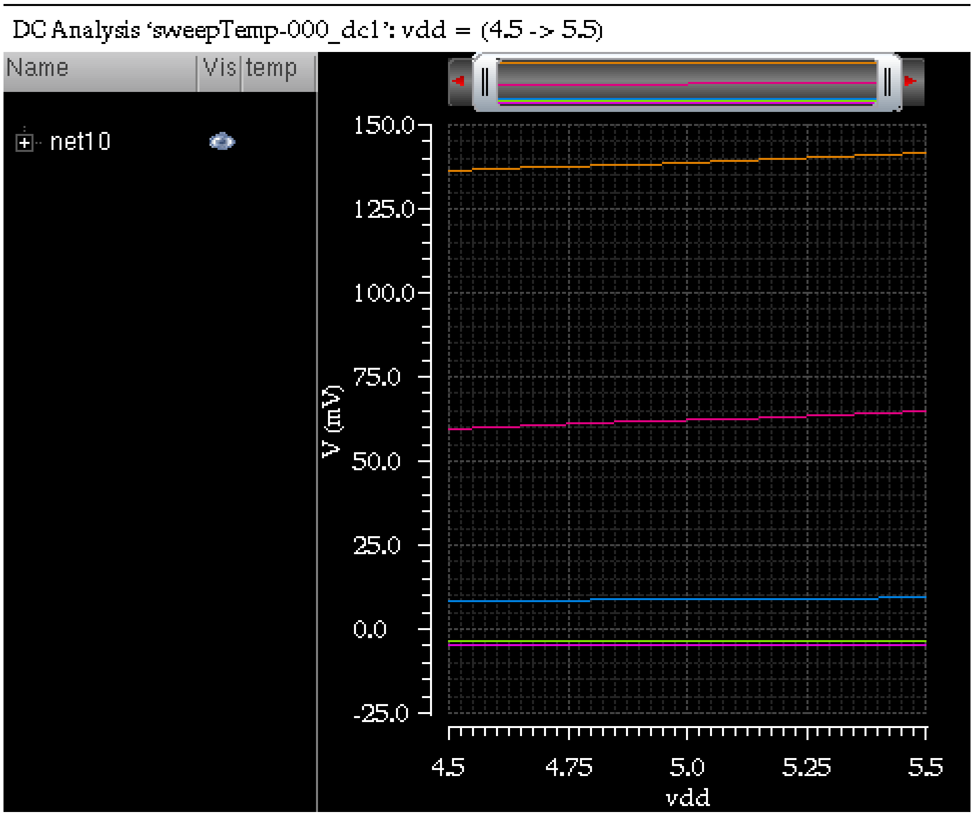
  The *YvsY* form appears. You cannot perform any action in the Virtuoso Visualization and Analysis XL window when the YvsY form is open.



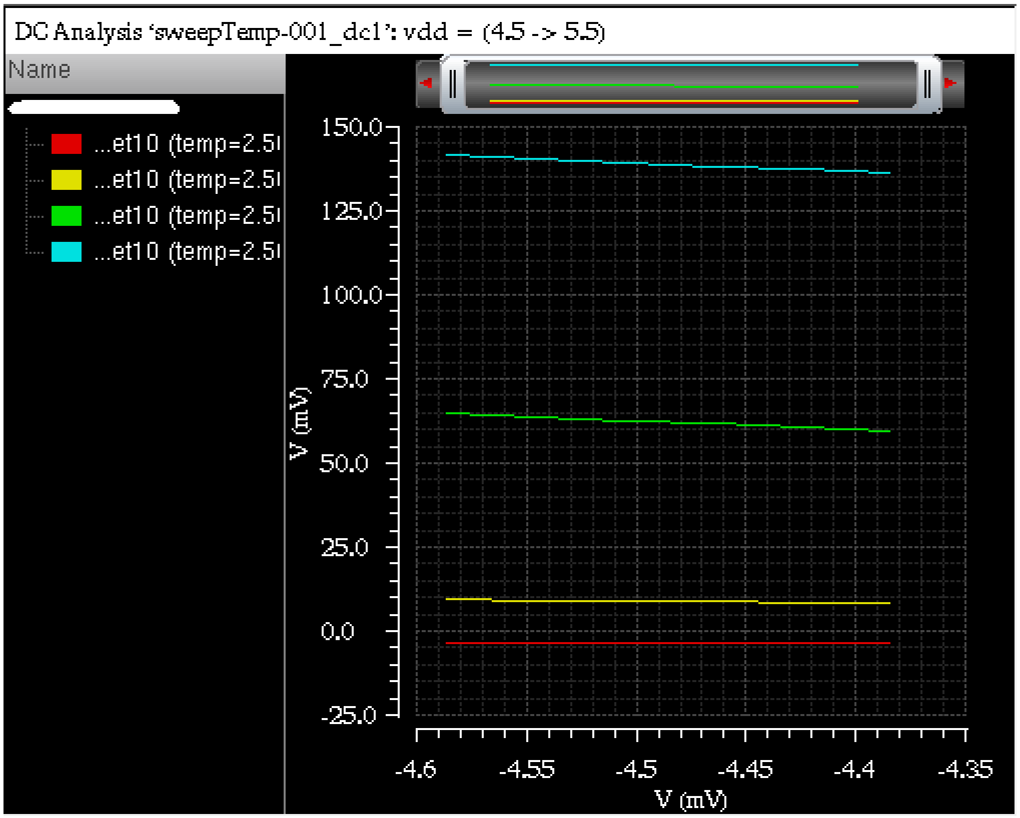
- Select a trace in this form and specify the plot destination, which can be a new window or a subwindow.
- Click *OK.*

  The YvsY plot is created in the destination window that you select. For example, the YvsY plot for `net10` is displayed below.

Signal `net10`:



YvsY of `net10`:



Plotting Normal Quantile Graphs

Currently, you cannot plot normal quantile graphs using the stand-alone mode of the Virtuoso Visualization and Analysis XL tool. You can plot normal quantile graphs only for the Monte Carlo results from the *Results* tab of the ADE XL window. For details on how to plot these graphs, refer to Plotting Histograms in the *Virtuoso Analog Design Environment XL User Guide*.

Saving and Loading Graphs

You can save a graph to a file for future use. When you save a graph, the graph settings, such as zooming and panning, changing font type and font color, setting labels for X and Y axis, or changing the trace color are also saved with the graph file. As a result, you save on the effort required to customize the graph when you display the same graph again.

When you load the saved graph in a window and append a trace to that window, the new trace is displayed with the same window attributes.

Note: If you open a new graph in a subwindow, the graph appears in the default graph attributes.

This section includes the following topics:

- Saving a Graph on page 210
- Loading a Graph on page 212
- Saving a Graph as an Image on page 213

Saving a Graph

The graph is saved as an XML file with the `.grf` extension. The following information is saved with the graph:

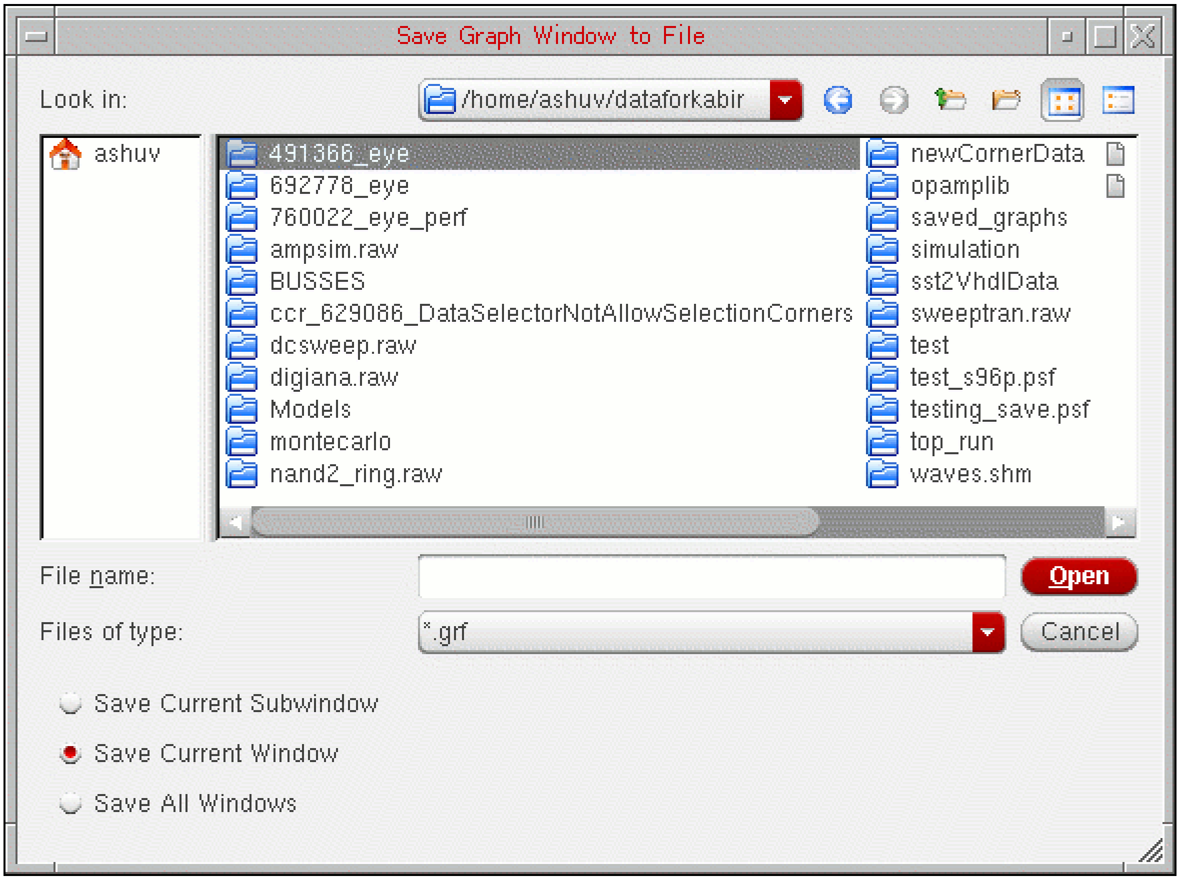
- The location of the data—data directory, data set, and trace name—and not the actual data. Therefore, if your simulation data changes between sessions, the graph reflects those changes.
- All graph objects and attributes, such as grids, background and foreground color, labels, and markers.

Save a group of windows with the file extension `.grf.group`.

To save a window, plot a signal in the window and do the following:

- Choose *File – Save Window As*, or press `Ctrl+S`.

The *Save Graph Window to File* form appears.



- In the *Look in* field, select the directory where you want to save the graph. The file extension is displayed as `.grf` in the *Files of Type* field.
- Now, do one of the following:
  - ❑ If you want to overwrite an existing graph file, select that graph file from the list box below the *Look in* field.
  - ❑ In the *File name* field, type a name for the graph file to which you want to save the graph.
- Select one of the following options:
  - ❑ *Save Current Subwindow*
  - ❑ *Save Current Window*
  - ❑ *Save All Windows*

When you select the *Save All Windows* option, the graphs are saved in a `.grf.group` file.

- Click *Save*.

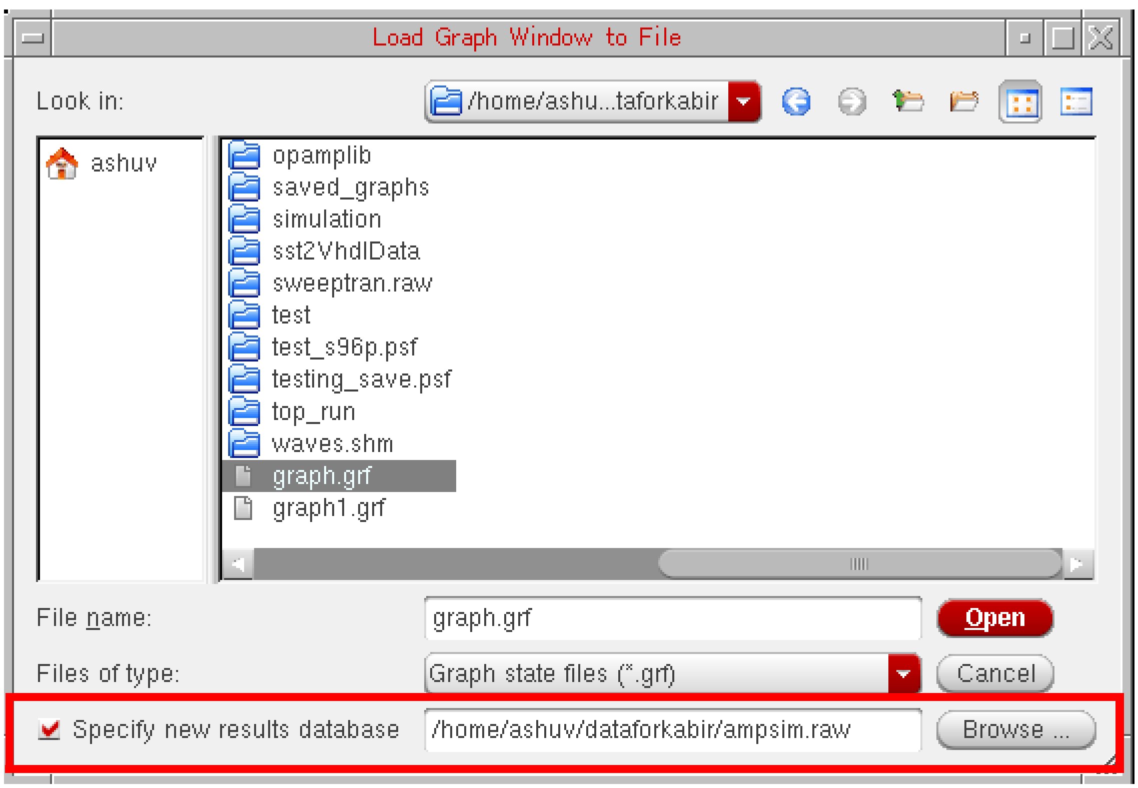
  The window that you selected is saved at the specified location.

Loading a Graph

Perform the following steps to load a graph that you have saved.

1. Choose *File – Load Window,* or press `Ctrl+L`.

   The *Load Graph Window to File* form appears.



   In the *Look in* field, select the directory in which the `.grf` file that you want to open exists.

   Select the *Specify new results database* check box and click the *Browse* button to load the same `.grf` file, either with the original data or different data.

2. Now, do one of the following:

- Select the graph file you want to open from the list box below the *Look in* field.
- In the *File name* field, type the name of the file you want to open.
- Select the `.grf` extension to display the graph files and `grf.group` file extension in the *Files of type* drop-down list box to select a graph file group. The extension for graph files is specified by the `filesuffix` variable in the `.cdsenv` file.
- Select the *Specify new results database* check box to plot the saved graph from a new results directory. Also, specify the name of the results directory from which you want to plot the graph. Ensure that the signal plotted in the saved graph exists in the results directory you specified.

  When you use this option, the saved graph is updated with the data from the new results directory and all the trace settings that you have applied to the saved graph are also retained. Hence, saved graphs can be used as a template when you reload a graph.

3. Click *Open*.

   The saved graphs are displayed in a new window. This graph has all the attributes that you saved with the graph.

   Note: If you have multiple windows open, loading a new window does not affect these windows.

Important

The `.grf` graph files saved from the previous IC6.1 releases, such as IC6.1.3, IC6.1.4, and the graph files saved from the IC6.1.5 Java graph cannot be loaded into the default graph in IC6.1.5.

Saving a Graph as an Image

If you save the graph as an image, the active graph or subwindow is saved with the trace legend area. You can insert the graph image into a document or print it.

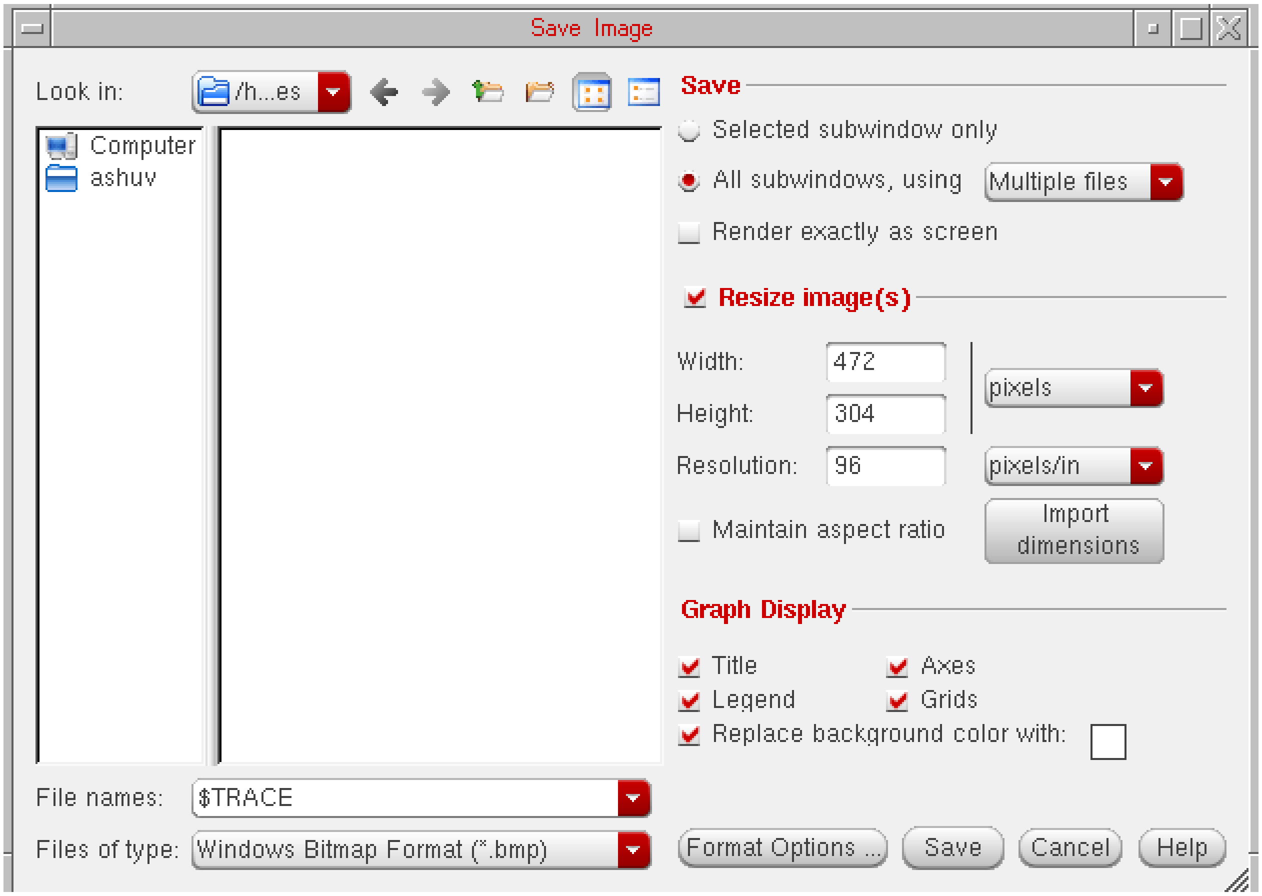
Note: You cannot load a graph image in a window.

The Virtuoso Visualization and Analysis XL tool provides several image formats to support a variety of applications and environments. While all image formats are functionally equivalent to binary storage formats, the size of a typical file varies greatly according to the format chosen. For example, a simple graph saved in the PNG format is typically less than 100 KB, while the same file saved in the TIFF or BMP format may exceed 1MB.

Perform the following steps to save a graph as an image file.

1. Plot a signal in a graph.
2. Choose *File – Save Image*

   The *Save Image* form appears.



3. In the *Look in* field, browse to locate the directory where you want to save the image file.
4. In the *File name* field, type a name for the image file. You can also specify macros, `$TRACE` and `$SUBWIN`, in the file name. These are used to save image files with the trace names and subwindows names.
5. In the *Files of type* drop-down list box, select the format in which you want to save the image file.

   By default, the file is saved in a format based on the file extension you specify. For example, if you type `output.png`, the file is saved in the PNG format. The image file can be saved with `.png`, `.bmp`, and `.tiff` (or `.tif`) file extensions. On the other hand, if you select *PNG (Best compressed)* and type `output.tiff`, the file is saved as `output.png`. Though PNG and TIFF files are compressed, there is no loss in image quality with these image formats.

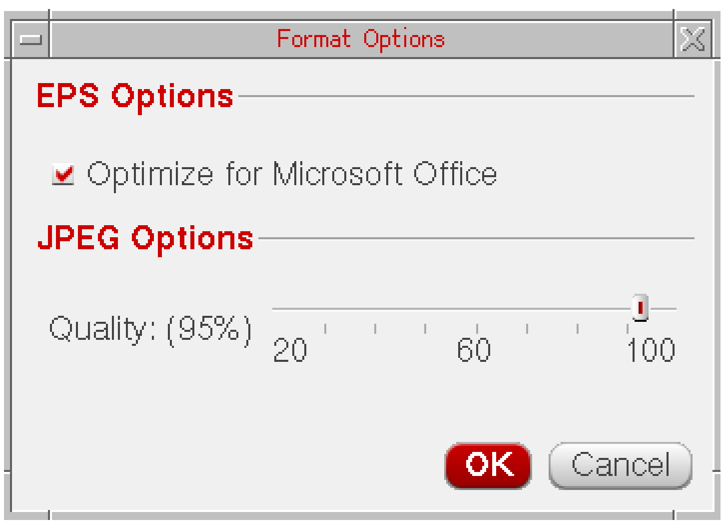
The graph image file can be saved in the following formats:

- Windows Bitmap Format (*.bmp)
- JPEG Format (*.jpg)
- Portable Network Graphics (*.png)
- Portable Pixmap Format (*.ppm)
- Tagged Image File Format (*.tif)
- X Pixmap Format (*.xpm)
- Encapsulated PostScript (*.eps)
- Adobe PDF (*.pdf)
- Scalable Vector Graphics (*.svg)

The default file type is `JPEG`.

6. To specify the format properties, click the *Format Option* button.

The *Format Option*s form appears. This form includes the following fields:



- *Optimize for Microsoft Office*—Select this check box if you want to import the image in the Microsoft office application. This option simplifies the image output so that it can be ready by Microsoft Office 2003 and 2007 applications.

  Note: If you select this option, any embedded font information is not saved.

- A quality bar is displayed under the *JPEG Options* section that you can use to specify the quality of the graph image on a scale of 20 to 100. Note that this quality bar is activated only if the file type is `JPEG`.

7. In the *Save* section, select the following fields:

- *Selected subwindow only*—Select this button if you want to save only the selected subwindow.
- *All subwindows, using*—Select this button if you want to save all the subwindows. You can use the drop-down list to specify whether you want to save subwindows in a single file or multiple files.
- *Render exactly as screen*—Select this check box if you want to save the exact copy of graph as it is visible on the screen. This also saves all attributes of the graph as well as zooming and panning properties. When you select this option, the *Graph Display* section is disabled.

This option saves only those strips that are visible on the active graph. If you do not want to save a strip, you can adjust the graph window size to hide the strip.

8. *Resize Image(s)*—Select this check box if you want to resize the image. In this section, you can change the following image attributes:

- *Width*—Specify the width of the image.
- *Height*—Specify the height of the image
- You can select the units for the height and width from the drop-down list displayed at the right. The available units are `pixels`, `inch`, `cm`, `mm`, `picas`, and `points`. When you change the image units, the height and width values you have specified are automatically changed as per the selected units.
- *Resolution*—Specify the resolution of the image. This field is unavailable if you select the image type in the vector format. Select the resolution type as `pixels/cm` or `pixels/in` from the drop-down list displayed at the right. When you change the resolution type, the height and width fields also change accordingly.
- *Maintain Aspect Ratio*—The ratio of the width of the image to its height. Select this check box if you want to maintain the aspect ratio while modifying the height or width of the image.
- *Import Dimensions*—Click this button to specify the default values for height, width and resolution.

9. *Graph Display*—In the this section, specify the following fields:

- *Title*—Select this check box if you want to display the graph title in the graph image

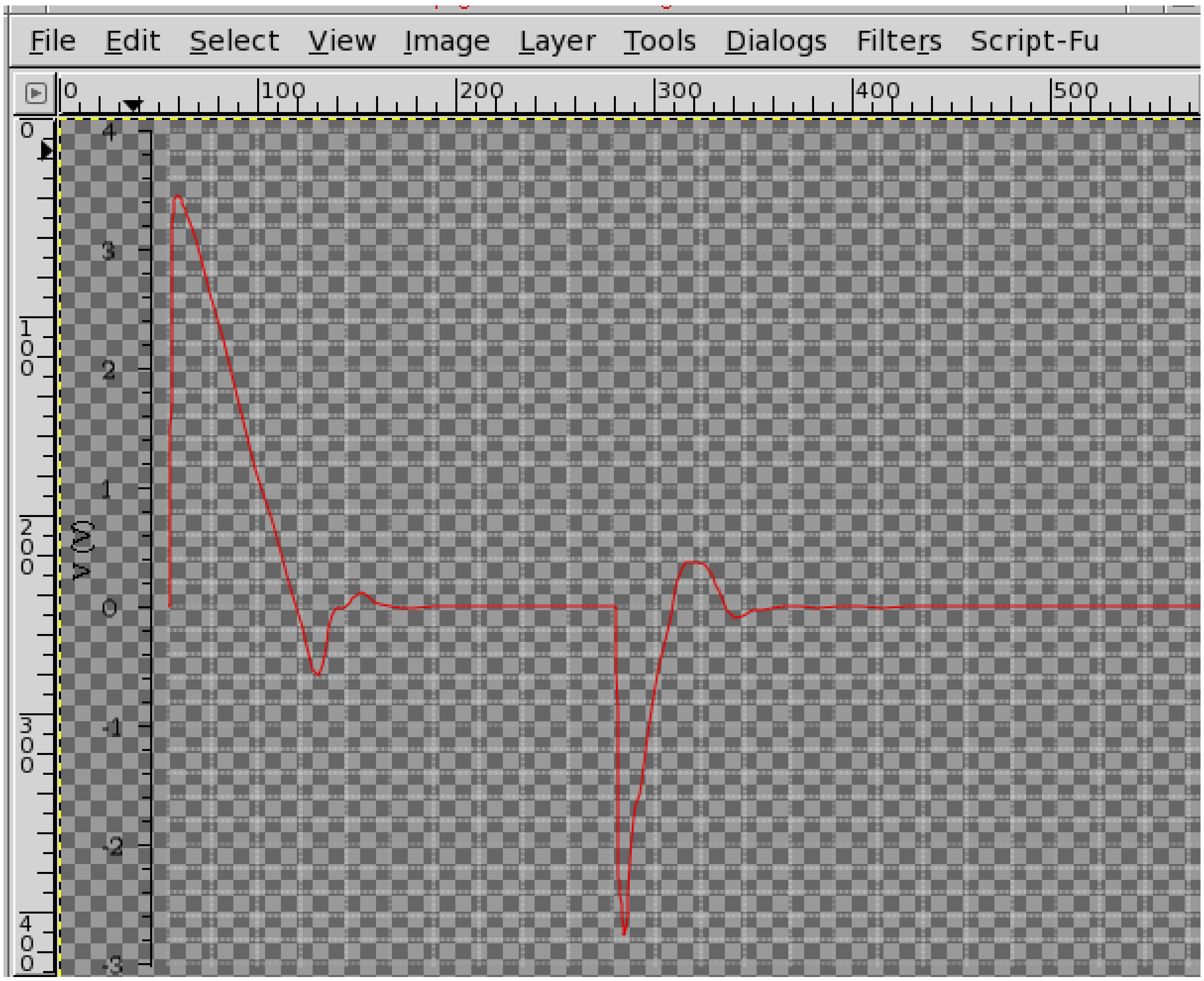
- *Legend*—Select this check box if you want to display trace legend in the graph image.
- *Axes*—Select this check box if you want to display axes in the graph image.
- *Grids*—Select this check box if you want to display grids in the graph image.
- *Replace background color with:*—Select this check box if you want to save the graph image in a different background color. You can specify a new color by clicking the button provided with this option. This option is not available when the file type is `pdf` and `svg`.

Note: All the above options are selected by default.

10. Click *Save*.

The graph is saved as an image file with the specified attributes.

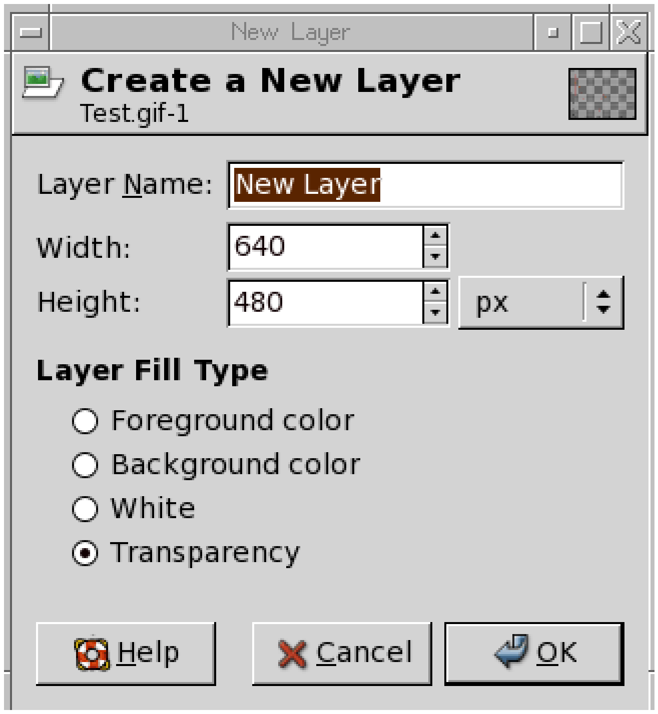
When you do not specify a background color and save the graph as an image, the saved image may not be in readable form. For example, when you open the saved image by using an image viewer, such as GIMP, the graph appears as shown in the figure below:



You can use the GIMP image viewer and editor to convert the image into a readable format by performing the following steps:

1. Choose *Layer – New Layer*.

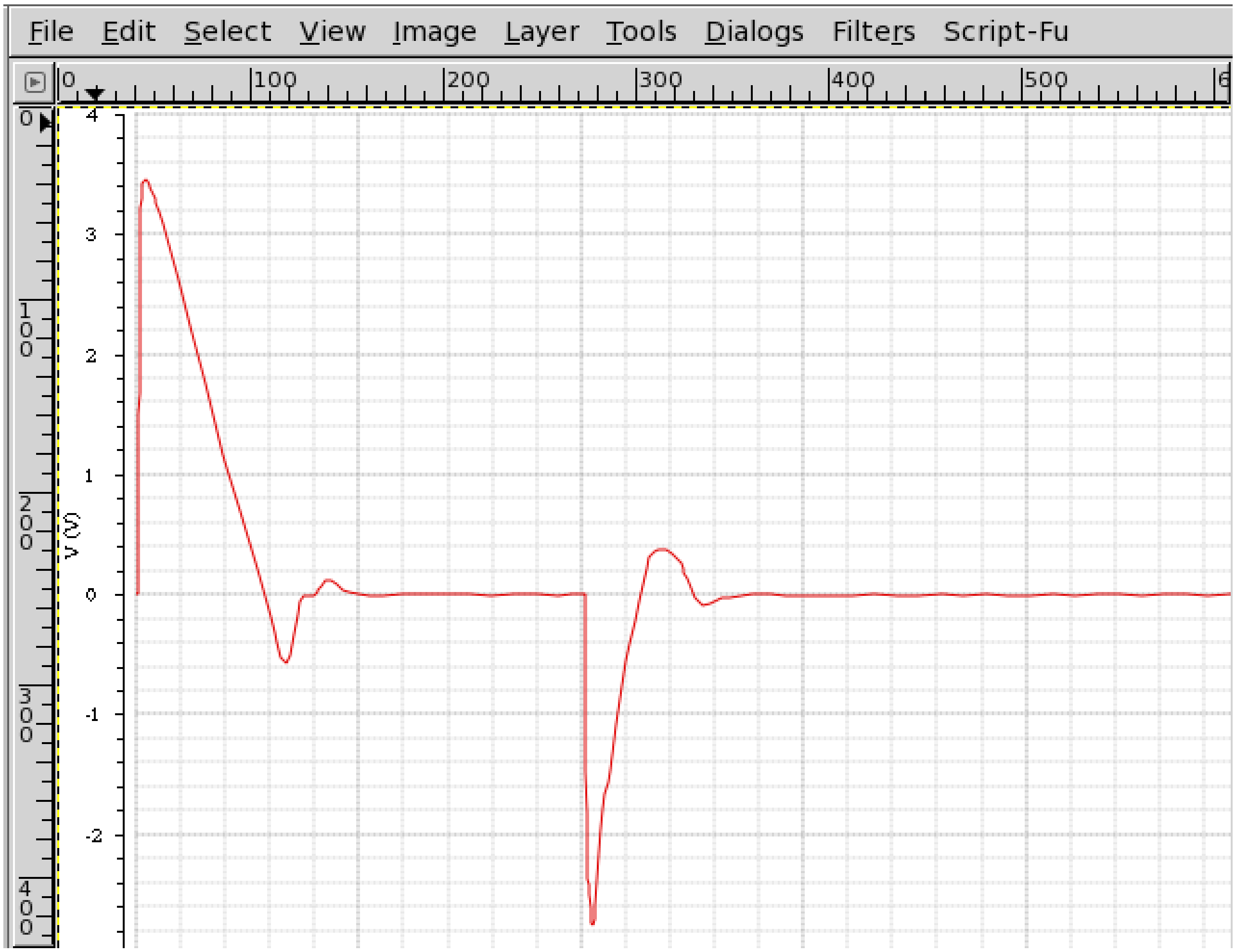
The *New Layer* form appears.



2. In this form, select the *Layer Fill Type* field as *Background color* and click *OK*.

   The image is filled with the white background color.

3. Choose *Layer – Stack – Layer to Bottom*. The image is now in readable format.



Reloading Graphs

Reloading a graph updates the already plotted traces with the latest simulation results based on the current in-context results directory. When you reload a graph, the settings that you have applied to a trace are also applied to the reloaded traces, such as, background color and font. For more information about the trace settings that are retained during reloading a graph, see Graph Settings Retained During Reloading on page 227.

You can reload a graph when you open the Virtuoso Visualization and Analysis XL tool in the stand-alone mode and also when you run the tool from ADE L and XL.

This section includes the following topics:

Reloading Graph When Opened in Stand-Alone Mode

In the stand-alone mode, you can reload an already open graph with the simulation results based on the current in-context results directory selected in the Results Browser.

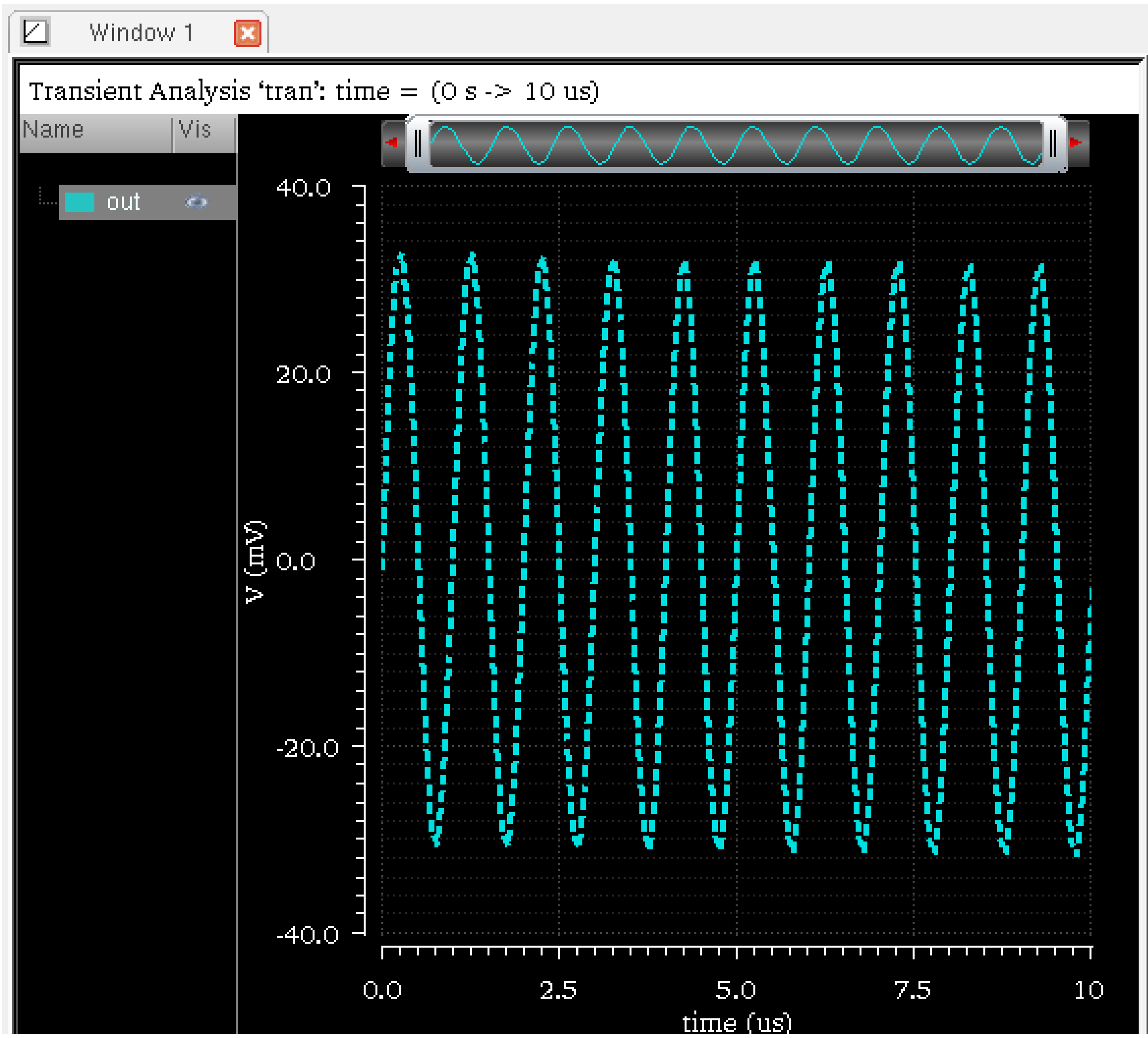
Following are the examples that describe how reloading works in the stand-alone mode:

Example 1

Consider the following scenario in which you plot a trace from a results directory, change the in-context results directory, and reload the trace with the data from a new in-context results directory.

1. Plot a signal, `out`, from an in-context results directory (`simulation1`) in a new window.

2. Set the trace *Style* to `Dot` (as shown in the figure below). Notice that the time range of the `out` signal varies from 0.0 to 10 microseconds.



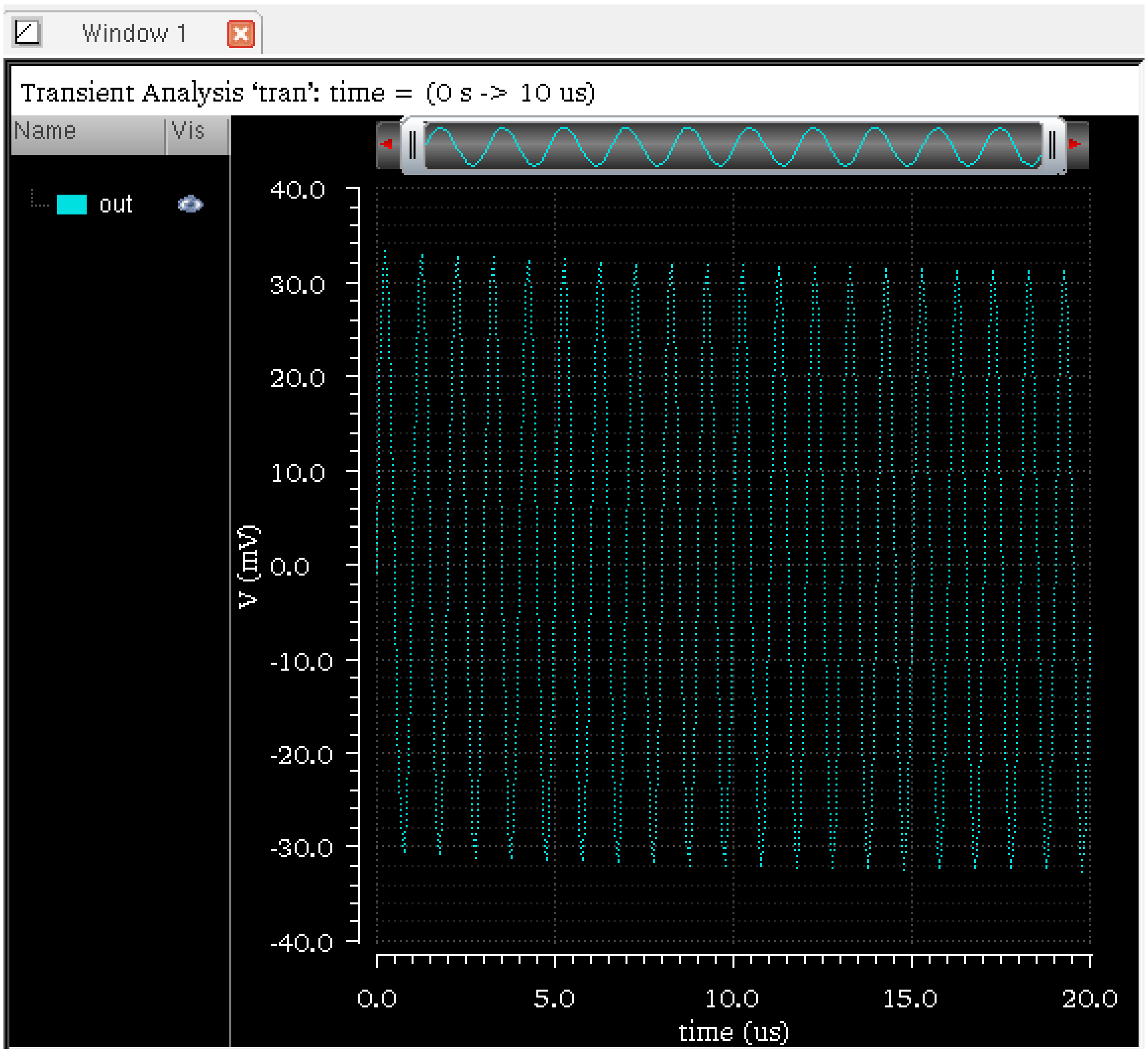
3. Open another results directory, `simulation2`, which contains the simulation results for the same design.

4. Set the database context to this new results directory, `simulation2`. For more information about how to change the in-context results directory, see Changing In-Context Results Directory on page 42.

5. Now, to reload the already plotted trace for the `out` signal from the previous results directory (`simulation1`) with the data from the new in-context results directory (`simulation2`), choose *File – Reload – Current Subwindow.*

Note: Alternatively, you can press `Ctrl+R` to reload graphs in the current window.

The trace for the `out` signal is reloaded using data from the latest in-context results directory, `simulation2`, and all the trace settings are retained (as shown in the figure below). Notice that the time range of the reloaded trace now varies from 0.0 to 20 microseconds.



To reload the traces in all the subwindows, choose *File – Reload – All Subwindows*. All the traces for the common signals are updated with data from the current in-context results directory and all the trace settings are retained.

To reload the traces in all the windows in Virtuoso Visualization and Analysis XL, choose *File – Reload – All Windows*.

Example 2

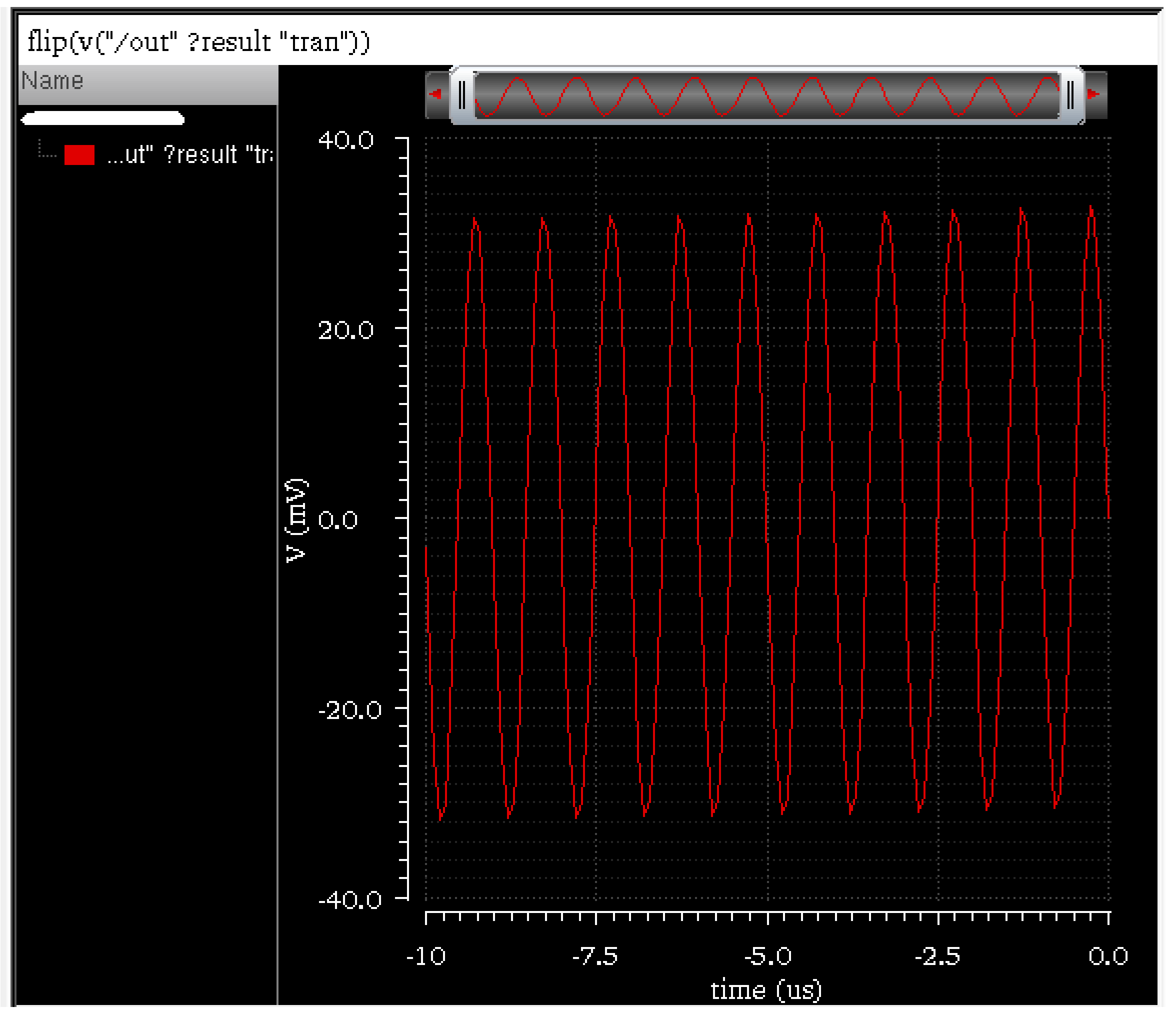
You can also reload the traces for expressions created in the Calculator. Consider the following scenario in which you plot a trace for an expression, change the in-context results directory in Results Browser, and then reload the plotted trace with the data from the new in-context results directory

1. In the Calculator, create an expression for the signal, `out`, from `simulation1` results directory. The following expression is displayed: `v("/out" ?result "tran")`.

   Note: `Simulation1` is the in-context results directory selected in the Results Browser.

2. Apply `flip` function to this expression. The expression changes to: `flip(v("/out" ?result "tran"))`

3. Now, evaluate this expression and plot the output in a new window. The following output trace is displayed:

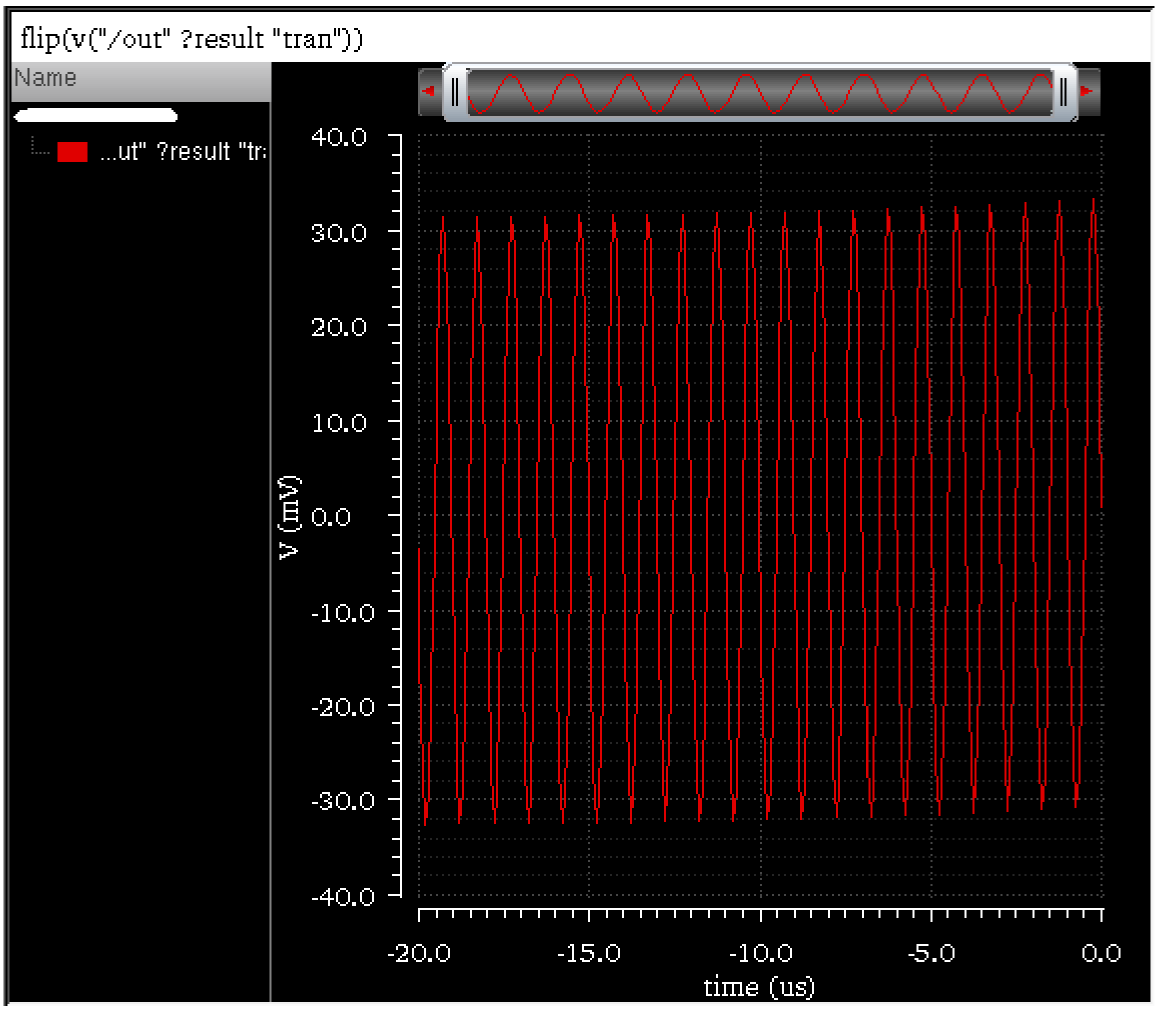


Notice that the time range of the output trace varies from -10 to 0.0 microseconds.

4. In the Results Browser, change the in-context results directory to `simulation2`, which contains the simulation results for the same design.

5. Choose *File – Reload – Current Window*.

   The trace for the expression you plotted earlier is updated with the data from the new in-context results directory (see figure below). Note that the time range of the updated trace now varies from -20 to 0.0 microseconds.



If you create an expression for a signal from a results directory that is not set as the in-context results directory in Results Browser, the expression for this signal also displays the path of the results directory. For example,

```
v("/out" ?result "tran" ?resultsDir "./simulation_10/ampTest/
spectre/config/psf")
```

This indicates that the database context of this signal is specified within the expression. When you plot the expression, it is always plotted with the specified results directory. The context for this expression does not change with the change in the in-context results directory in the Results Browser; and therefore, when you perform *File – Reload* on this trace, it is not updated with the new data.

Reloading Graph When Opened From Within ADE L and XL

While working in ADE L and XL, you can use the Refresh plotting option to update already open graphs with current simulation results. This option retains all the trace settings that you have applied to the traces in open graphs.

Note: Only those open graphs are updated that are common in the current simulation run.

Note: You can also use *File – Reload* to reload the graphs plotted from within ADE L and XL based on the results from a new in-context results directory selected in the Results Browser.

To know more about refreshing graphs through ADE XL, see Refreshing Graphs in *Analog Design and Environment XL User Guide.*

To know more about refreshing graphs through ADE L, see Refreshing Graphs in *Analog Design and Environment L User Guide.*

Disabling Trace Reload

You can lock the database context of a trace to disable the trace reloading. You use this feature if you do not want to update the trace while reloading graphs with data from the in-context result directory.

Note: The trace for which you lock the database context are not updated when you do *File – Reload.*

To disable the reloading of a trace, do one of the following:

- Select the trace and choose *Trace – Disable Reload*.
- Right-click a trace and choose *Disable Reload*.

A lock appears on the trace color symbol displayed in the subwindow title, which indicates that the context of the selected trace is locked and it is not updated when you reload traces with data from the in-context results directory.

Graph Settings Retained During Reloading

The Virtuoso Visualization and Analysis XL graph window saves and maintains the following settings for the graphs when you reload the graphs with the data from the latest in-context results directory:

- Trace color, type, style, width, or symbols
- Visibility status of graphs
- Axes settings
- Pan and zoom settings
- Graph layout
- Strip layout
- Markers and marker locations

How Marker Locations Change After Graph Reloading?

- Point Marker—If a point marker is attached to a trace and the trace remains plotted after you reload graphs, the point marker stays attached to the trace on the same X-axis value. However, the marker is snapped to the nearest Y-axis value on the trace.

  If a point marker was detached from the trace before reloading, the point markers remain detached and continues to exist at the same XY location.
- AB Marker—If an AB marker is attached to a trace and the trace remains plotted after you reload graphs, the AB marker stays attached to the trace on the same X-axis value; however, the marker is snapped to the nearest Y-axis value on the trace. If the trace is deleted after refreshing, the corresponding AB marker is also deleted. This results in the display of only A and B markers on the graph without a delta line between them.
- Delta Marker—If a delta marker is attached to a trace and the trace remains plotted after you reload graphs, the delta value is updated based on the new point marker locations. If the point markers do not exist after reloading the graph, the corresponding delta marker is also deleted.

Printing Graphs

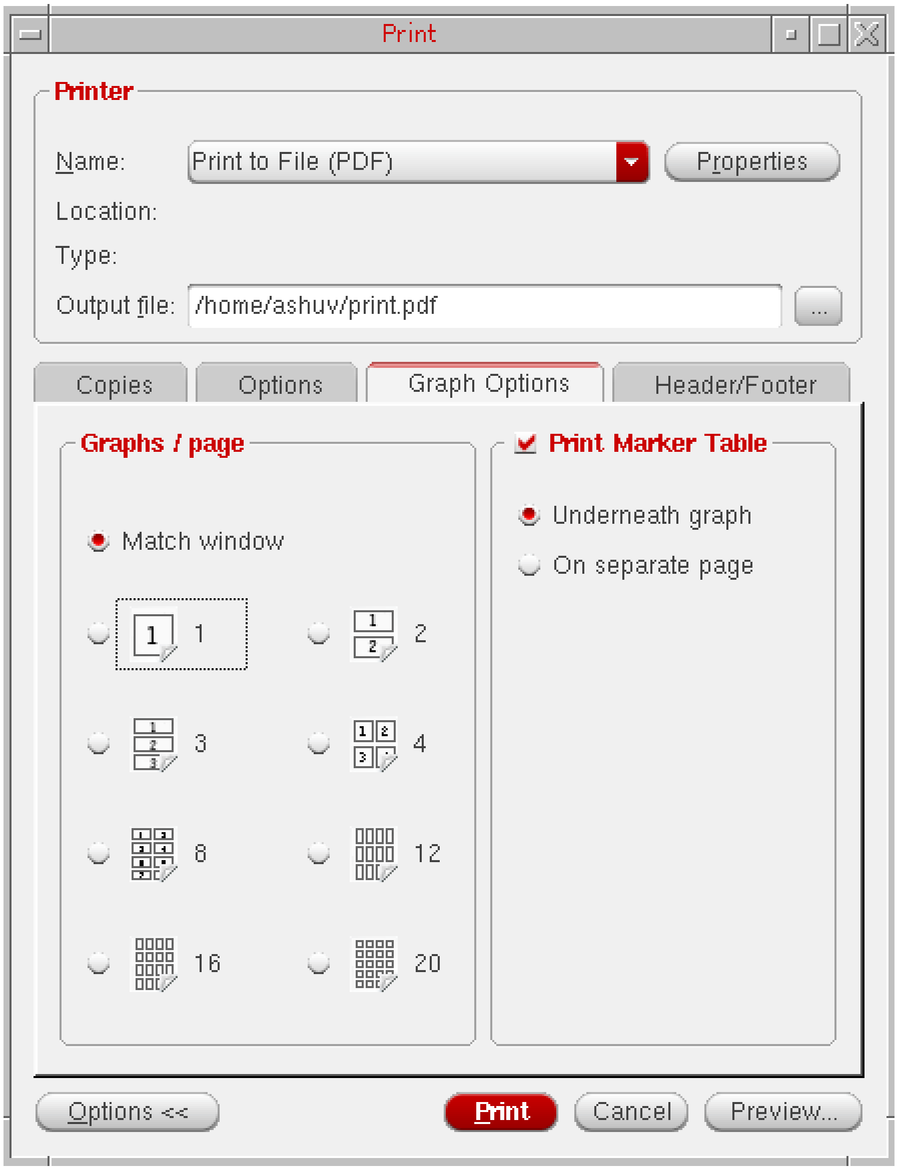
You can save a selected graph or all the graphs in the selected window to a file in PDF or Postscript (PS) format, and then print the graphs on a network printer that you have installed on your computer. The Postscript format is fully scalable and is used in many UNIX printers.

Before you print, you can use the print preview mode to see how the graph will look after it is printed.

Perform the following steps to print a graph:

- In the window, choose *File – Print*.

The *Print* form appears.



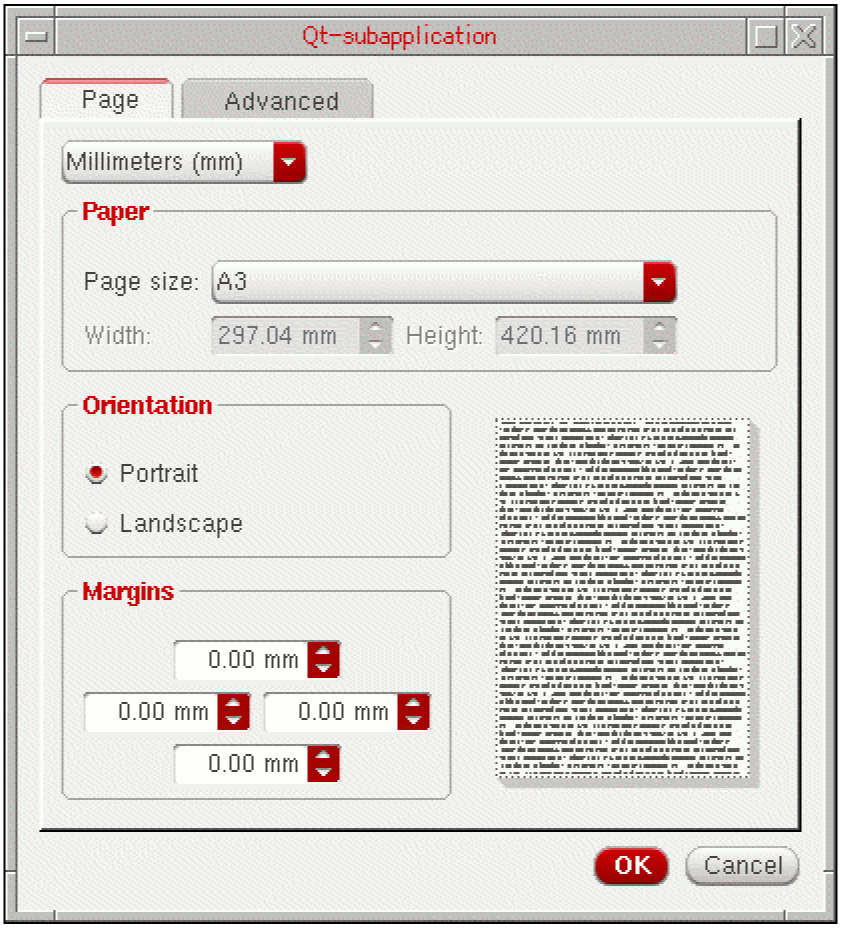
In the *Printer* group box, do the following:

❑ In the *Name* drop-down list box, specify the destination. The destination can be a PDF file or a Postscript file.

If you select *Print to file (PDF)*, the *Location* field displays *Local file* and the *Type* field displays *Write PDF file.* If you select *Print to file (Postscript)*, the *Location* field displays *Local file* and the *Type* field displays *Write Postscript file.*

❑ To set properties for the print job, click the *Properties* button next to the *Name* field.

The *Qt-subapplication* form appears. This form has two tabs—*Page* and *Advanced*. The *Advanced* tab is currently not available.



- On the *Page* tab, specify the following fields:
  - Select the units, such as *Centimeters (cm),* or *Millimeters (mm)*, in which you want the paper size to be displayed.
  - In the *Paper* group box, select the required page size in the *Page size* drop-down list box. The *Width* and *Height* fields display the default paper settings for that page size in the specified units.

Note: The default settings are defined by the `LC_ALL` environment variable on your computer. Reset this variable to use your original settings while loading a postscript file.

  - In the *Orientation* group box, select the print orientation as *Portrait* or *Landscape*.

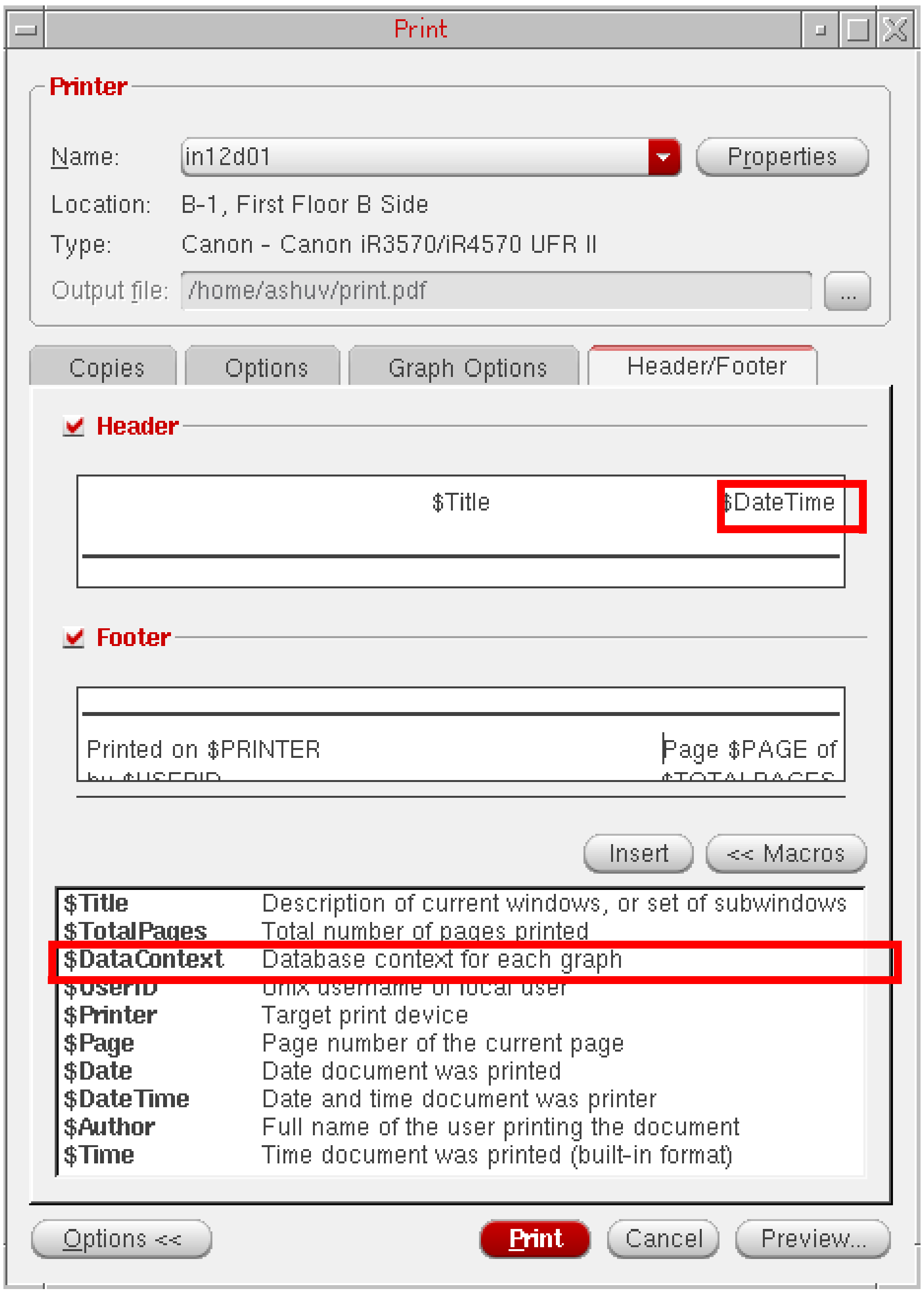
- In the *Margin* group box, select the top, left, right, and bottom margins.
- Click *OK* to save the settings.

- In the *Output file* field, specify the name and location of the output file. To specify a file at a location other than the default, click the browse button. By default, the name of the output PDF file is `print.pdf` and the name of the output postscript file is `print.ps`.
- Click the *Options* button to view the other printing options. When you click this button, two tabs are displayed in the *Print* form—C*opies* and *Options.*
  - On the *Copies* tab, do the following:
    - In the *Print range* group box, specify the range of pages that you want to print. By default, all pages are printed.
    - On the *Output Settings* tab, specify the number of copies you want to print. In addition, specify the type in which you want the pages to be printed—*Collate* or *Reverse*.
  - On the *Options* tab, do the following:
    - Specify *Duplex Printing* as *None, Long side*, or *Short side*.
    - Specify *Color Mode* as *Color* or *Grayscale*.
- On the *Graph Options* tab, do the following:
  - In the *Graphs/page*, specify how many graphs you want to print on a page.

    If you want to print all the subwindows in a PDF file in the same order in which they are arranged in the graph, select the *Match window* check box. However, if you do not select this check box, the visibility settings of columns in the trace legend are reflected in the output PDF file. Also, if you adjust the width of the trace legend area and hide few columns, only visible columns are printed in the output file.
  - Select the *Print Marker Table* check box to print marker tables for the graphs. In addition, specify how you want the marker tables to be printed—*Underneath graph* or *on separate page*.
- On the *Header/Footer* tab, do the following:
  - Click the *Header* and *Footer* check boxes if you want to display header and footer in the output file.
  - Specify the information that you want to display in header and footer of the output file. Click the *Macro* button at the bottom of the form, a table listing all the available macros appears. Place the pointer in one of the boxes provided below the *Header*

and *Footer* check boxes where you want to insert the macro. Select a macro in the table and click the *Insert* button.

The selected macro is inserted at the specified location. For example, if you want to display date and time in the output file header in the right-most corner, you can insert the `$DateTime` macro, as highlighted in the figure below.



The table below contains a list of macro that you can specify:

Macro	Description
$Title	Displays the window or subwindow title.
$TotalPages	Displays the total number of pages that are printed.
$DataContext	Displays the results directory for the plotted expressions and signals in the graph. Note that this macro does not display the current in-context results directory in the Results Browser. If the signals from multiple results directories are plotted and then printed, specifying this macro in header or footer prints the name of all the databases that are used for plotting.
$UserID	Displays the user's UNIX login name.
$Printer	Displays the name of the printer.
$Page	Displays the current page number.
$Date	Displays the date when the document is printed.
$DateTime	Displays the date and time when the document is printed.
$Author	Displays the full name user name.
$Time	Displays the time when the document is printed.

- Click *Preview* to generate the print preview.
- Click *Print*.

Note: The upper-right corner of the PDF displays the time and date when the PDF is created.

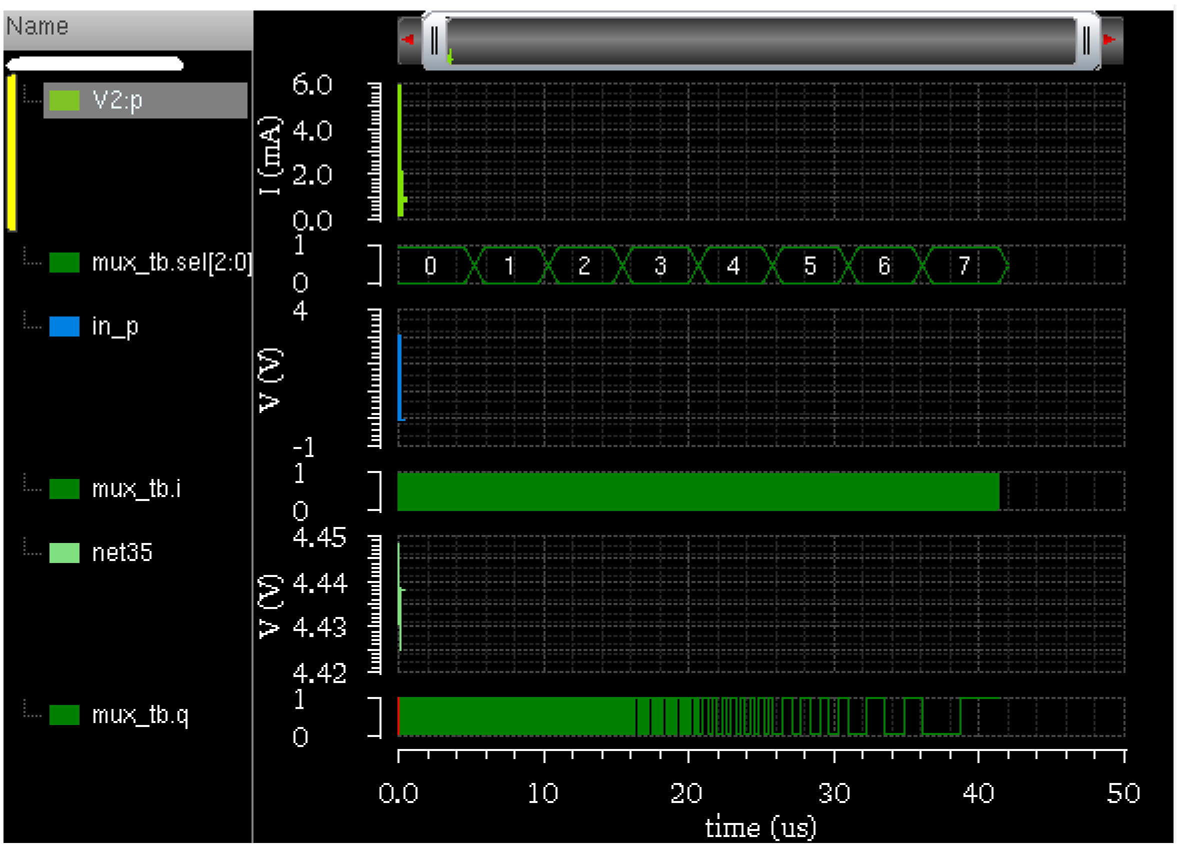
Supporting Mixed Signals

You can use the Virtuoso Visualization and Analysis XL tool to plot analog and digital signals together in one window. The multiple digital signals in a window are displayed in separate strips, while analog signals can be combined into a single strip. The digital signals are displayed in green by default.

To split the analog signals into different strips, see Working with Strips on page 185.

If you want to work with analog and digital signals at the same time, you can plot analog and digital signals in the same window. The signals are plotted in the order they are selected in the Results Browser or ADE.

You can also drag an analog signal to place it in between two digital signals. The analog signal that you drag is displayed in a different strip. You can also impose an analog signal on a digital signal. In this case, the strip height for a digital signal is adjusted according to the height of the analog signal.



You can convert analog signals into digital signals and digital signals into analog signals, if required.

Note: The Verilog and VHDL states are also displayed in specific colors to denote their strength and condition.

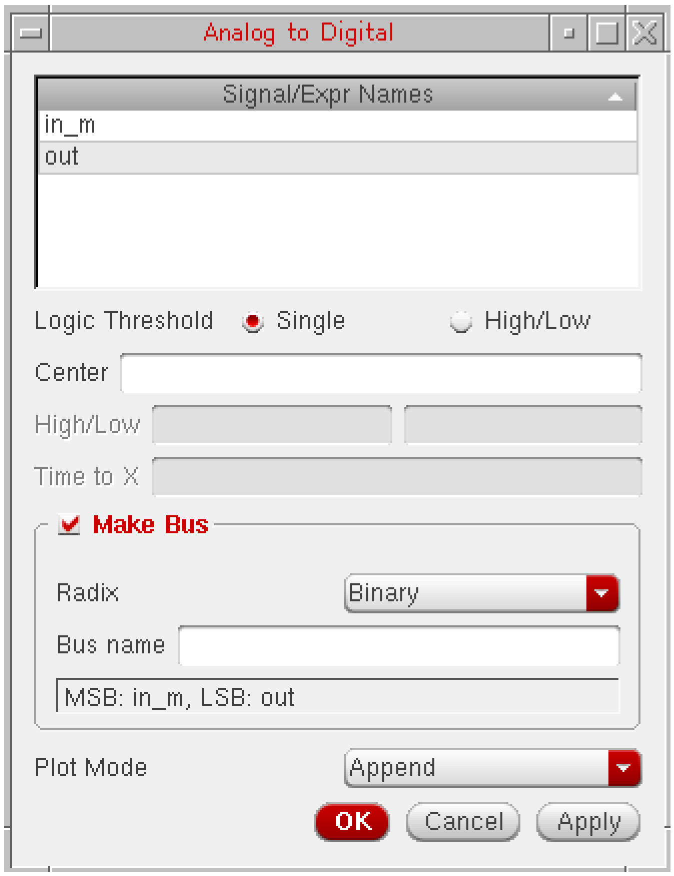
Converting an Analog Signal into a Digital Signal

In the stand-alone SKILL mode, you can create a digital representation of an analog signal.

To convert an analog signal into a digital signal, do the following:

1. In the graph, select a trace and choose *Measurements – Analog To Digital*. You can select more than one analog trace at a time.

   The *Analog to Digital* conversion form appears. This form displays the name of the selected analog signals. The signals are displayed on the basis of their selection order; however, you can rearrange the order either by clicking the column header or by using the drag operation. The assistant has the following fields



2. In the *Logic Threshold* field, select *Single* or *High/Low*.

   ❑ If you select *Single*, you need to specify a *Center* value. Analog values equal to or greater than the specified center value are mapped to a digital value of 1. Analog values less than the center value are mapped to a logical value of 0.

   ❑ If you select *High/Low*, you need to specify a high and a low threshold value. All analog values equal to or greater than the high threshold value are mapped to a digital value of 1. All analog values equal to or less than the low threshold value are mapped to a logical value of 0.
   The value *Time to X* puts a time limit on the interval that the signal may remain

between the high and low threshold values before the signal is assigned a value of *X*.

3. If you want to make a bus of digital signals from the analog signal, select the *Make Bus* check box.
4. Then, select the radix type in the drop-down list and provide a bus name.

   Note: The MSB (topmost signal) and LSB (signal at the bottom) values for the bus are indicated below the *Bus name* field.
5. In the *Plot Mode* drop-down list box, select whether you want to *append* the digital trace to an existing graph, *replace* an existing graph with the digital graph, or add the digital trace to a *new subwindow* or *new window*.
6. Click *OK*.

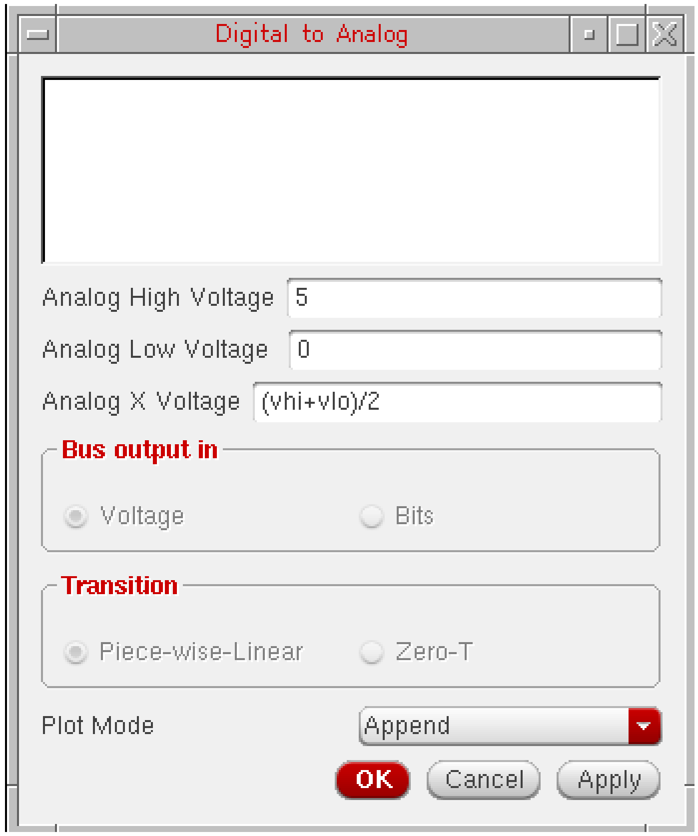
Converting a Digital Signal to an Analog Signal

In the stand-alone SKILL mode, you can also create an analog representation of a digital signal.

To convert a digital signal into an analog signal, do the following:

1. In the window, select a digital trace and choose *Measurements – Digital To Analog*. You can select more than one digital trace at a time.

   The *Digital to Analog* Conversion form appears. This form displays the digital signals you select. The signals are displayed on the basis of their selection order; however, you can rearrange the order either by clicking the column header or by using the drag operation. The assistant has the following fields:



2. In the *Analog High Voltage* field, specify the high analog value to which the digital value 1 (for single bit waveforms) or the maximum bus value is to be converted to. The default value is controlled by the `dToAHiVoltage` variable in the `.cdsenv` file. For more information, see Appendix A, "Virtuoso Visualization and Analysis XL Tool Environment Variables."

3. In the *Analog Low Voltage* field, specify the low analog value to which the digital value 0 (for single bit waveforms) or the minimum bus value is to be converted to. The default value is controlled by the `dToALoVoltage` variable in the `.cdsenv` file. For more information, see Appendix A, "Virtuoso Visualization and Analysis XL Tool Environment Variables."

4. In the *Analog X Voltage* field, specify the value to which state *X* of the digital wave is converted to. The `X` value may be given as a:

   ❑ Number

   ❑ Keyword `vhi`, `vlo`, or `vprev`, where

- `vhi` substitutes the X value with the value in the *Analog Hi Voltage* field.
- `vlo` substitutes the X value with the value in the *Analog Low Voltage* field.
- `vprev` implies that the previous (non-X) state, either `vhi` or `vlo` is used.

- Simple expression, such as `(vhi + vlo)/2`

5. In the *Bus Output in* group box, select *Voltage* to return the selected bus as a single analog signal or select *Bits* to return the selected bus as a wave list, which is one analog wave for each bus bit. The field is available only if you select a bus.
6. In the *Transition* group box, select *Piece-wise-Linear* to join the points in the analog waveform with straight lines or select *Zero-T* for voltage transitions in zero time.

   The *Transition* field is available only if the *Bus Output in* field is selected.
7. In the *Plot Mode* drop-down list box, specify whether you want to *append* the analog trace to an existing graph, *replace* an existing graph with the analog graph, or add the analog trace to a *new subwindow* or *new window*.
8. Click *OK*.

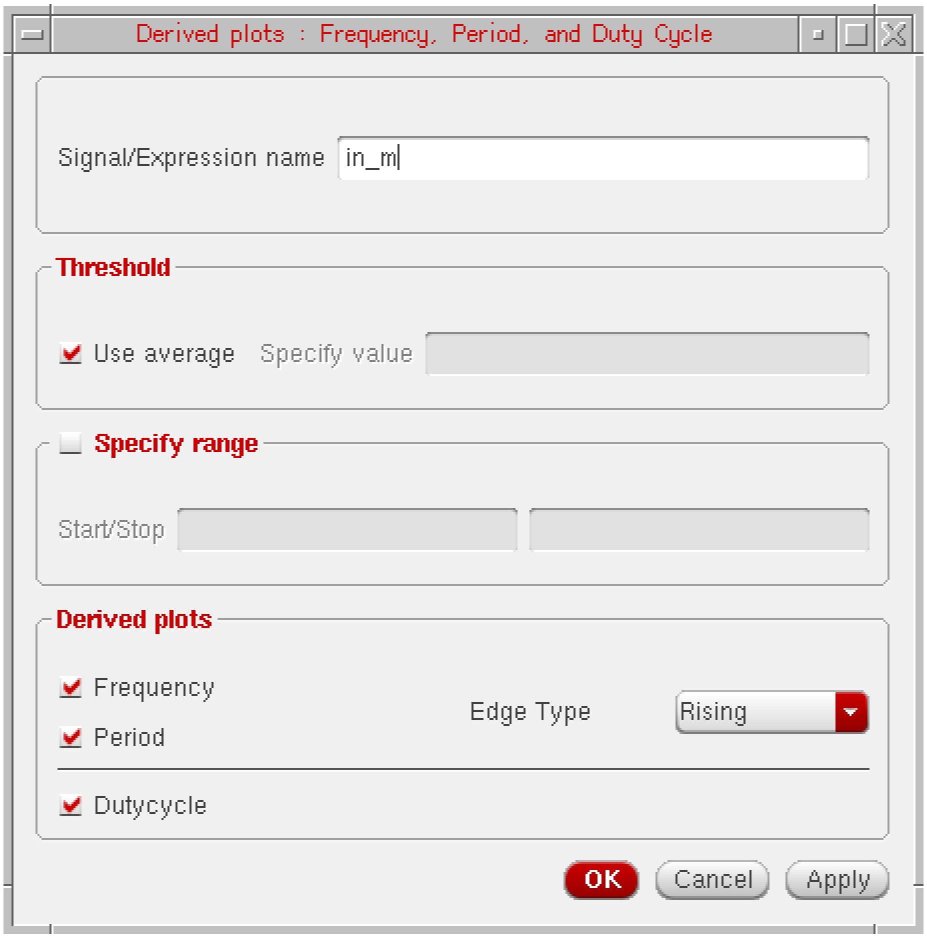
Generating Derived Plots

For transient periodic signals, you can generate the derived plots, such as frequency, duty cycle, and period that can be plotted against time.

Do the following to generate a derived plot:

- Choose *Measurements – Derived Plots*.

The *Derived plots: Frequency, Period, and Duty Cycle* form appears.



- The *Signal/Expression name* field displays the name of the selected trace or expression in the graph. Select a trace for which you want to generate the derived plot, the selected trace is displayed in this field.
- In the *Threshold* field, select the *Use average* check box if you want to use the average threshold value calculated by Virtuoso Visualization and Analysis XL. Otherwise, specify the threshold value in the *Specify value* field. The *Use average* option is selected by default.
- Select the *Specify range* check box if you want to provide the start and stop values, which indicate the range of the derived plots.
- In the *Derived plots* field, select the derived plot you want to generate, such as *Frequency*, *Period*, and *Dutycycle*.
    - Select *Frequency* to plot a waveform representing the frequency of a signal versus time. For more information about *Frequency*, see frequency on page 696.

- ❑ Select *Period* to plot a waveform representing the time period of a signal versus time.
- ❑ Select *Dutycycle* to plot a waveform representing the calculated ratio of the time, for which signal remains high, to the period of the signal. For more information about *Dutycycle*, see dutycycle on page 662.

- Select the *Edge Type* as `Falling` or `Rising` for Frequency and Period.
- Click *Apply* and then click *OK*.

  The selected derived plots are plotted in different strips in the same subwindow.

Limitations

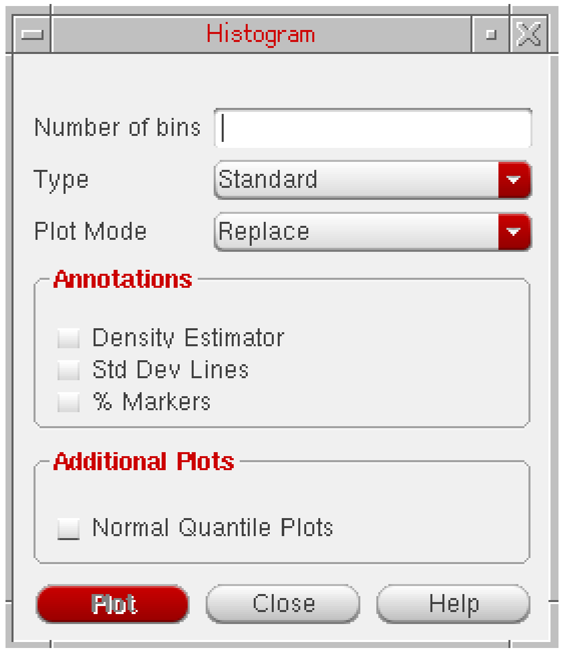
- Derived plots are currently not supported for family data.
- Derived plots are supported only for periodic signals and if you try to generate a derived plot for the non-periodic signals or buses, an error is displayed in CIW.

Plotting Histogram

To generate the histogram plot directly on a graph, do one of the following:

- Select the trace for which you want to create a histogram plot and choose *Measurement – Histogram*.
- Right-click the trace and choose *Measurement – Histogram*.

The *Histogram* form appears.



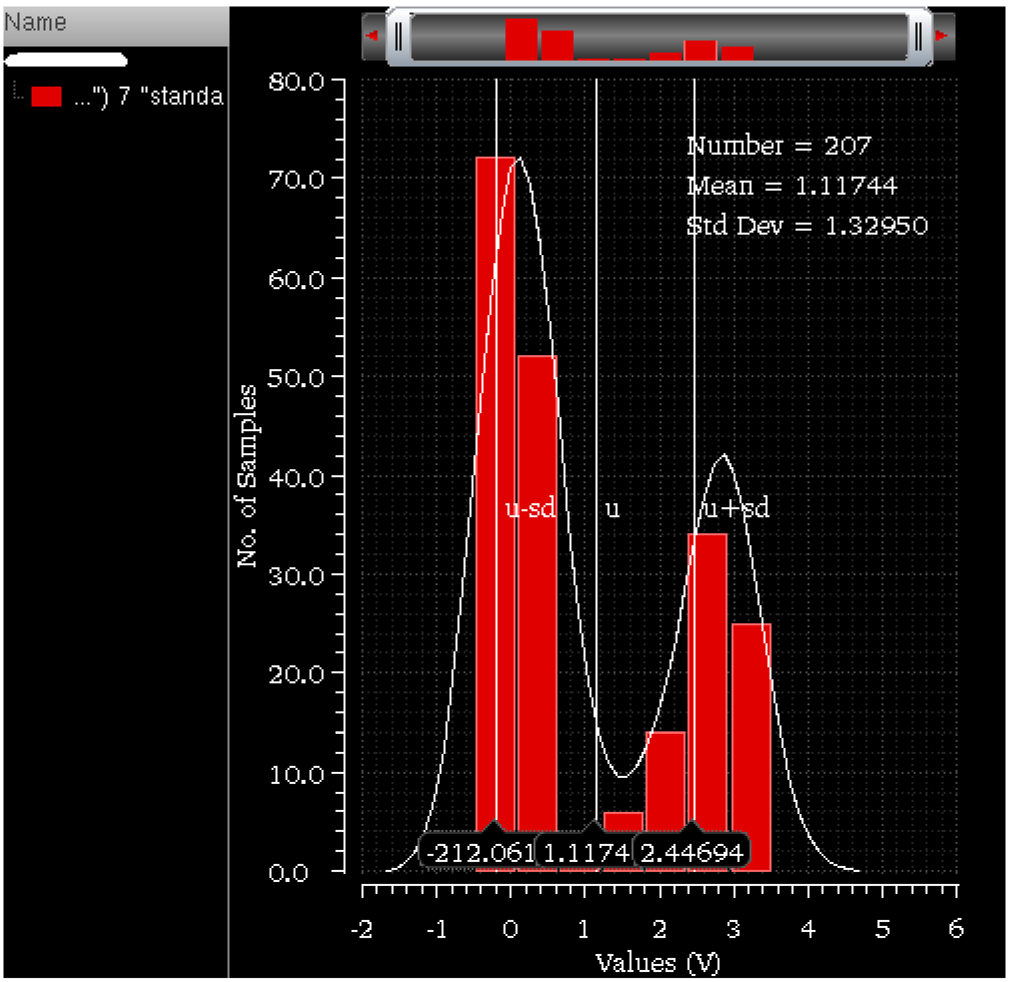
You can specify the following fields in this form:

- *Number of Bins*—Specify the number of bars to plot in histogram. The default value is `10`.
- *Type*—Select the histogram type. You can select one of the following:
    - *Standard*
    - *Cumulative Line*
    - *Cumulative Box*
- *Plot Mode*—Specify the plotting mode. You can specify the following plot modes:
    - *Replace*—This plotting mode is selected by default.
    - *New Subwindow*
    - *New Window*
- In the *Annotations* section, you can select the following annotations that you want to display on the histogram plot:
    - *Density Estimator*—Plots a curve that estimates the distribution concentration.

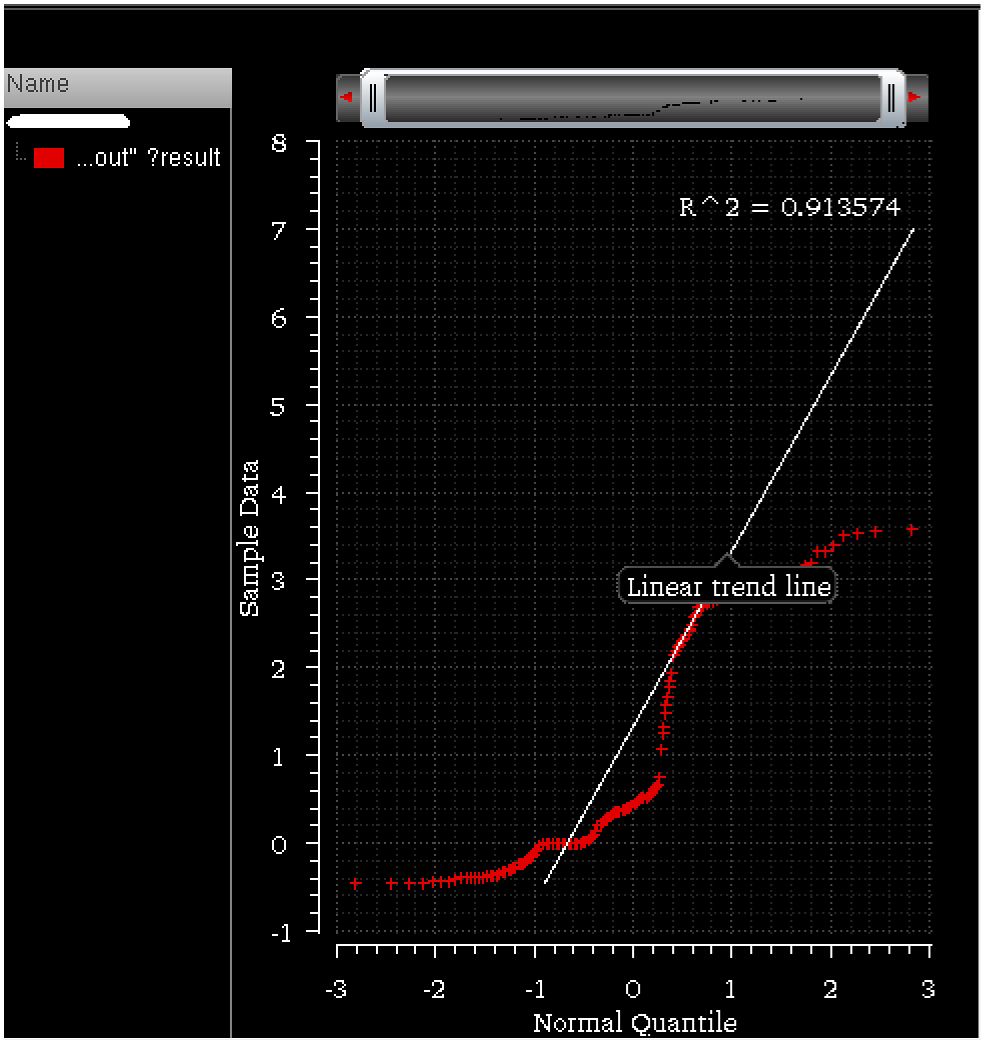
- ❑ *Std Dev Lines*—Shows the standard deviation lines in the graph indicating the mean, (mean - standard deviation), and (mean + standard deviation) values. Note that the standard devitaion is a sample standard deviation.
- ❑ *% Markers*—Displays the markers associated with the histogram trace.

- In the *Additional Plots* section, select the following field:
  - ❑ *Normal Quantile Plots*—Generates the quantile plots.
- Click the *Plot* button.

The histogram plot is generated with the specified properties. The figure below displays a histogram plot with *Density Estimator* and *Std Dev Lines* fields enabled.



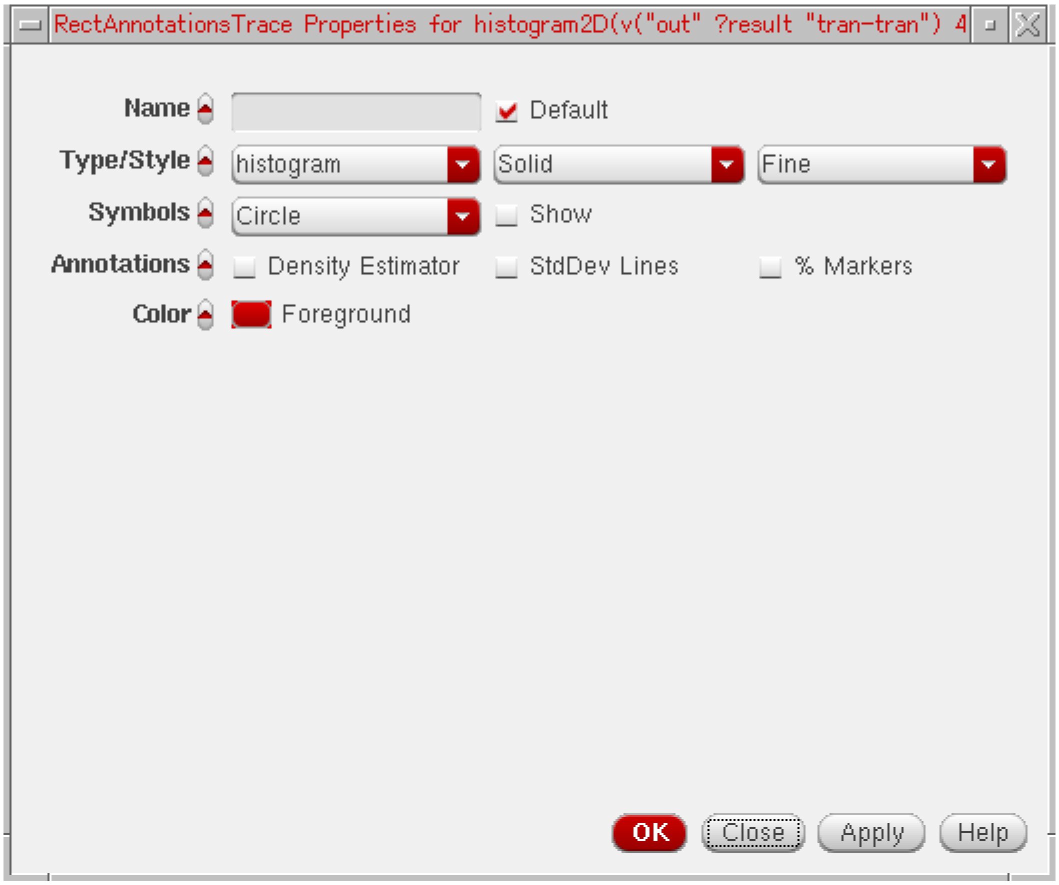
The figure below displays a normal quantile plot:



Changing Histogram Trace Properties

To change the properties of the histogram plot, right-click the histogram trace and choose *Trace Properties*.

The *RectAnnotations Trace Properties for <histogram_trace_name>* form appears.



Specify the following fields in this form:

1. In the *Name* field, type the name for the trace or select the *Default* check box to display the default trace name. When you select the *Default* check box, the *Name* field becomes unavailable. The *Name* and *Default* fields are not available if you select more than one trace.

- In the *Type/Style* fields, do the following:
  - Specify whether you want to represent the trace by a *line*, *points*, *histogram*, *bar*, *spectral,* or *sampleHold*.
  - Specify whether you want the trace style to be *Solid*, *Dashed*, *Dotted*, or *DotDashed,* or *DashDotDot*.

    Note: The trace style option does not work for *Bars* and *Spectrum*.
  - Specify whether you want the trace to be *Fine*, *Medium*, *Thick* or *ExtraThick*.

Note: The trace thickness option does not work for *Bars* and *Spectrum*.

- ❑ In the *Symbols* field, select the *Show* check box to display data points on the trace and select the symbol type from the drop-.down list.
- ❑ In the *Annotations* field, select the following check boxes to display annotaions:
  - ❍ *Density Estimator*—Select to display density estimator curve.
  - ❍ *StdDev Lines*—Select to display standard deviation lines.
  - ❍ *% Markers*—Select to display markers.

- In the *Color* field, select the foreground color for the trace. Alternatively, you can also set the trace color by right-clicking the trace and selecting *Color.*

Working with Buses

A group of digital signals can be converted to a create a bus. You can expand a bus to view its component signals.

If you want to create a bus of analog signals, you need to convert the analog signals to the corresponding digital signals. For information about how to convert an analog signal to the corresponding digital signal, see Converting an Analog Signal into a Digital Signal on page 234. After the conversion is complete, you can create bus from the digital signals.

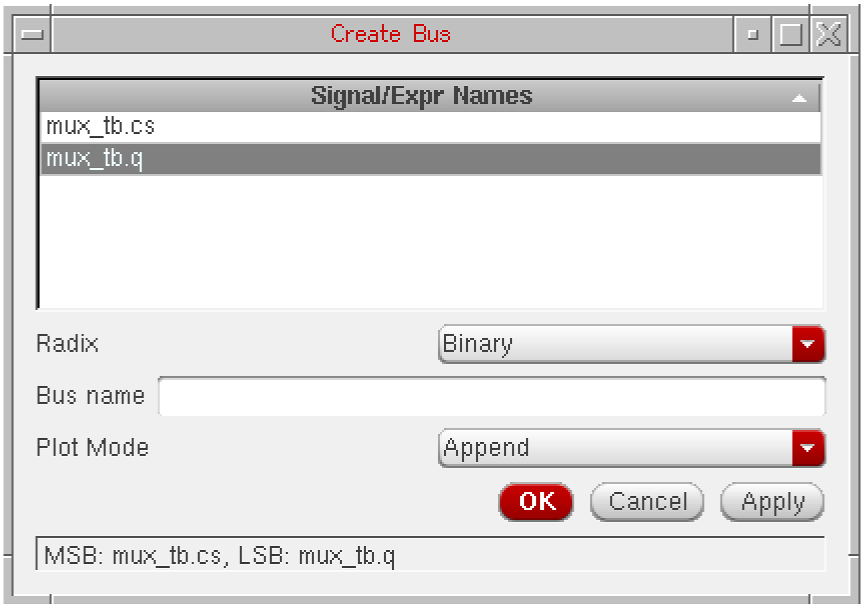
To combine the signal conversion and bus creation processes in a single step, you can select the *Make Bus* check box in the *Analog to Digital* conversion form, and specify a bus name while converting the analog signal to a digital signal.

Creating a Bus

To create a bus, do the following:

1. In a window, select the digital traces that you want to use to create a bus.
2. Choose *Trace – Bus – Create.*

The *Create Bus* form appears.



The *Signal/Expr Names* section in this form displays the selected traces, which you use to create the bus, in the order of significance (top to bottom—from the least significant bit (LSB) to the most significant bit (MSB)). However, you can rearrange the trace order either by clicking the column header or by using the drag operation.

3. Now, in the *Create Bus* form, do the following:

a. Type a name for the bus in the *Bus Name* field.

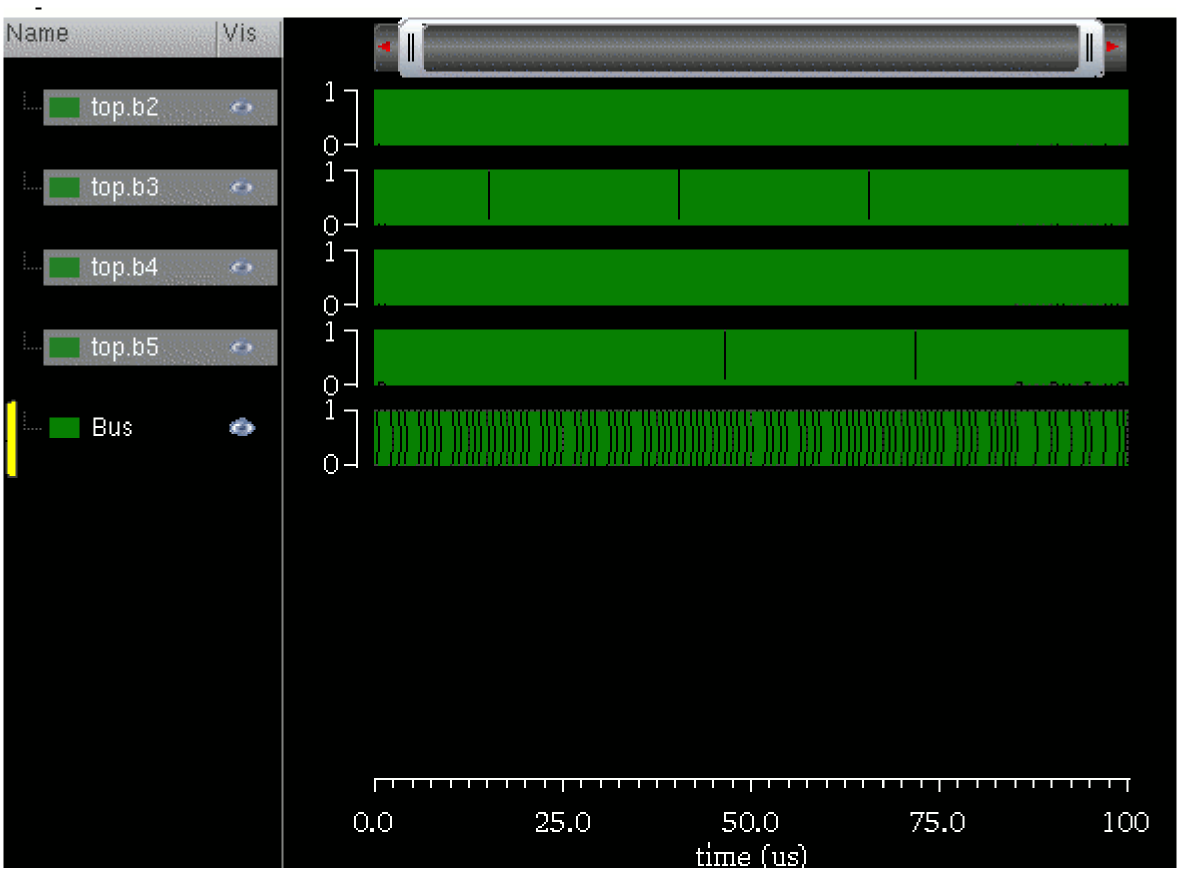
b. Select the *Radix* for the bus, such as Ascii, Binary, Hex. Alternatively, to change the radix type after a bus is created, right-click the bus and choose *Radix*.

c. Specify whether you want the bus to be appended to the digital traces in the graph or to replace the selected traces in the *Plot Mode* field.

d. Click *OK.*

Note: The MSB and LSB values are indicated at the end of the *Create Bus* form.

The bus is created from the selected digital traces.



There are three states in which a signal can exist in the bus—Hi, Lo, and XZ, where X is a transition from Lo to Hi and Z is a transition from Hi to Lo. Following is the color pattern for the traces in a bus that belong to a particular state:

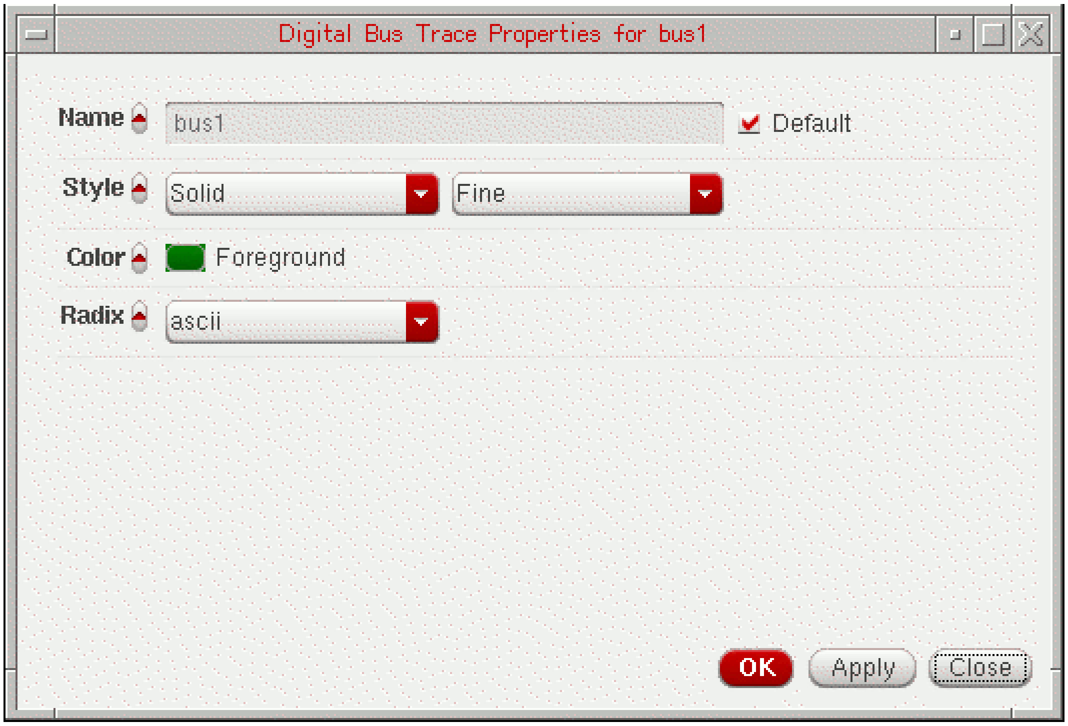
- If the bus is all Z values, it is displayed in orange.
- If the bus is all X values, it is displayed in red color rectangles.
- if the bus includes both X and Z values, it is displayed in yellow colored rectangles.

If you want to create a bus of analog signals, convert them to digital signals by using the Analog to Digital assistant. Then, follow the above mentioned steps to create the bus.

Setting Bus Properties

To set the properties of a digital bus, right-click the bus name and choose *Digital Bus Properties*.

The *Digital Bus Trace Properties for <Bus_Name>* form appears.



This form displays the following information:

- *Name*—Displays the default name of the bus. You can change the bus name if the Default check box is not selected.
- *Style*—Specifies the bus signal properties, such as *Solid* and *Dash*, and specifies the thickness, such as *Fine* and *Medium.*
- *Color*—Specifies the foreground color of the bus.
- *Radix*—Specifies the radix type, such as *hex*, *binary*, and *octal*. You can also set the radix type by right-clicking the bus and choosing

Expanding a Bus

After you add a bus trace to the window, you can expand the bus to display the digital traces contained in the bus into individual strips.

To expand a bus, select the bus you want to expand and do one the following:

- Choose *Trace – Bus – Expand.*
- Right-click the bus or the bus name in trace legend area and choose *Expand*.

The window displays the individual digital traces in the selected bus and also displays the parent bus.

Note: After a bus is expanded, you cannot move, cut, drag, or delete the parent bus and the individual bits in the parent bus.

Collapsing a Bus

To collapse an expanded bus, select the bus you want to expand by selecting the bus name in the trace legend area and do one of the following:

- Choose *Trace – Bus – Collapse.*
- Right-click the bus or the bus name in trace legend area and choose *Expand*.

  The window displays only the bus of digital signals.

Exporting a Bus

To export the data from a bus signal into a CSV file, do the following:

- Select the bus and choose *Trace – Export*.
- Right-click the bus and choose *Send to – Export*.

  The *Export Waveforms* form appears. For more information about this form, see Exporting Signals on page 62.

Sending a Bus Signal to Calculator

To send a bus signal to Calculator, do the following:

- Right-click the bus signal in graph or in the trace legend and choose *Send To – Calculator*.
- Select the bus and choose *Tools – Calculator*.

  The selected bus signal data is displayed in the Calculator.

Consider an example in which you convert an analog signal to digital using the Analog To Digital assistant, then create the bus from the converted digital signal, and finally send the bus signal to Calculator, the following expression is created in the Buffer:

```
awvCreateBus("Test_bus" list(awvAnalog2Digital(v("net10" ?result "tran") nil nil 2
nil "centre") ) "Binary")
```

Here, `awvAnalog2Digital(v("net10" ?result "tran") nil nil 2 nil "centre"` indicates that the `net10` analog signal has been converted to a digital signal.

Sending a Bus Signal to Table

To send the bus signal to Table, do the following:

➡ Right-click the bus signal in graph or in the trace legend and choose *Send To – Table.*

The selected signal is displayed in Table.

Important

If you send the bus that you created after converting an analog signal to digital, the bus data may not appear in Table. To display the bus data in Table, first send the bus signal to Calculator and then send the corresponding Calculator expression to Table.

Working with Markers

A marker attaches a description to a point on the graph. The default label for a marker displays the X and Y coordinates of its intersection with the trace—if it is attached to the trace—or the coordinates of the point location of the marker. You can associate an expression with a marker label. The expression is evaluated when you place the marker on the graph and updated when you choose *File – Reload*.

If you use the replace mode to plot signals obtained from the simulation runs on the same design, the signals are updated with the new data and the expressions are re-evaluated.

Markers are of the following types:

- Point
- Reference Point
- Vertical
- Horizontal
- Circular
- Delta
- AB Delta
- Spec

- Edge

This section includes the following topics:

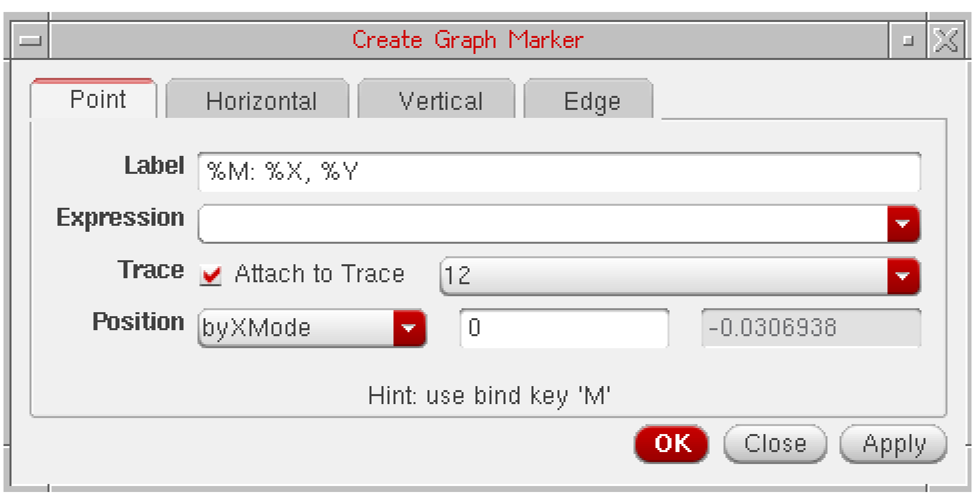
Adding Markers

You can add point, vertical, horizontal, and delta markers to a trace. The circular markers can be added to circular graphs that are obtained from AC analysis, Smith Charts, and polar plots.

To add markers to a trace, do the following:

➡ Select a point on the trace and choose *Marker – Create Marker*.

The *Create Graph Marker* form appears. This form includes various tabs that help to create rectangular markers—point, horizontal, and vertical.



Perform the following steps to create different types of markers:

- To add a point marker, click the *Point* tab. The following fields appear:

- *Label*—Specify a label for the marker. You cannot insert multiline text in the marker labels. Use the format strings listed in the table below to create marker labels. These format strings are evaluated and inserted into the string when you place or edit a marker. As a result, labels can reference properties, such as marker coordinates, trace slope, trace name, and so on, or the result of a scalar expression.

  Note: Each marker label displays the default value set in the `defaultLabel` .cdsenv variable.

  The following table describes the available marker label format strings:

Marker Label	Description
%M	Marker name
%X	X-coordinate
%Y	Y-coordinate
%x	Second X-coordinate for delta markers
%y	Second Y-coordinate for delta markers
%W	Delta value on X-axis (Δx)
%H	Delta value in Y-axis (Δy)
%S	Slope (Δy/Δx)
%N	Name of the trace
%E	Expression
%F	Frequency value

If you do not enter text in the *Label* field, the X- and Y-coordinates of the marker are displayed.

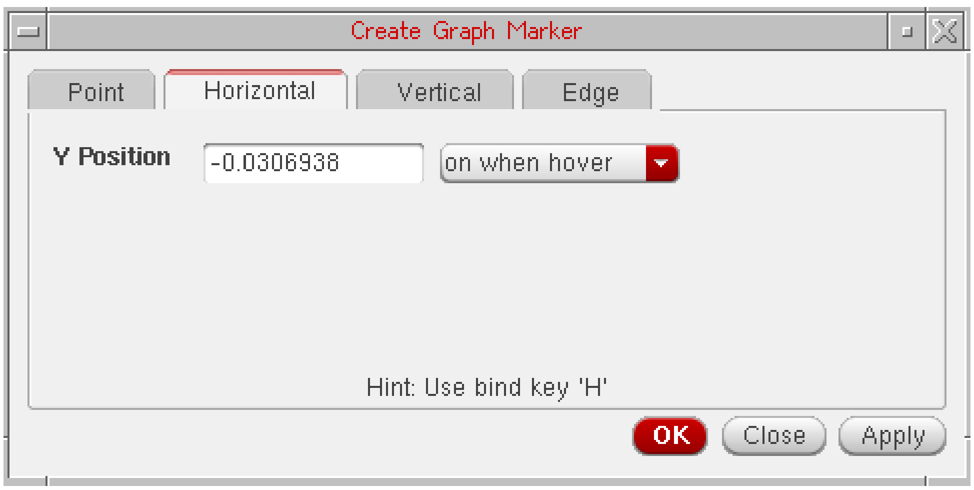
- *Expression*—Select the expression you want to display in the marker label. Click the *Expression* arrow to view the *Buffer Contents* and all defined memories (in SKILL mode) or variables (in MDL mode). If you entered *%E* in the *Label* field, you can choose the variable you want to use in your expression, or you can choose *Calculator Buffer.* If you choose a variable, the expression associated with the selected variable appears in the *Expression* field. If you choose *Calculator Buffer*, the expression in the Calculator Buffer appears in the *Expression* field.

- *Trace*—Select the *Attach to Trace* check box to attach the marker to the closest interpolated point on the trace. When multiple traces are appended to a graph, select the trace to which you want to attach the marker.

  Note: When this check box is selected, the marker color remains same as that of the trace color. When you change the color of the trace, the marker color also changes.

- *Position*—Select the marker position from the drop-down list box. You can specify the marker position `byX Mode`, `byY Mode`, and `byXY Mode`. The default value is `byX Mode`.

■ To add a horizontal marker, click the *Horizontal* tab.



The following field appears:

- *Y Position*—Specify the position on the Y-axis where you want to create the marker. You can select a point on the trace; the Y-axis value of that point is displayed in this field.

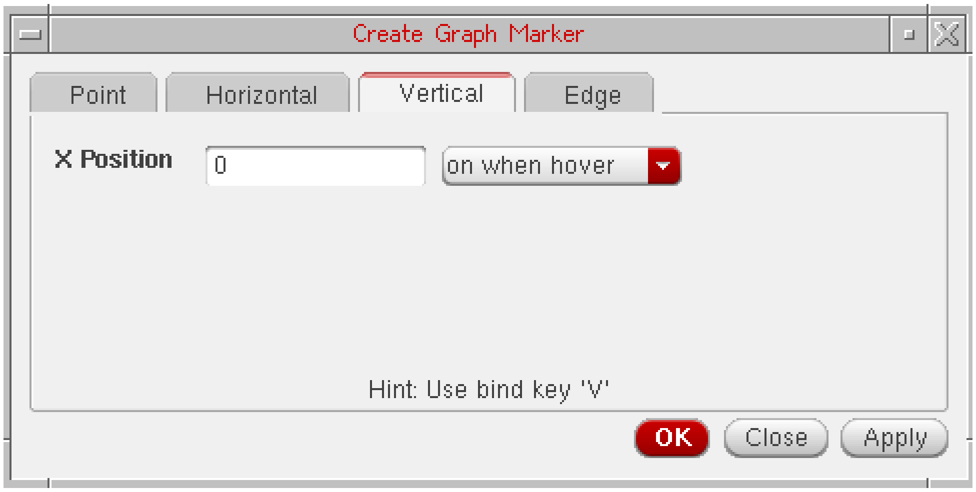
- In the drop-down list box, select the event for which you want to display the horizontal marker.

You can create multiple horizontal markers at the specified locations in a single step by providing a set of Y position values. For example, `10n 20n 30n`.

To optimize the performance of the tool, the horizontal marker displays 10 intercepts at the maximum. If you have multiple traces plotted in a graph, you need to turn off the visibility of other traces to view the intercepts of a given trace. Alternatively, to view all the horizontal marker intercepts, open the horizontal marker table. For information about horizontal marker table, see Horiz Marker Table on page 154.

The horizontal marker appears in a dash line style. To change the line style, right-click the marker and choose *Horizontal Marker Properties*. For more information see, Setting Marker Properties on page 258.

- To add a vertical marker, click the *Vertical* tab.



The following field appears:

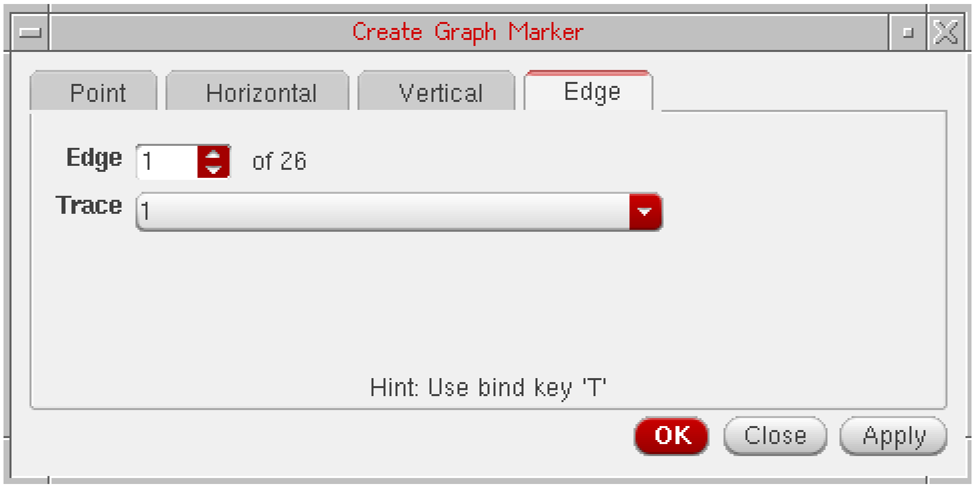
- *X Position*—Specify the position on the X-axis where you want to create the marker. You can select a point on the trace; the X-axis value of that point is displayed in this field.
- In the drop-down list box, select the event for which you want to display the vertical marker.

You can create multiple vertical markers at the specified locations in a single step by providing a set of X Position values. For example, `10n 20n 30n`.

The vertical marker appears in a dash line style. To change the line style, right-click the marker and choose *Vertical Marker Properties*. For more information see, Setting Marker Properties on page 258.

The horizontal and vertical markers show all intercepts on all traces across all strips in a subwindow. To display the horizontal and vertical marker intercepts, right-click a horizontal or vertical marker and choose one of the following options in the *Intercepts* menu – *Off*, *On When Hover*, or *On*.

■ To create an edge marker, click the *Edge* tab.



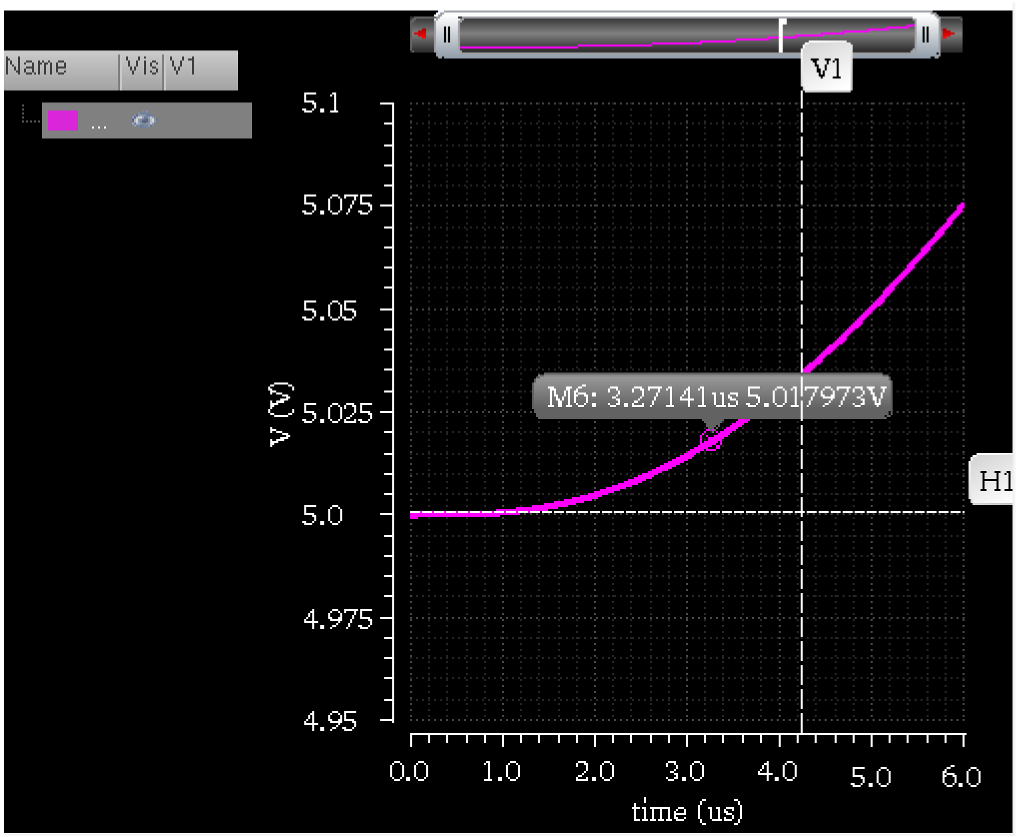
The following fields appear:

❑ *Edge*—Specify the edge number where you want to create an edge marker. The total number of edges are displayed next to this field.

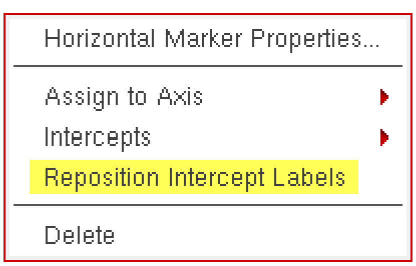
❑ *Trace*—Select a trace from the drop-down on which you want to place the edge marker. This drop-down lists the name of all traces that are plotted in the selected window.

❑ For more information about how to create edge markers, see Creating an Edge Marker on page 279

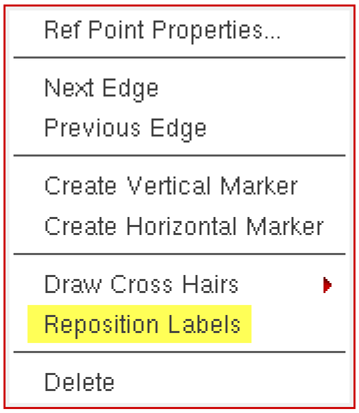
The following figure shows a point, vertical, and horizontal marker.



You can move the labels attached to the horizontal and vertical markers. To bring the marker labels back to their original position, right-click the horizontal, or vertical markers and choose *Reposition Intercept Labels*.



After moving labels attached to the point, reference points (A/B marker), delta, and circle markers, you can bring the marker labels back to their original position by right-clicking the point marker and choosing *Reposition Labels.*



Adding Vertical and Horizontal Markers on Point Marker Location

You can add a vertical or a horizontal marker at the same location where a point marker is placed.

To add a vertical marker, right-click the point marker and choose *Create Vertical Marker*.

To add a horizontal marker, right-click the point marker and choose *Create Horizontal Marker*.

Adding Markers with Bindkeys

To add a marker with the help of bindkeys, do the following:

1. Click a point on the graph where you want to place a marker.
2. Press one of the following keys:
   - `M`—Adds a point marker
   - `H`—Adds a horizontal marker
   - `V`—Adds a vertical marker

   A marker is placed on the trace based on the bindkey you use.

Adding Markers with Marker Toolbox

To add a vertical, horizontal, point, or reference point (ARefPoint or BRefPoint) marker with the help of the Marker Toolbox, do the following:

➡ Choose *Window – Assistants – Marker Toolbox*.

The *Marker Toolbox* assistant appears to the left of the window. It includes tools that you can use to add the required type of markers. To create a marker, drag the desired marker to the specific location where it needs to be placed.

Setting Marker Properties

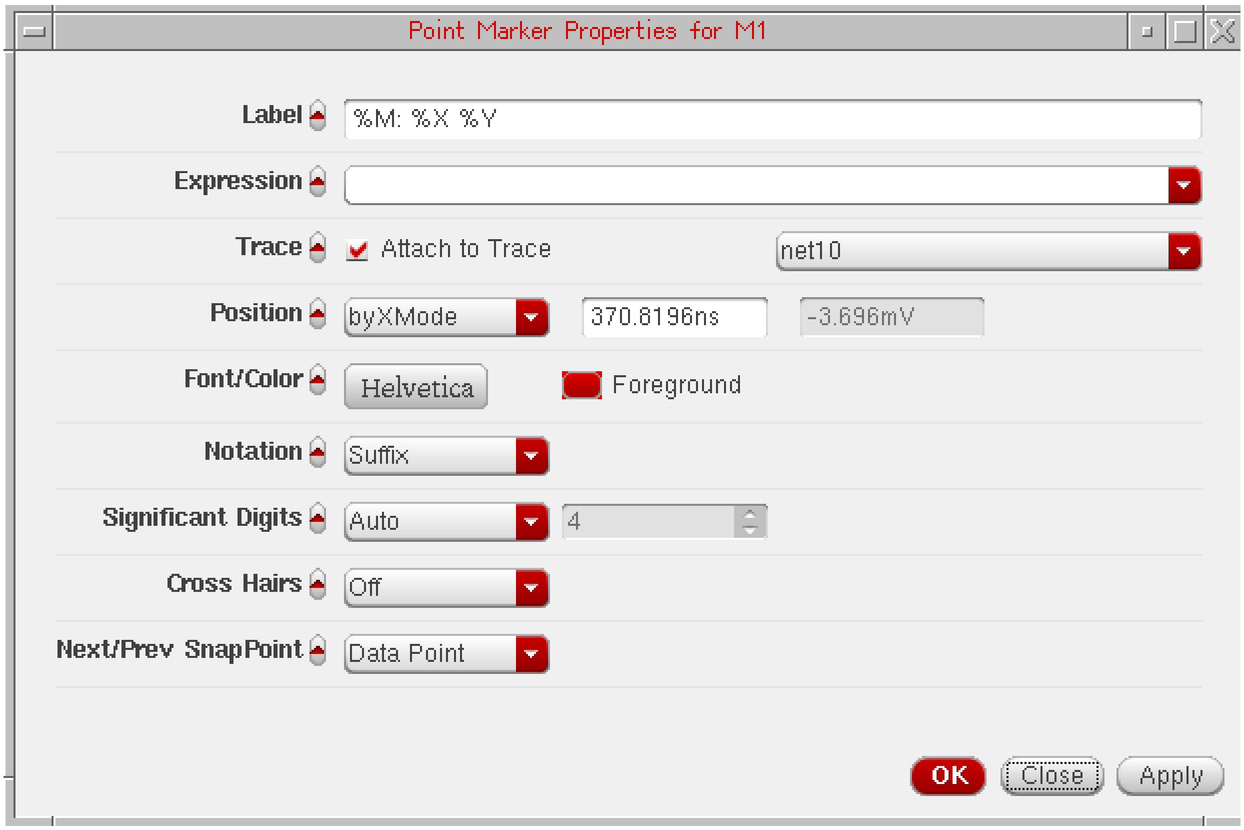
To view or change the properties of markers, do one of the following.

- Double-click the marker.
- Right-click the marker and choose *Marker Properties*.

  The *Marker Attribute Properties* form appears. The properties form displays different fields based on the type of marker selected.

Setting Properties for Point Markers

The following fields are displayed for setting the properties of a point marker:



- *Label*—Specify the label for the marker. For more information about marker labels, see Adding Markers on page 251.
- *Expression*—Specify the expression associated with the marker. For more information about expressions, see Adding Markers on page 251.
- *Trace*—Specify the name of the trace to which you want to attach the marker.
- *Position*—Specify X-axis and Y-axis position for point marker. By default, the position is `byXMode`.
- *Font/Color*—Specify the font name, style, and size for the label.
- *Notation*—Specify the notation to be displayed on labels. The available options are—`Suffix`, `Engineering`, and `Scientific`. Default value: `Suffix`.
- *Significant Digits*—Specify the number of significant digits if you select `manual` from the drop-down. The other option is `auto`.

- *Cross Hairs*—Specify whether you want to display the cross hairs for point marker or delta marker. If you select *Dynamic* from the *Cross Hairs* drop-down list box and click the reference point marker, the cross hairs are displayed.

  Alternatively, to turn on the cross hairs for a delta marker, right-click one of the reference point markers and choose *Draw Cross Hairs – On/Off/Dynamic*.

- *Next/Prev SnapPoint*—Specify the criterion based on which the selected reference point marker should be snapped.

  The available options are—`Local Maxima`, `Local Minima`, `Local Max or Min`, `Specific Y Value`, `Specific X Value`, `Data Point`, `Global Maxima`, and `Global Minima`.

  The `Specific X Value` option is not available for horizontal markers.

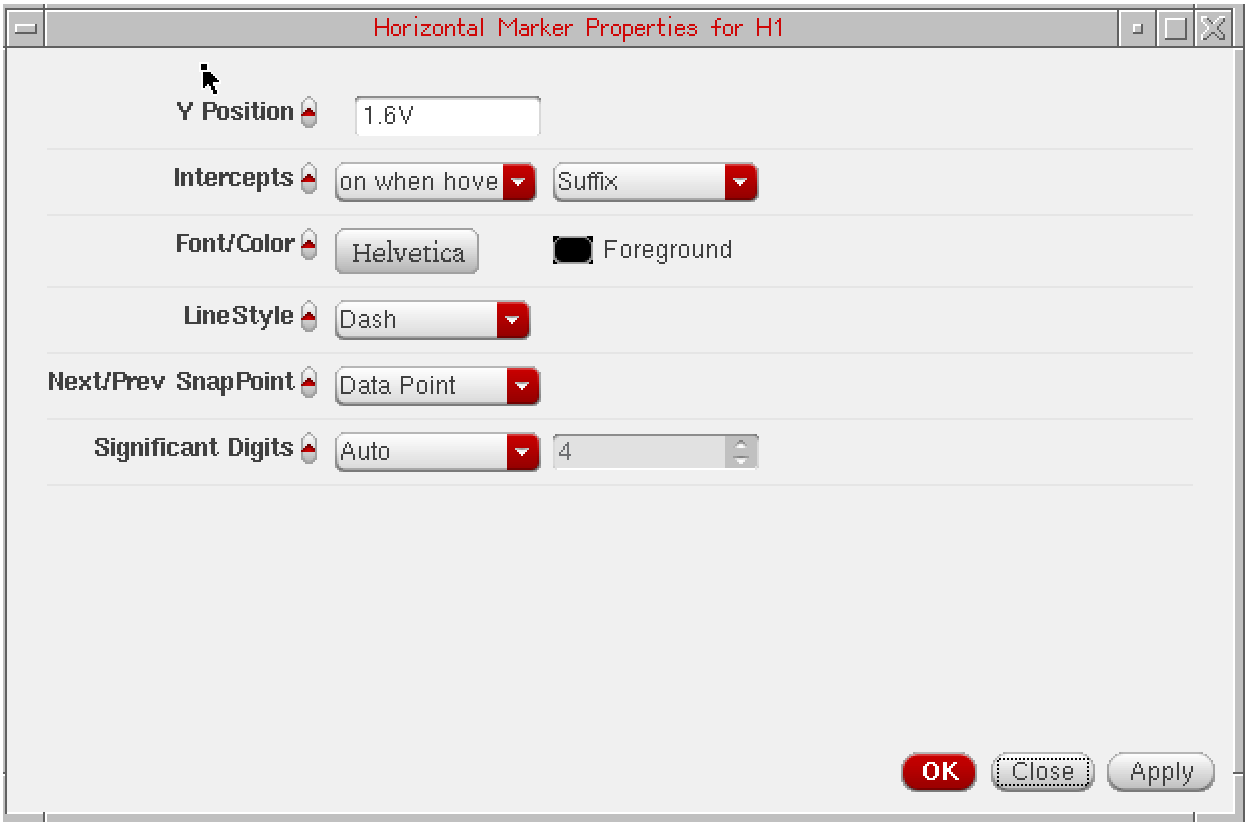
  Default value: `Data Point`.

Setting Properties for Horizontal and Vertical Markers

To change the properties for horizontal or vertical marker, do one of the following:

- Select the marker and choose *Marker – Properties*.
- Right-click the marker and choose *Vertical Marker Properties* for vertical marker and *Horizontal Marker Properties* for horizontal marker.
- Right-click the marker column in the Vertical or Horizontal Marker Table and choose *Vertical Marker Properties* for vertical marker and *Horizontal Marker Properties* for horizontal marker. For more information about marker tables, see Displaying Intercept Data for Markers in Marker Tables on page 265.

The following fields are displayed for setting the properties of a horizontal or vertical marker:



- *Y Position*—Specify the Y-coordinates where you want to place the horizontal marker. You can specify multiple Y-coordinates to place multiple horizontal markers at a time.
- *X Position*—Specify the X-coordinates of the point where you want to place the vertical marker. You can specify multiple X-coordinates to place multiple vertical markers at a time.
- *Intercepts*—Specify the event for the marker intercept label display. Alternatively, right-click the marker and choose *Intercepts – Off/On When Hover/On*.
- *Font/Color*—Specify the font name, style, and size for the label.
- *Foreground*—Select the foreground color for the symbol, the arrow and the label.
- *Line Style*—Specify the marker line style, such as dotted.
- *Significant Digits*—Specify the number of significant digits if you select `manual` from the drop-down. The other option is `auto`.

- *Next/Prev SnapPoint*—Specify the criterion based on which the selected reference point marker should be snapped.

  The available options are—`Local Maxima`, `Local Minima`, `Local Max or Min`, `Specific Y Value`, `Specific X Value`, `Data Point`, `Global Maxima`, and `Global Minima`.

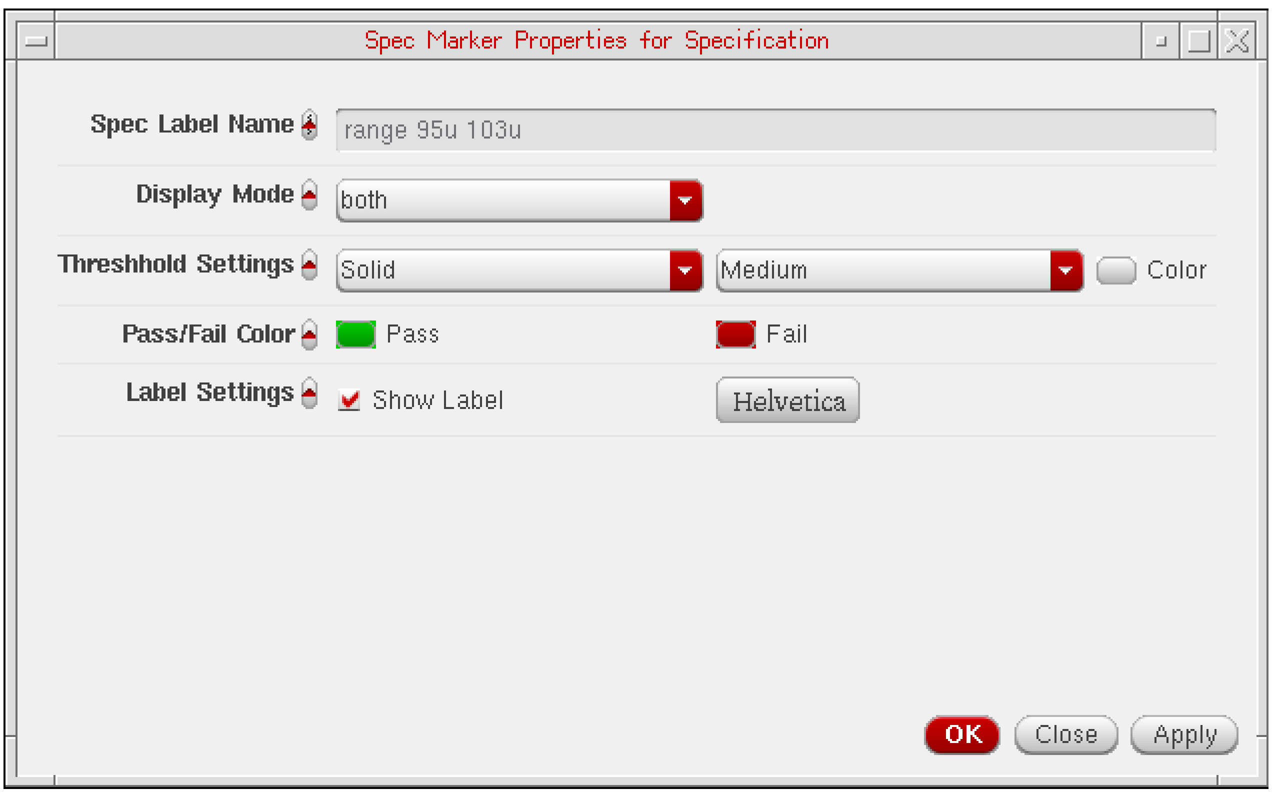
  The `Specific X Value` option is not available for horizontal markers.

  Default value: `Data Point`.

Setting Properties for Spec Markers

To view or change the properties of spec markers, right click the corresponding trace and choose *Spec Properties*.

The *Spec Marker Properties for Specification* form is displayed, as shown in the following figure:



In this form, you can edit the following properties:

- *Spec Label Name*—Shows the spec name that is displayed on the label.

- *Display Mode*—Specify the spec marker pass/fail display style.
- *Threshold Settings*—Specify the line style, line thickness, and line color for the spec marker.
- *Pass/Fail Color*—Specify the colors to shade pass or fail regions.
- *Label Settings*—Specify the font and display settings for spec marker labels.

If a graph contains multiple traces, spec marker properties cannot be changed for an individual trace. Changes in spec marker properties are applicable for all the traces of a graph.

Snapping Markers

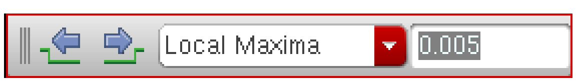
You can snap markers to analog and digital traces. In analog traces, you can also set the criteria based on which you want to snap markers, where as digital markers can be snapped only to the edge transitions, low to high and high to low. If the marker extends beyond the display area, the marker is panned automatically.

For digital traces or buses, the vertical, delta, and point markers can be snapped and for analog traces, the horizontal, vertical, delta, and point markers can be snapped based on the snapping criterion.

To set the snapping criteria to snap the markers to analog traces in the window, do the following:

1. Select a maker and choose a snapping criterion from the drop-down list box displayed on the snap toolbar based on which you want to snap the marker. For example, local maxima, local minima, and so on. By default, `Data Point` is selected in the drop-down list box.

   When you add a vertical or horizontal marker on a trace, the *Value* field displays the X-axis location of the selected marker. This field is updated automatically if you move the marker.

2. Specify a value for the selected snap criterion by which you want to snap the marker.
3. Then, to snap the marker to the next and previous snap points, do one of the following:
   - On the Snap toolbar, click the *Next Edge* and *Previous Edge* buttons.

- Right-click the selected marker and choose *Next Edge* and *Previous Edge* respectively. Note that these options are not available in Horizontal marker context menu.
- Press the `N` or `P` bindkeys to move to the next or previous edges, respectively.

The selected marker is snapped based on the snap criterion you selected. For example, if you select *Local Maxima* as the snap criterion, the marker is shifted to the maxima value (peak) local to the curve when you click the *Next Edge* button.

You can select any one of the following snap criterion:

- **Local Maxima**—Defines the transition point when there is a change in the slope from the rising to falling edge starting from the marker's current position. The local maxima is calculated as the change in slope from rising to falling edge starting from the current marker position and the transition point is known as the local maxima.
- **Local Minima**—Defines the transition point when there is a change in the slope from the falling to rising edge starting from the marker's current position.

  Note: At any point, the double derivative of a waveform should be zero to find the local maxima or the local minima. **Local Max or Min**—Defines that the snap point can be either local maxima or local minima.
- **Specific Y Value**—Defines the snap point of the marker to a specific Y-axis value.
- **Specific X Value**—Defines the snap point of the marker to a specific X-axis value.

  Note: This snap criterion works for all types of markers except the horizontal markers.
- **Data Point**—Defines that a specific data point on the curve should be considered as the snap point. This is the default snap criterion.
- **Global Maxima**—This snap criterion is similar to local maxima. The only difference is that it applies the snap settings to the global maxima or positive peak.
- **Global Minima**—This snap criterion is similar to local minima. The only difference is that it applies the snap settings to the global minimum value.

If you want to use the same snap criterion to snap a marker to more than one analog trace, select the maker and then hold down the `Ctrl` key and click the analog traces. Next, click the *Next Edge* or *Previous Edge* button to snap the marker to the snap points on the selected traces.

To snap digital markers, select the digital marker and click the *Next Edge* and *Previous Edge* buttons on the Snap toolbar.

Customizing Markers

This section includes the following topics:

- Displaying Intercept Data for Markers in Marker Tables on page 265
- Deleting a Marker on page 269
- Editing a Marker on page 270
- Moving a Marker on page 270

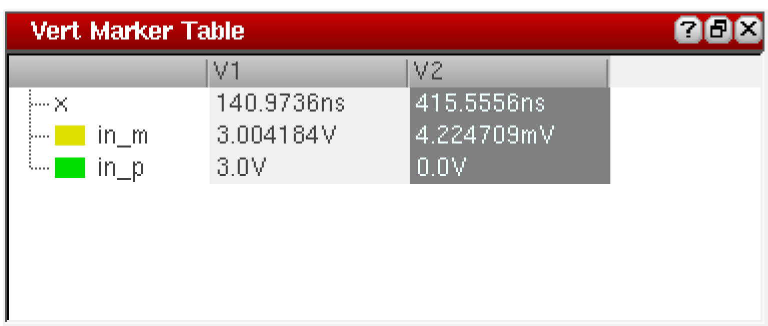
Displaying Intercept Data for Markers in Marker Tables

Vertical Marker Table

To display the vertical marker intercepts, do the following:

➡ Choose *Window – Assistants – Vertical Marker Table*.

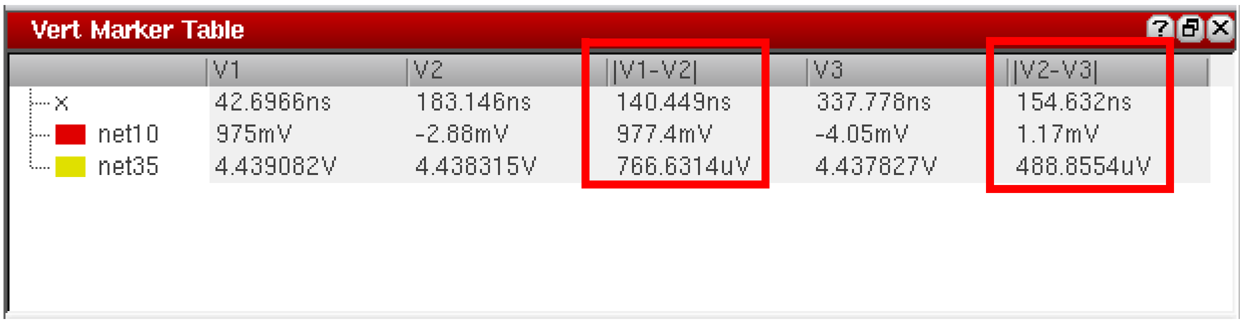
The *Vert Marker Table* assistant appears at the bottom of the window, displaying all vertical marker intercepts for each trace. When you add a vertical marker on a trace, the vertical marker intercepts for all the traces are displayed in the marker table.



In the vertical marker table, rows display the trace names and columns display the intercept points of each vertical marker.

Note: The active vertical marker intercepts for each trace are also displayed in the trace legend area.

If you create a delta marker between two or more vertical markers, the vertical marker table includes an additional column to display the vertical marker delta values on traces (as shown in the figure below).



Vert Marker Table

	V1	V2	|V1-V2|	V3	|V2-V3|
x	42.6966ns	183.146ns	140.449ns	337.778ns	154.632ns
net10	975mV	-2.68mV	977.4mV	-4.05mV	1.17mV
net35	4.439082V	4.438315V	766.6314uV	4.437827V	488.8554uV

You can create a delta marker between two vertical markers by using one of the following methods:

- Create a vertical marker by using the bindkey `V`. Keeping the marker selected, place the mouse pointer at a point on the trace where you want to create the other vertical marker. Press the bindkey `D`. A delta marker is created between the two vertical markers.
- Select all vertical markers by using the `Ctrl` key and choose *Marker – Create Delta Marker*.
- Select all vertical markers by using the `Ctrl` key and press bindkey `Shift+D`.

Note: The delta value displayed in the vertical marker table is always an absolute value.

If you do not want to display delta values in the vertical marker table, right-click the delta line joining two vertical markers and choose *Diff Visible*.

You use the horizontal marker table to view the trace intercepts for all the horizontal markers in a table.

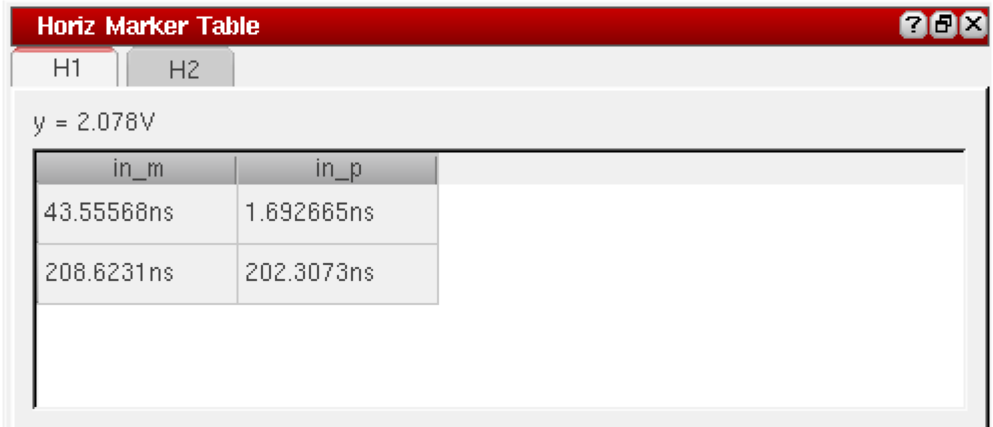
Horizontal Marker Table

To display the horizontal marker intercepts in a table, do the following:

➡ Choose *Window – Assistants – Horizontal Marker Table*.

The *Horiz Marker Table* form appears at the bottom of the window and displays the intercepts where horizontal marker intersects traces in the graph.

The horizontal marker table includes a separate tab for each horizontal marker. The marker table for the active horizontal marker is displayed. If you change the marker name, the tab name in the table is updated automatically.



Note: When you zoom-in a graph, the horizontal marker table lists only those intercepts that are visible in the zoomed-in portion of the graph. The horizontal marker table is updated only when you move the marker.

If the graph includes multiple Y-axes, do the following to change the axis of the horizontal marker:

➡ Right-click the marker and choose *Assign to Axis – axis-name*.

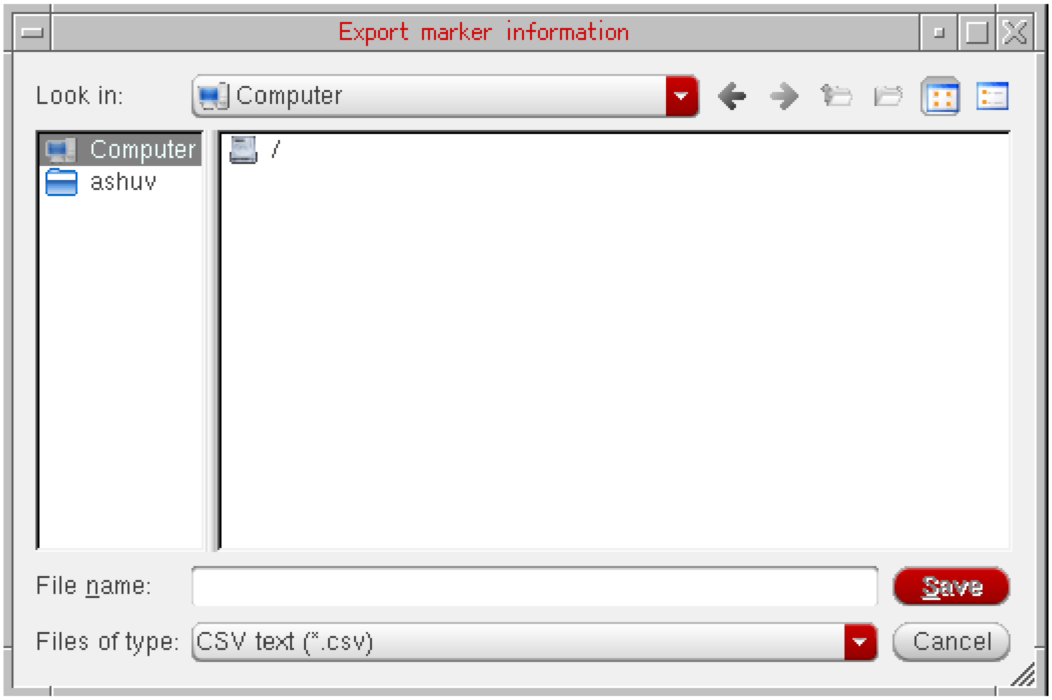
Horizontal marker now shows intercepts for the traces that are attached to the axis you select. The marker table is also updated with the new intercepts.

Exporting Markers

To export the vertical marker intercept data in a CSV file, do the following:

➡ Choose *Marker – Export – Vertical Marker*.

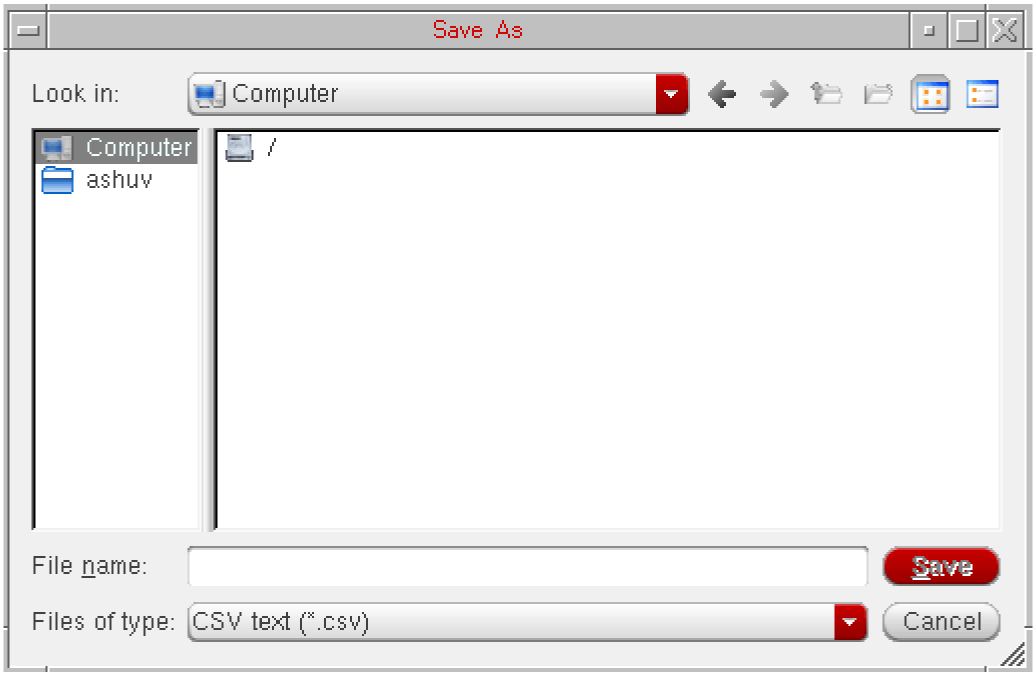
The *Export marker information* form appears. In this form, specify the name and location of the CSV file in which you want to save the vertical marker information and then click *Save*.



To export the horizontal marker intercept data in a CSV file, do the following:

➡ Choose *Marker – Export – Horizontal Marker.*

The *Save As* form appears. In this form, specify the name and location of the CSV file in which you want to save the horizontal marker information and then click *Save*.



Deleting a Marker

To delete a marker, do one of the following:

- Select a marker and choose *Edit – Delete*, or press the `Delete` key.
- Right-click a marker and choose *Delete*.

To delete all markers on a trace, select a marker and do one of the following:

- Choose *Edit – Delete All*
- Choose *Trace – Delete All*
- Press `Ctrl+E`

To delete a AB delta marker, you can right-click any of the two point markers or the delta marker line and choose *Delete*. The A and B markers in the delta marker are deleted.

Editing a Marker

The default marker attributes are controlled by the values assigned to variables in the `.cdsenv` file. For more information, see Appendix A, "Virtuoso Visualization and Analysis XL Tool Environment Variables."

To edit a marker, double-click the marker. The *Marker Properties* form appears. Edit the required fields in this form. For more information about the fields, see Setting AB Delta Marker Reference Point Properties on page 274.

Moving a Marker

To move a point marker, drag the point marker anywhere on the trace.

To move a vertical marker, place the pointer on the vertical marker. When the pointer becomes a bidirectional arrow, drag the pointer along the X-axis to move the marker. Similarly, drag a horizontal marker along the Y-axis to move the marker.

Moving a Delta Marker

To move a delta marker, you can set the snap criterion on a point marker in the delta marker. You can then use the *Next Edge* and *Previous Edge* buttons to move the selected point marker in the delta marker.

Working with Delta Markers

Delta markers are used to mark the difference between two points in a graph. A delta marker joins two point markers in the same or different traces. To place a delta marker you must first place a point marker or select one. Delta markers can be moved or deleted independent of their point markers.

You can move either end of a delta marker; X and Y coordinates are updated accordingly. You can use delta markers to measure delays or use them with the min and max functions to measure peak-to-peak values.

This section includes the following topics:

- Editing Delta Marker Properties on page 276

Adding Delta Markers

You can create delta markers between two or more point, vertical, and horizontal markers on one or more traces. You can also create delta markers between two different marker types. For example, you can create delta marker between point markers and vertical markers.

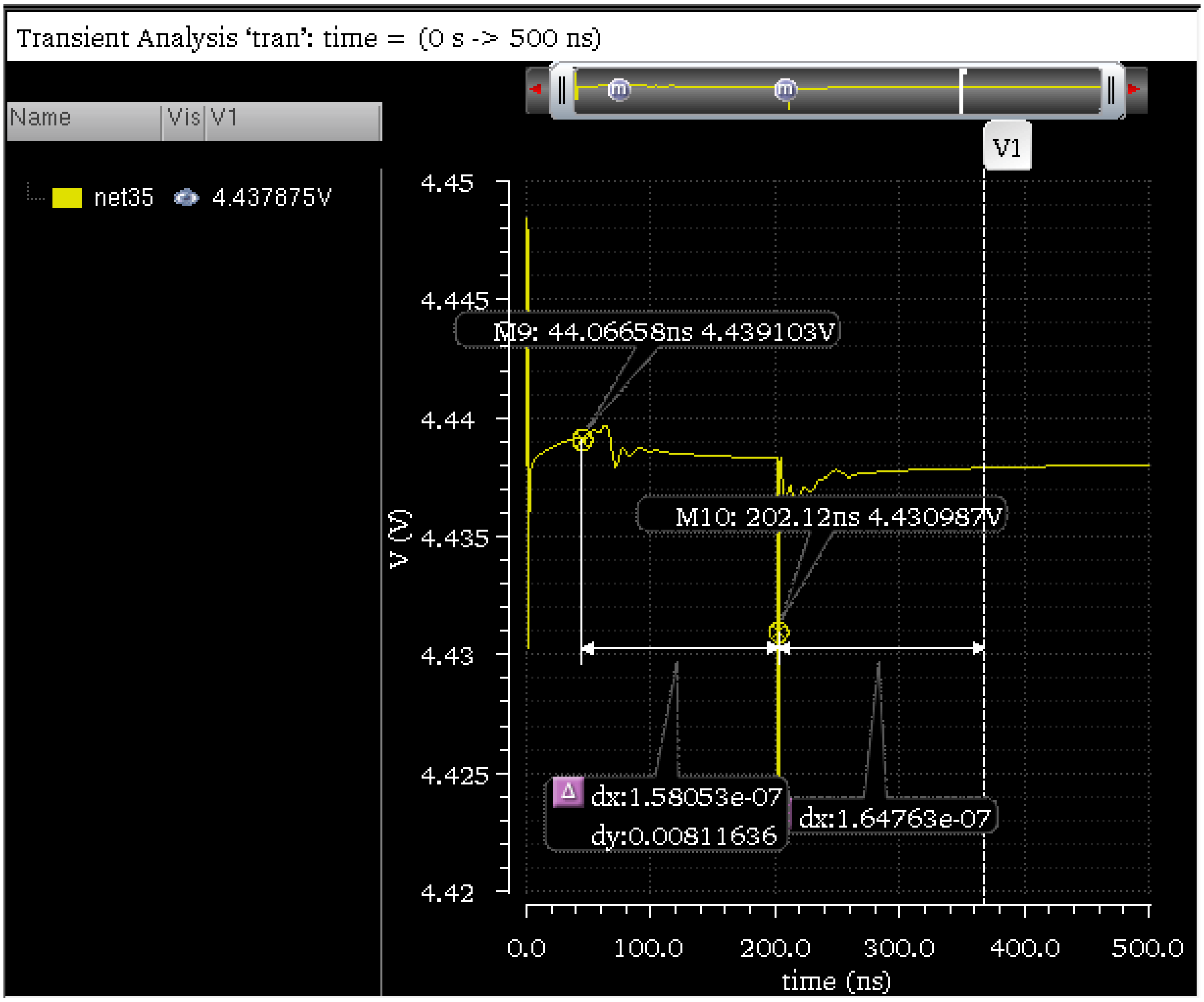
Following are the two methods that you can use to create multiple delta markers on a trace:

Method 1:

1. Add two or more point markers. To know how to create a point marker, see Adding a Point Marker on page 290.
2. Select all the point markers by holding down the `Ctrl` key and do one of the following:
   - ❑ Press the bindkey `Shift+D`.
   - ❑ Choose *Marker – Create Delta Marker*.

   The delta markers are created between all the selected point markers. The method can be applied to create delta markers between any combinations of point, vertical, and horizontal markers.

Note: You can add point, vertical, horizontal, delta, and AB markers in the eye diagram and spectrum plots. However, when you add a vertical or a horizontal marker in the eye diagram, the intercepts are not displayed in the plot.



Method 2:

1. Create a point marker. To know how to create a point marker, see Adding a Point Marker on page 290.

   The point marker you created remains selected.

2. Place the mouse pointer on the trace where you want to create the second marker. Note that you can create delta markers on multiple traces.

3. Press the bindkey `D`.

A new point marker is created at the same point where you placed the mouse pointer and a delta between this new marker and the previously created point marker is also created. This new marker is of the same type as the marker type of the previously selected marker. For example, if you created a point marker in step1, the new maker created after step2 is also a point marker.

You can repeat this method to create delta markers between multiple point markers. The last marker that you create remains selected.

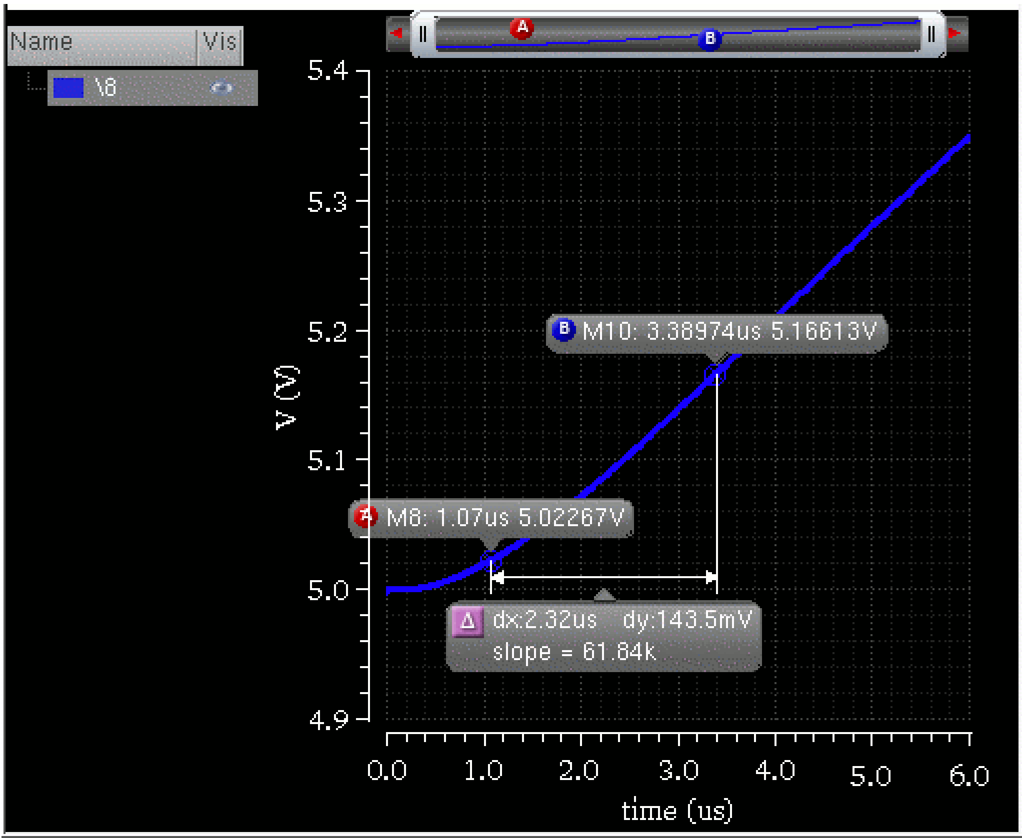
Repeat steps 2-3 to create delta marker between two or more vertical or horizontal markers.

Adding AB Marker

AB marker is a delta marker of XY type and displays the `dx`, `dy`, and the `slope` values. Do the following to add an AB marker to the trace with the help of bindkeys:

1. Move the mouse pointer to a location on the trace where you want to create an AB marker.
2. Press `A`.
3. Move the mouse pointer to another point on the trace to specify the second location for the delta marker.
4. Press `B`.

A delta marker of XY type appears on the graph. If one of the traces is a digital trace, the delta marker label displays only the `dx` value.



You can add multiple AB markers by repeating these steps.

Note: To convert an AB delta marker into a delta marker, right-click the delta marker line and choose *Convert A/B Marker to Delta*.

Displaying Marker Labels in Delta Markers

To show or hide the marker labels for the point markers in a delta marker, do one of the following:

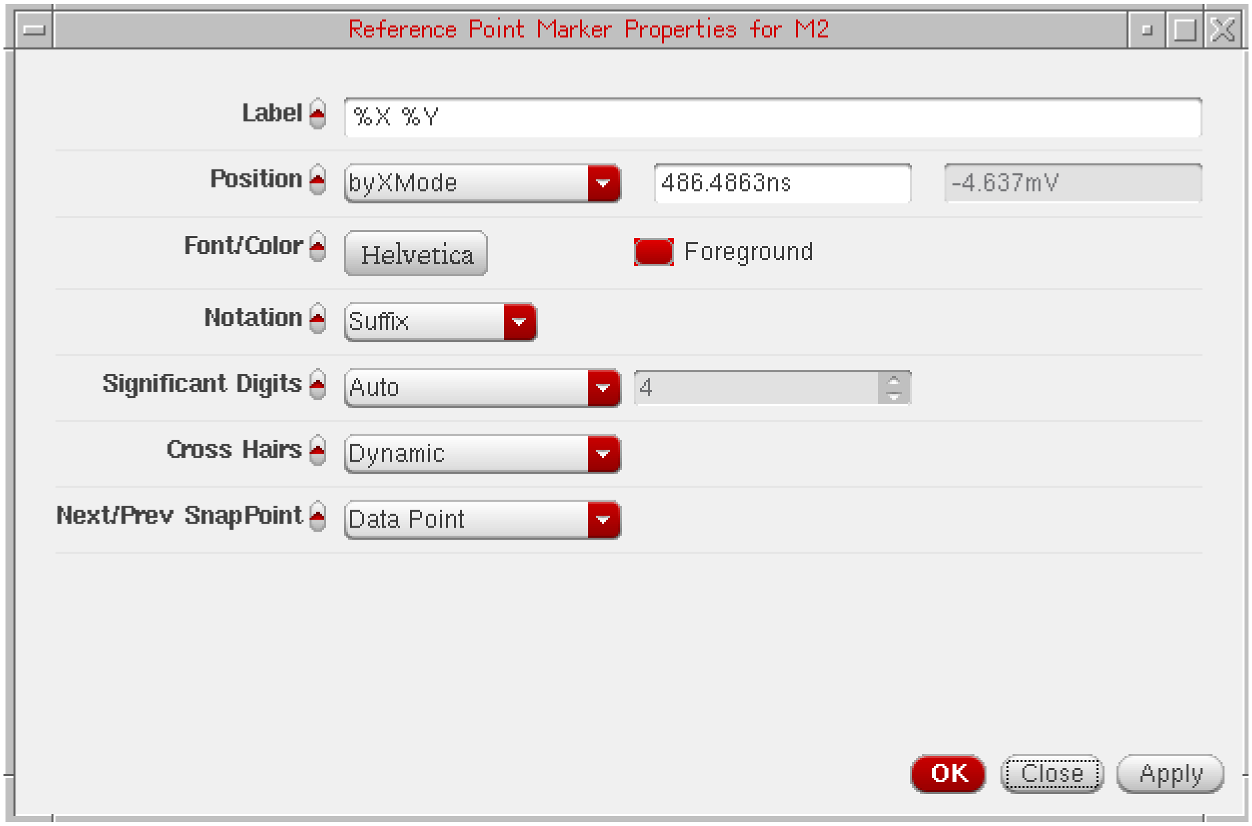
- Select the delta line and choose *Marker – Show Child Labels*.
- Right-click the delta line and choose *Show Child Labels.*

Setting AB Delta Marker Reference Point Properties

To view or change the properties of an AB reference point marker, do one of the following:

- Double-click any of the two point markers that compose the delta marker.
- Right-click a point marker in the delta marker and choose *Marker – Properties*.

  The *Reference Point Marker Properties for <marker name>* form appears.



This form includes the following fields:

- *Label*—Specify the label for the marker. For more information about marker labels, see Adding Markers on page 251
- *Position*—Specify the X and Y coordinates of the reference point marker. You can specify the position by XY, X, and Y modes. By default, the position is set to `byXMode`.
- *Font/Color*—Specify the font type and font color for the marker label.
- *Foreground*—Select the foreground color for the symbol, the arrow and the label.
- *Notation*—Specify the numerical format (notation) to be displayed on labels. The available options are—`Suffix`, `Engineering`, and `Scientific`. Default value: `Suffix`.

- *Significant Digits*—Specify the number of significant digits if you select `manual` from the drop-down. The other option is `auto`.
- *Cross Hairs*—Specify whether you want to display the cross hairs for point marker or delta marker. If you select *Dynamic* from the *Cross Hairs* drop-down list box and click the reference point marker, the cross hairs are displayed.

  Alternatively, to turn on the cross hairs for a delta marker, right-click one of the reference point markers and choose *Draw Cross Hairs – On/Off/Dynamic*.
- *Next/Prev SnapPoint*—Specify the criterion based on which the selected reference point marker should be snapped.

  The available options are—`Local Maxima`, `Local Minima`, `Local Max or Min`, `Specific Y Value`, `Specific X Value`, `Data Point`, `Global Maxima`, and `Global Minima`.

  The `Specific X Value` option is not available for horizontal markers.

  Default value: `Data Point`.

Deleting Delta Markers

To delete the delta marker, right-click the line joining two point markers and choose *Delete*. All the delta markers are deleted.

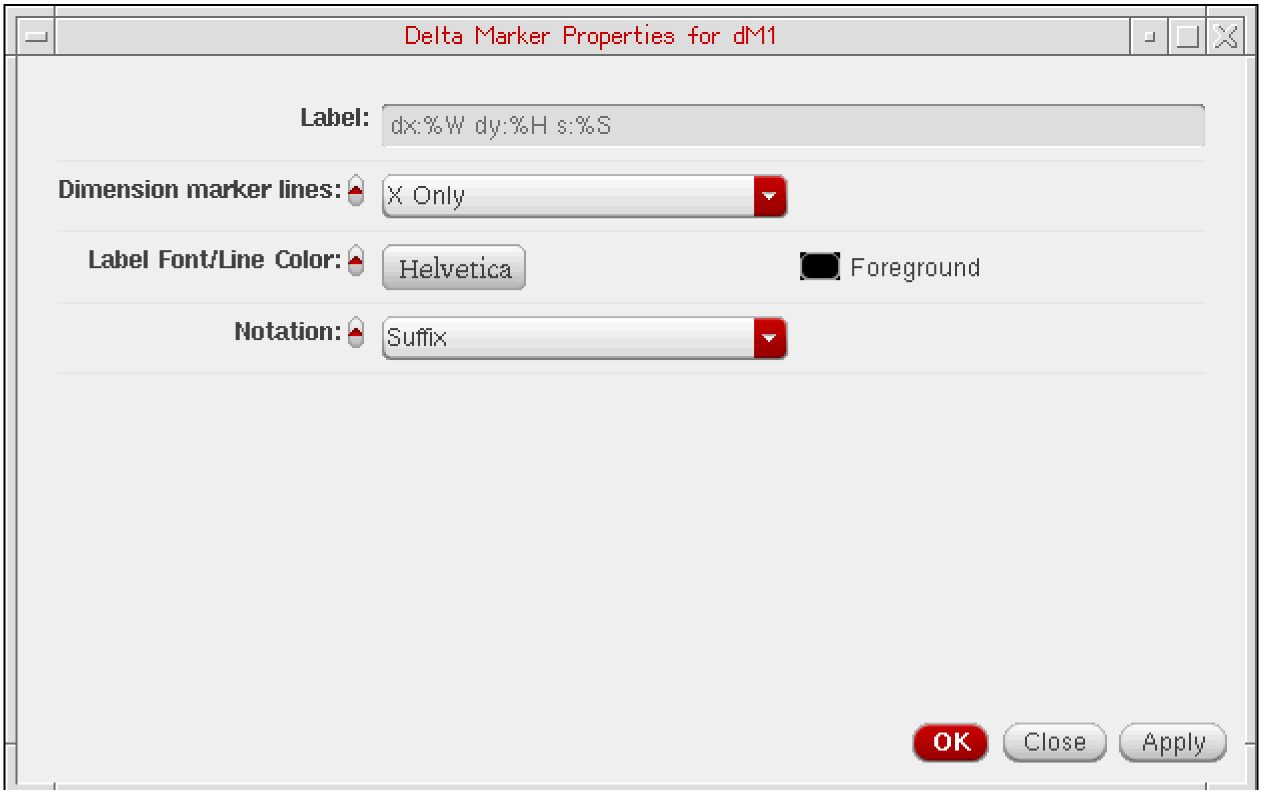
To delete a particular set of delta marker, right-click the point marker which you want to delete and choose *Delete*. The selected point marker and the delta marker joining the point marker are deleted.

Editing Delta Marker Properties

To view or change the delta marker properties, do the following:

➡ Right-click the delta marker line joining two or more point markers and choose *Delta Marker Properties*. The *Delta Marker Properties for <delta marker name>* form appears.

.

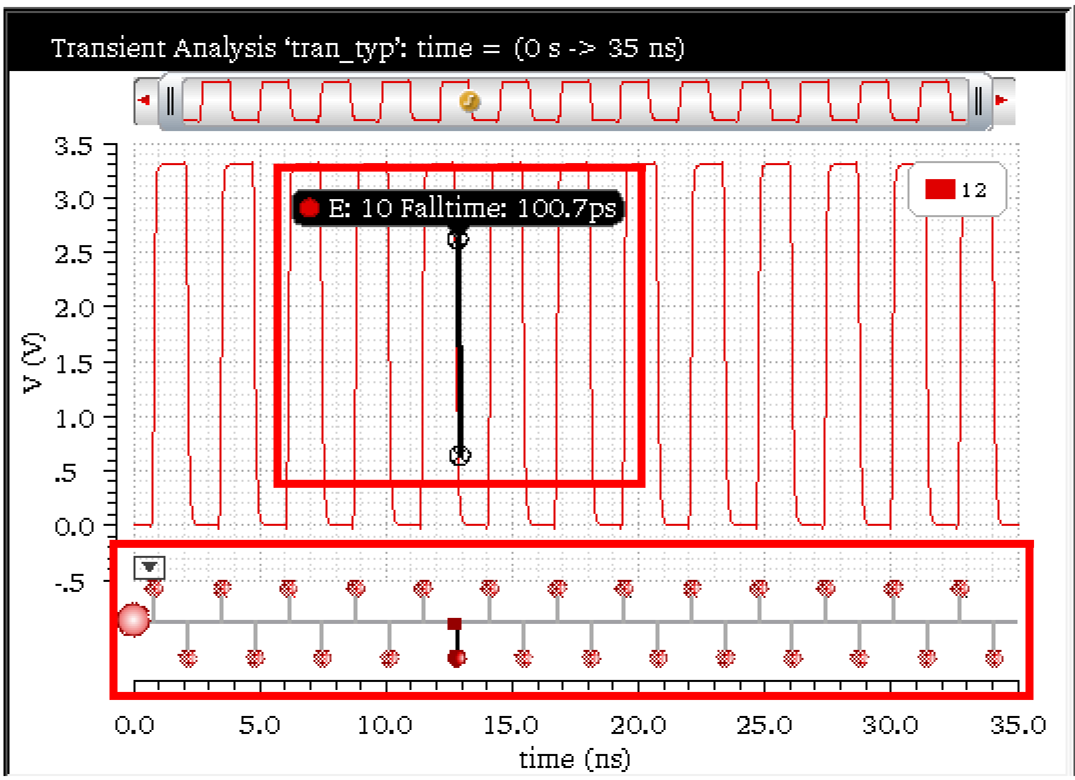


The form includes the following fields:

- *Label*—Specify the label for delta marker. For information about the values that you can use for marker labels, see Adding Markers on page 251.
- *Dimension Marker Lines*—Specify the X and Y-axis dimension lines for markers. To display the X-axis dimension line, select `X Only`. To display the Y-axis dimension line, select `Y Only`. To display both the X and Y-axis dimension lines, select `X and Y`.
- *Label Font/Line Color*—Specify the font properties for the label and also specify the dimension line color.
- *Notation*—Specify the notation to be displayed for the delta values on labels. The available options are—`Suffix`, `Engineering`, and `Scientific`.

Working With Edge Markers

Edge Markers are special markers that can be attached to the rising or falling edges of a trace to measure the transient properties of the selected edge. The figure below displays an edge marker placed on the rising edge of a trace. Note that the marker label displays the risetime value for this edge. If you place the edge marker on the falling edge of the trace, the marker label displays the falltime value for that edge.



The figure above also displays the Edge Browser at the bottom of the strip that you can use to view and analyze the various edges in the trace. By default, the Edge Browser is hidden in the graph. For more information about how to use Edge Browser, see Using Edge Browser on page 159.

This section includes the following topics:

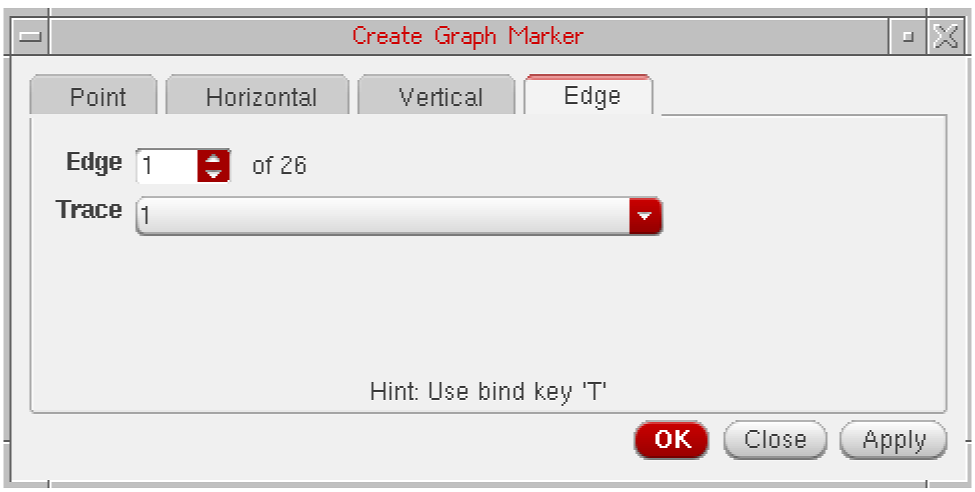
- Creating an Edge Marker on page 279
- Setting Edge Marker Properties on page 280
- Edge Marker Context-Sensitive Menu on page 281

Creating an Edge Marker

You can create an edge marker by using one of the following methods:

- Choose *Marker – Create Marke*r.
- Click the  button on the Marker toolbar.
- Use the Transient Measurement assistant to create an edge marker. For more information about transient measurement, see Transient Measurement on page 155
- Right-click the Edge Browser and choose *Create Edge Marker*.

  The *Create Graph Marker* form appears, as displayed in the figure below.



  In this form, on the *Edge* tab, specify the following fields:

  - *Edge*—Specify the edge number where you want to create an edge marker. The total number of edges are displayed next to this field.
  - *Trace*—Select a trace from the drop-down on which you want to place the edge marker. This drop-down lists the name of all traces that are plotted in the selected window.
  - Click *OK*.

  An edge marker is placed at the specified edge on the selected trace.

- Place the mouse pointer on the edge where you want to create an edge marker and press the bindkey `T`.
- Right-click an edge in the Edge Browser and choose *Create Edge Marker*. The edge marker is placed at the selected edge of the trace in the graph window.

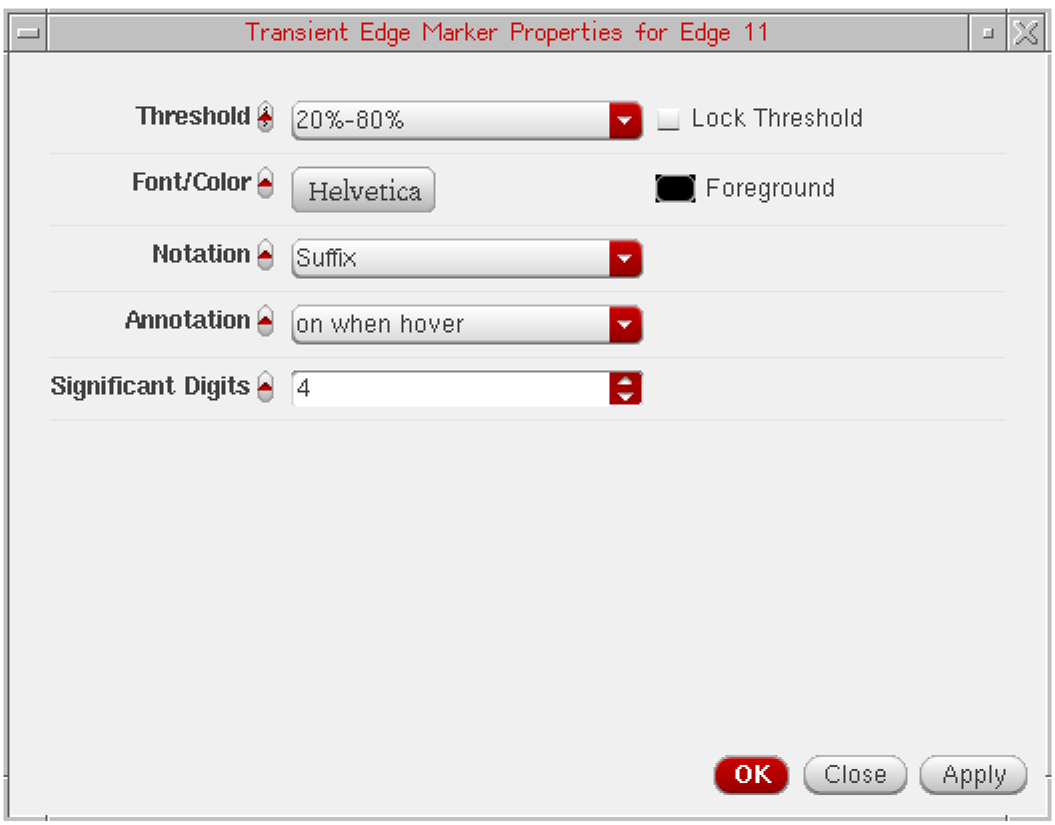
- In the Transient Measurement assistant, select the edge on which you want to place the edge marker from the *Edge* drop-down and click the  button. For more details, see Transient Measurement on page 155.

Setting Edge Marker Properties

Do the following to set the properties for the edge markers:

- Right-click the edge marker and choose *Edge Marker Properties*.
- Select the edge marker and choose *Marker – Properties.*
- Select the edge marker and press the bindkey `Q`.

  The *Transient Edge Marker Properties for Edge <edge number>* form appears



This form includes the following fields:

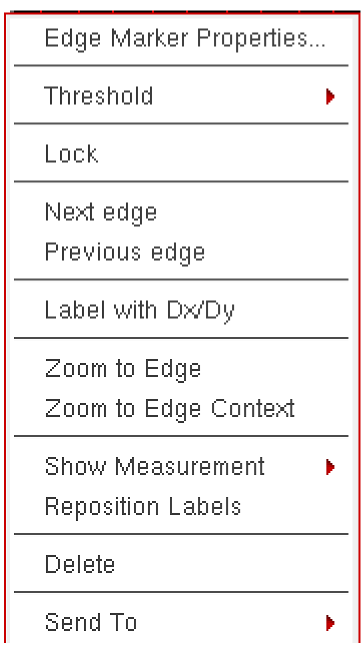
- *Threshold*—Specify the threshold value. The available options are: *10%-90%*, `20%-80%`, and `30%-70%`. The default value is *20%-80%*. The threshold value is controlled by the following environment variable:

```
envSetVal(“viva.trace” “threshold” 'string “20_80”)
```

- *Lock Threshold*—Select this check box to lock the threshold value so that the threshold setting for an individual marker is not overridden if you change the settings of the trace.
- *Font/Color*—Specifies the font and color of the edge marker.
- *Notation*—Specifies the notation of the edge marker. The valid values are `Scientific`, `Engineering`, and `Suffix`. The default value is `Suffix`.
- *Annotation*—Specifies the event when to display the risetime and falltime values on the marker labels. The available values are `off`, `on when hover`, and `on`. The default value is `on when hover`.
- *Significant Digits*—Specifies the number of significant digits to be displayed in the calculated values. The default value is `4`.

Edge Marker Context-Sensitive Menu

Right-click the edge marker to use the various options listed in the context-sensitive (shortcut) menu. The following figure displays the available menu options:



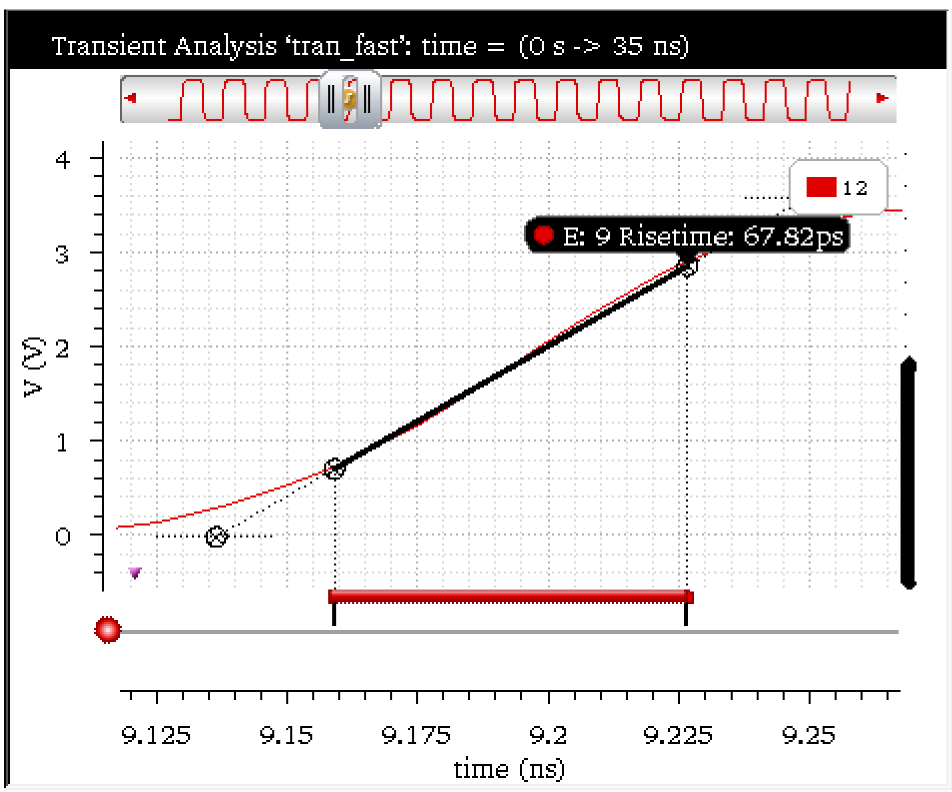
The shortcut menu options are explained as below:

- *Edge Marker Properties*—Opens the Edge Marker properties form, where you can specify the marker properties for the selected marker. For more information on marker properties, see Setting Edge Marker Properties on page 280.

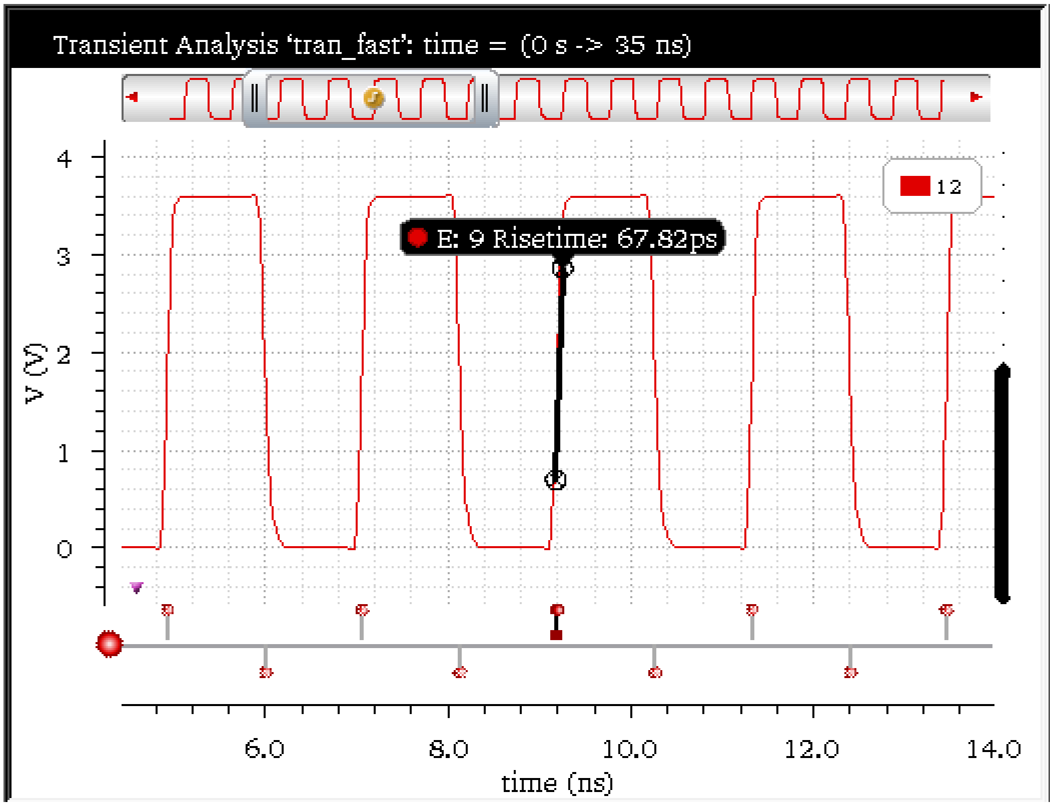
- *Threshold*—Specifies the threshold value for the selected edge marker. The available options are: *10%-90%*, `20%-80%`, `30%-70%`, and `default`. The default value is *20%-80%*. The threshold value is controlled by the following environment variable:

  ```
  envSetVal("viva.trace" "threshold" 'string "20_80")
  ```

- *Lock*—Locks the threshold value so that the threshold setting for an individual edge marker is not overridden if you change the settings of the trace.
- *Next Edge*—Moves the selected edge marker to the next edge on the trace.
- *Previous Edge*—Moves the selected edge marker to the previous edge on the trace.
- *Label with Dx/Dy*—Creates a delta marker by adding two point markers and a delta line between them.
- *Zoom to Edge*—Zooms in the selected edge marker to view the details of the edge on which it is placed. The following figure illustrates the zoomed-in graph.



- *Zoom to Edge Context*—Zooms in the selected edge marker to the context of the edge on which it is placed. The following figure illustrates the zoomed-in graph.



- *Show Measurement*—Specifies the event on which you want to show the measurement label for the selected edge marker. The available options are `off`, `on when hover`, and `on`. You can also set the following environment variable to specify an event:

```
envSetVal("viva.transEdgeMarker""showMeasSummary"'string "on
when hover")
```

- *Reposition Labels*—Moves the selected marker label back to its default position.
- *Delete*—Deletes the selected edge marker.
- *Send To*—Sends the edge marker measurements, such as slewRate, overshoot, undershoot, risetime and falltime to ADE or Calculator.

Working with Circular Graphs

You can display complex data values from AC analysis in the form of Smith charts and polar plots. The Smith chart shows the unity circle that is R = 1 circle, the resistance circles, and the reactance circles.

This section includes the following topics:

Creating a Circular Graph

To create a circular graph, do the following:

1. Open a results directory in the Results Browser, and then open the *ac-ac* analysis folder.

   The *Graph Type* drop-down list appears on the Results Browser toolbar.

2. Select a graph type from this list. The available graph types are—`Default`, `Rectangular`, `Polar`, `Impedance`, `Admittance`, and `RealvsImag`.

   To create a polar plot, select the `Polar` graph type.

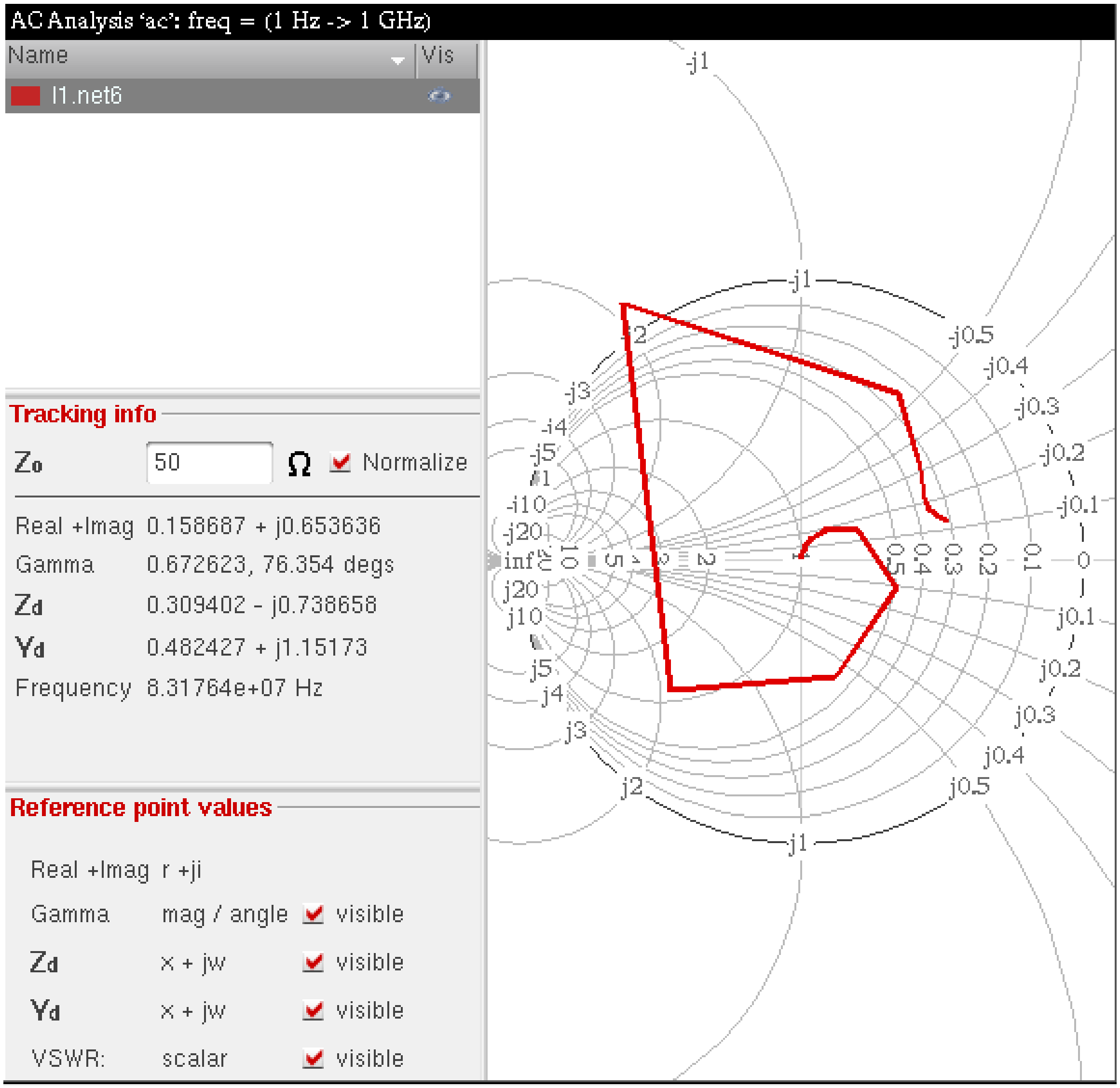
   To create a Smith chart, select either the `Impedance` or `Admittance` graph type.

   For more information about the graph types, see Selecting the Graph Type on page 50.

3. Plot a signal in the selected destination. For more information about how to plot a signal, see Plotting Signals on page 47.

   For information about how to select a plot destination, see Selecting the Signal Plot Destination on page 48.

The circular graphs can be of two types—Smith charts and polar graphs. The Smith chart can further be of two types—Impedance Smith Charts (Z Smith) and Admittance Smith Charts (Y Smith). The circular graph of type Z Smith is displayed in the figure below.



After a Smith chart is plotted, the following sections are displayed in the trace legend area on the left:

- *Name*—Displays the trace name and trace color.
- *Visibility*—Controls the display of the trace. You can show or hide the trace by clicking the visibility button ().

- *Tracking Info*—Displays the following tracking information for the points that you click or point to:
  - *Zo*—Displays the characteristic impedance (`Zo`). The default `Zo` value is 50.
  - *Normalize*—Normalizes the readings. When you select the *Normalize* check box, the Smith reference values are multiplied with the impedance value that you specify in the `Zo` field.

    Note: When you change the `Zo` value and the *Normalize* check box is not selected, press `Tab` or `Enter` to update the reference values on the Smith chart.
  - *Real + Imag*—Displays the real and the imaginary values of the point selected on the trace
  - *Gamma*—Displays the magnitude and the angle of the selected point on the trace.
  - *Zd*—Displays the impedance of the point selected on the trace.
  - *Yd*—Displays the admittance of the point selected on the trace.
  - *Frequency*—Displays the frequency of the point selected on the trace.
- *Reference point values*—Displays the reference marker readout. The reference point readout includes *Real + Imag*, *Gamma*, *Zd* and *Yd* values of the reference point, and the Voltage Standing Wave Ratio (*VSWR*). The VSWR is a scalar value.

  The resistance and reactance circles, along with the reflection coefficient and VSWR circle are displayed on the graph. You can turn these values off and on by clicking the respective *visible* check boxes next to each field.

The default scale attributes for circular graphs are controlled by the values assigned to the variables in the `.cdsenv` file. For more information, see Appendix A, "Virtuoso Visualization and Analysis XL Tool Environment Variables."

Important

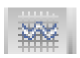
If you plot circular graphs that have the same plot type, the circular graphs are plotted in the same subwindow. For example, two Y-Smith or two Z- Smith can be plotted in the same window. However, if you plot graphs from different plot types, the graphs are plotted in a new subwindow. For example, Y-Smith and Z-Smith are plotted in two different subwindows.

Tracking Cursor

The tracking cursor displays the trace name, trace color, and frequency of the point you that you select or point to on the circular graph. For a Smith chart, the tracking cursor also displays the real and imaginary values. However, for polar plots, the tracking cursor displays the magnitude and angle, instead of the real and imaginary values.

The rest of the circular graph values are displayed dynamically in the various sections in the trace legend area.

To show or hide the tracking cursor, do one of the following:

- Right-click anywhere in the graph and choose Trace Marker Always Visible.
- Choose *Marker – Tracking Cursor*
- Click the  button on the Marker toolbar.

You can also set the `traceMarkerDisplay` .cdsenv variable to display values for the tracking cursor, which is also called a trace marker. The following formats are supported to display the trace marker values:

- `%C` —Displays the real and imaginary Cartesian values
- `%Z`—Displays the impedance values, such as resistance and reactance
- `%A`—Displays the admittance values, such as conductance and susceptance
- `%R`—Displays the reflection coefficients, such as mag and angle
- `%P`—Displays the polar values, such as mag and angle
- `%F`—Displays the frequency value, which includes the independent axis data

Displaying Symbols on Circular Traces

To display symbols on a circular trace, do one of the following:

- Choose *Trace – Symbols On.*
- Right-click the trace and choose *Symbols On*.
- Right-click the trace and choose *Trace Properties*.

The *Complex Trace Properties for <trace-name>* form appears.

In this form, select the *Turn On Symbols* check box to display symbols on the trace. By default, the symbols are displayed for all trace points. To display symbols for a given number of trace points, specify the count in the *Points per Symbol* field.

After turning on the symbols, you can select the symbol type by doing one of the following:

- Right-click the trace and choose *Symbols,* and then select the symbol type that you want to apply
- Right-click the trace and choose *Trace Properties*.

  The *Complex Trace Properties for <trace-name>* form appears. In this form, select the symbol type from the *Symbol* drop-down list.

Adding Markers on Circular Graphs

You can add reference, point, circular, and delta markers on circular graphs. By default, the markers in the circular graphs are always visible. To hide the markers you can set the `tracemarkeralwaysvisible .cdsenv` variable to false.

Adding a Smith Reference Point Marker

To add a Smith reference point maker on a circular graph, select a point on the trace and press the bindkey `R`. The reference maker is created from the center to the selected point. The readout for the reference point marker—`Real + Imag`, `Gamma`, `Zd`, `Yd`, and `VSWR`—is displayed in the *Reference point values* section in the trace legend area.

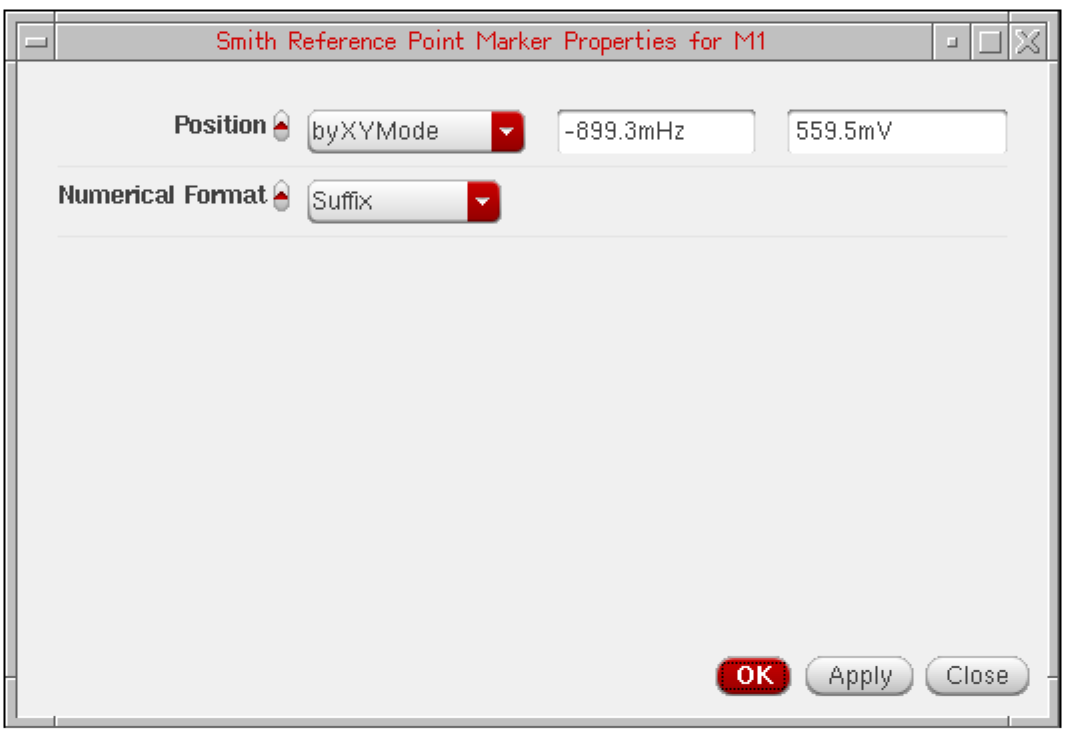
Note: You can add multiple reference markers on the circular graph. When you select a reference marker, the reference point values and graphical measurements change based on the circles and annotation for the selected reference point.

Setting Smith Reference Point Marker Properties

To set the reference marker properties, right-click a reference marker and choose *Smith Ref Point Properties*.

The *Smith Reference Point Marker Properties for <marker-name>* form that includes the following fields appears:

- *Position*—Specify the X- and Y-axis coordinates where you want to place the reference marker.
- *Numerical Format*—Select the numerical format as `Scientific`, `Engineering`, and `Suffix`.



Adding a Point Marker

To add a point marker, do one of the following:

- Select the point on the trace where you want to add the point marker and press `M`.
- Press the bind key `M`.

  The point marker is created on the point you click on the trace.

  To change the point marker properties, right-click the point marker and choose *Marker Properties*. The *Point Marker Properties for <marker-name>* form appears. For more information about the point marker properties form fields, see Adding Markers on page 251.

Adding a Circular Marker

To add a circular marker, do the following:

➡ Choose *Marker – Create Marker*.

The *Create Graph Marker* form appears. On the *Circular* tab, in the *Position* field, specify the real and imaginary values for the circular marker. In the *Radius* field, specify the radius. For more information about the circular marker fields displayed in the *Create Graph Marker* form, see Adding Markers on page 251.

Deleting Markers

To delete a marker, see Deleting a Marker on page 269.

Zooming Circular Graphs

You can zoom in or out the circular graph by doing the following:

- To fit the circular graph to the data values, choose *View – Fit* or right-click anywhere on the circular graph and choose *Fit.*
- To fit the trace in the circular graph to the window, do one of the following:
  - Right-click the trace in the circular graph and choose *Fit Trace*.
  - Select the trace in the circular graph and choose *View – Fit Trace*.
  - Select the trace in the circular graph and choose *Trace – Fit Trace*.

- To fit the circular graph to the Smith values, do one of the following:
  - ❑ Right-click anywhere on the circular graph and choose *Smith Fit*.
  - ❑ Choose *View – Fit Smith*.

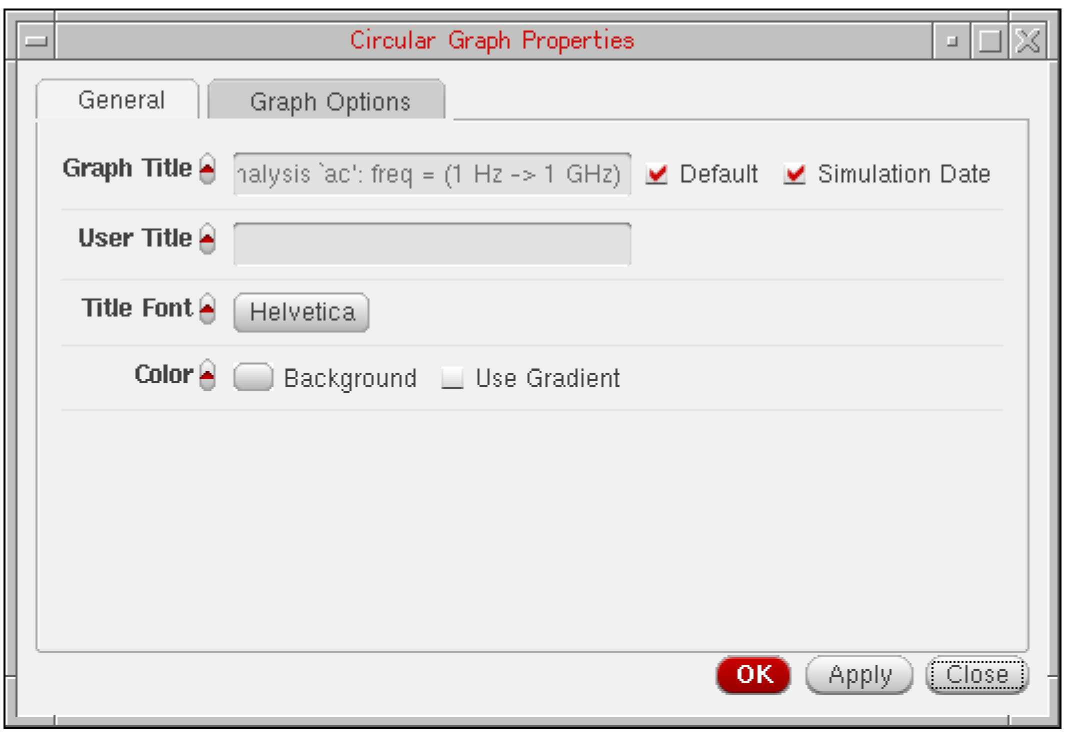
Note: When you zoom in a graph, the labels for the zoomed-in area are displayed outside the standard unit circle.

Editing Circular Graph Properties

To set the properties of the circular graph, do one of the following

- Choose *Graph – Properties*.
- Right-click anywhere in the circular graph and choose *Smith Graph Properties* for a Smith chart and *Polar Graph Properties* for a polar graph.

  The *Circular Graph Properties* form appears.



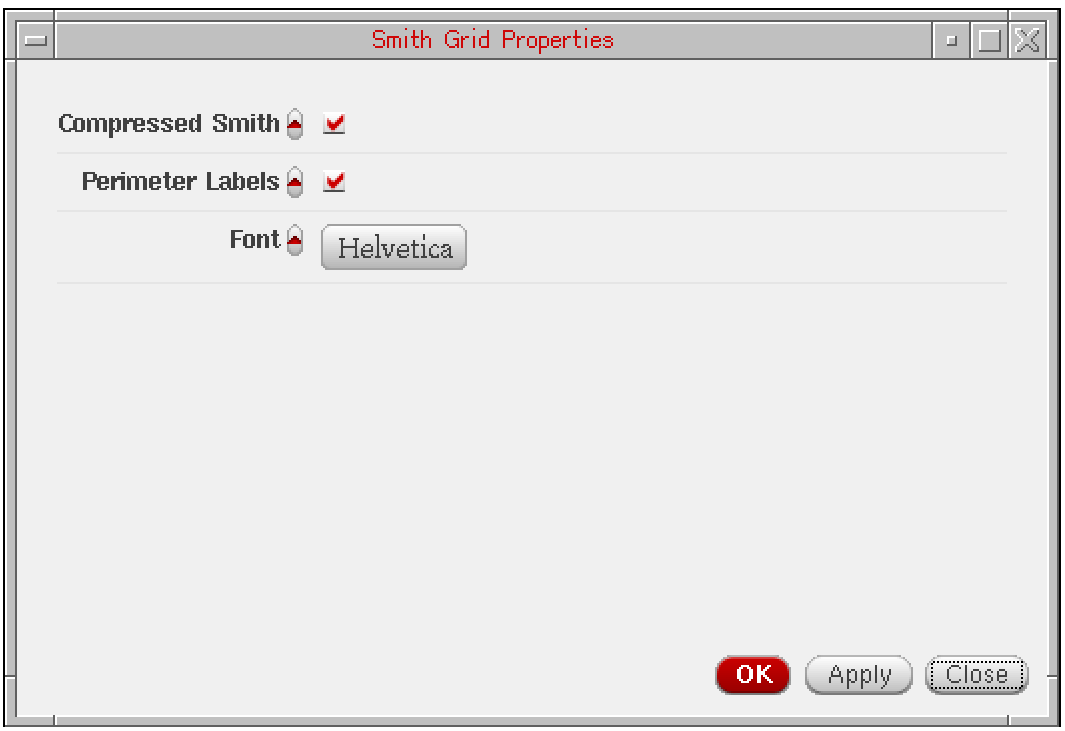
  This form includes the following two tabs:

  - ❑ *General*—For information about the fields displayed on the *General* tab, see Editing Graph Properties on page 108.

- *Graph Options*—Includes the following fields:
  - *Font*—Select the font properties of the Smith Chart.
  - *Notation*—Select the graph notation as `Scientific`, `Engineering`, or `Suffix`. Default value: `Suffix`.
  - *Grid Type*—Select the grid type as `Polar`, `Impedance`, or `Admittance`.
  - *Smith Data*—Specify the characteristic impedance Zo. Select the *Normalize Smith Value* check box if you want to normalize the Smith data values according to the impedance you specify.
- Click *OK*.

Setting Smith Grid Properties

To set the Smith grid properties, right-click a Smith chart and choose *Smith Grid Properties*. The *Smith Grid Properties* form appears.



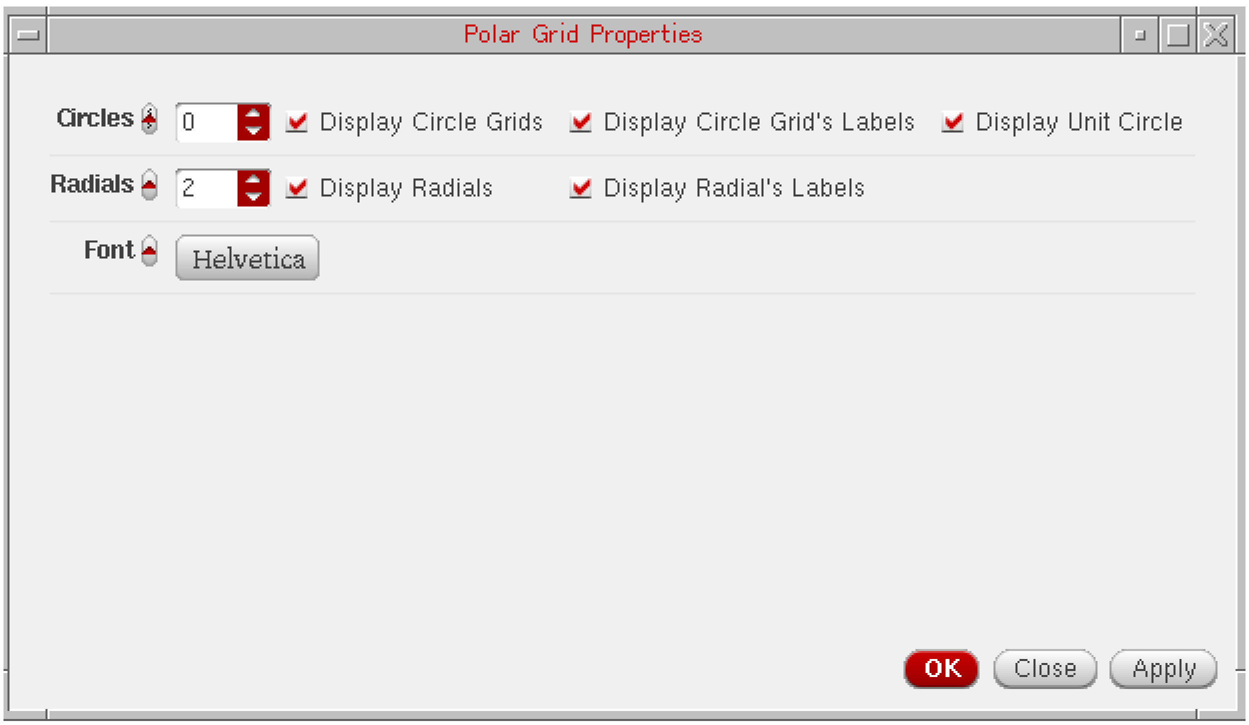
The form includes the following fields:

- *Compressed Smith*—Select this check box if you want to display the extra horizontal grid lines (arcs) outside the Smith chart boundary.

- *Perimeter Labels*—Select this check box if you want to display labels at arc intersections on the Smith unity circle.
- *Font*—Specify the font properties for the Smith chart labels.

Setting Polar Grid Properties

To set the polar grid properties, right-click a polar plot and choose *Polar Grid Properties*. The *Polar Grid Properties* form appears.



This form includes the following fields:

- *Circles*—Specify the number of circles you want to draw on the polar grid. Default value is `0`.
  - ❑ Select the *Display Circular Grid's Labels* check box if you want to display the labels for circular grids.
  - ❑ Select the *Display Unit Circle* check box if you want to display a unit circle, which means circle with R=1.
- *Radials*—Specify the number of radials you want to draw in each quarter of the polar grid. Default value is `2`.

- ❑ Select the *Display Radials* check box if you want to display the radials in the polar graph.
- ❑ Select the *Display Radial's Labels* check box if you want to display labels for radials in the polar graph

- *Font*—Select the font properties for the polar grid labels.

Setting Dependent Modifiers for a Complex Trace

You can set the dependent modifiers for an AC or a complex dataset, such as Mag, dB10, dBm, and dB20. You can also calculate these modifiers based on the resulting eye diagram and the spectrum waveform.

To change the modifier for an AC or a complex dataset, do one of the following:

- Right-click the AC waveform and choose *Dependent Modifier – Magnitude/Phase/ dB10/dB20/dBm/WrapPhase/Real/Imag*.

  The selected modifier is plotted in the window in which the graph was plotted.

- Right-click the AC waveform and choose *Trace Properties*.

  The *Complex Trace Properties* form appears. In this form, select from the *Dependent Modifier* drop-down list box the modifier you want to apply to the trace and click *OK*.

  The selected modifier is plotted in the window in which the circular graph was plotted.

Setting Bindkeys

A bindkey is a key or a sequence of key press events linked (bound) to a task. When you press the key or the sequence of keys, the associated task is performed. The Virtuoso Visualization and Analysis XL too provides a set of default bindkeys, which are displayed next to the relevant commands on the menus. These bindkeys can be overwritten or modified by a customized bindkey file.

The sample bindkey files are found at the following locations:

`$CDSHOME/tools/dfII/samples/local/vivaBindKeys.il`

`$CDSHOME/tools/dfII/samples/local/vivaJavaBindKeys.il`

To view all the bindkeys for the Virtuoso Visualization and Analysis XL tool, choose *Help - Bindkey*. The *Bindkey Editor* appears that includes all the bindkeys for all applications, such as *vivaBrowser*, *vivaCalculator*, and *vivaGraph*.

You can map keystrokes to the tasks that you perform in the window. However, you cannot bind mouse actions to tasks. A task is defined as follows:

```
Task = { graph_task | menu_item_task | skill_function }
```

`graph_task`
: Specifies the task that the bindkey is linked to. You can link a bindkey to any of the following tasks:
`marker`, `vertmarker`, `horizmarker`, `deltamarker`, `vertcursor`, `horizcursor`, `deltacursor`, `tracecopywin`, `tracemovewin`, `tracecopysubwin`, `tracemovesubwin`, `logscale`, `cut`, `copy`, `paste`, `delete`, `cancel`, `undo`, `cancel`, `traceinfo`, `pandown`, `panup`, `panleft`, `panright`, `zoom`, `zoomx`, `zoomy`, `zoomin`, and `zoomfit`

`menu_item_task`
: Specifies the item name in the following format `graph.`*`menu`*`.`*`submenu1`*`.`*`submenu2`*.
For example, panning to the right is defined as follows:
`graph.zoom.pan.panright`.

`skill_function`
: Specifies the SKILL function call, such as `awvFitMenuCB`()
For information about SKILL functions, see the *SKILL Language Reference*.

A keystroke is defined as follows:

```
keystroke = simple_keystroke | composite_keystroke
```

`simple_keystroke`
: A single letter, number, symbol, or key name.
Examples: `a`, `2`, `@`, `Up`, `F1`

`composite_keystroke`
: The format is
`modifiers``<key>`*`simple_keystroke`*
where *`modifiers`* = `alt`, `ctrl`, `meta`, `shift`, `control`, `super`, `hyper`, `mod1`, `mod2`, `mod3`, `mod4`, `mod5`
Examples: `meta<key>Right`, `AltShift<key>F2`

4

Working with the Calculator

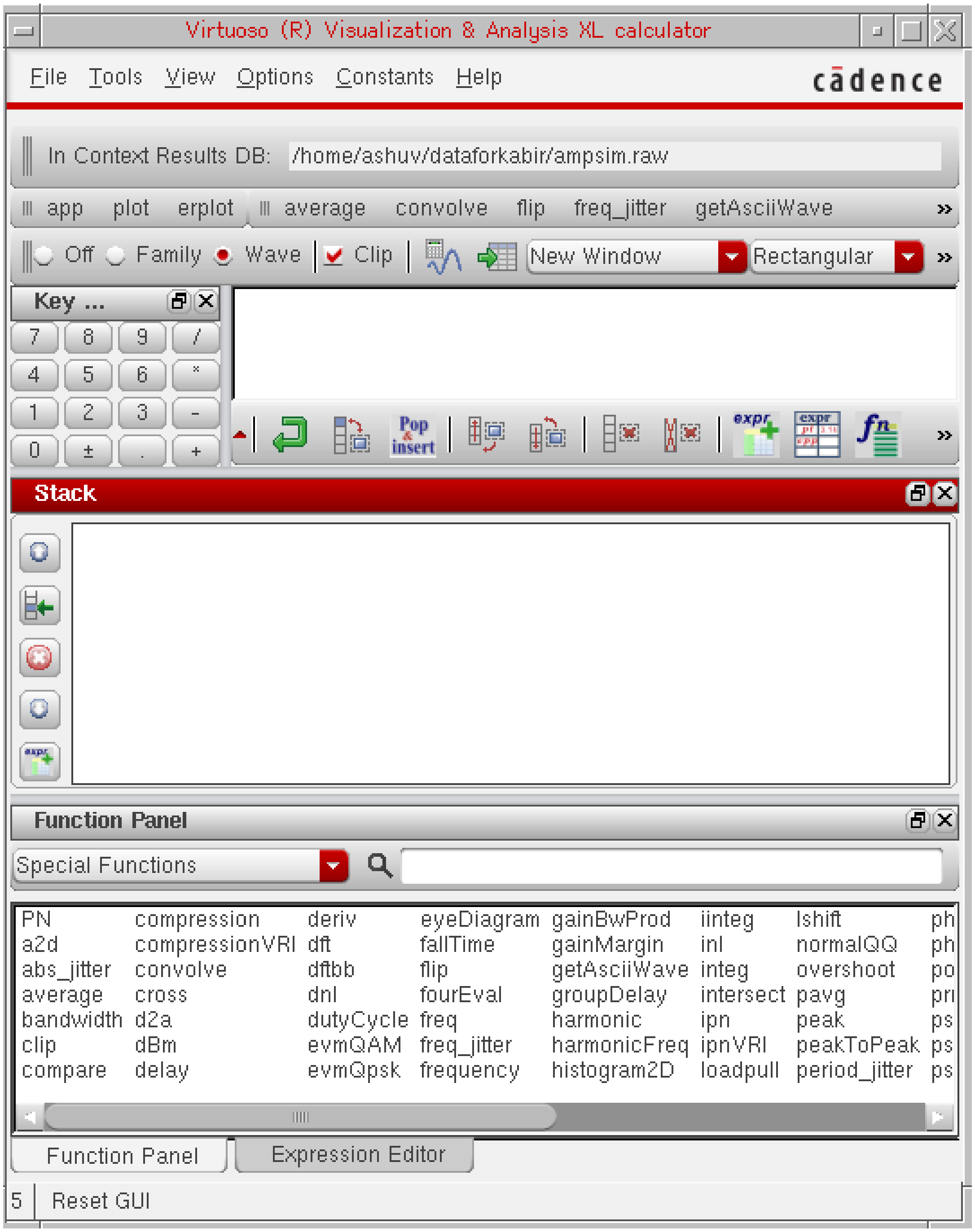
The Virtuoso® Visualizatiion and Analysis XL Calculator is a tool that you use to perform calculations on the data generated by a simulator.

This chapter includes the following topics:

About the Calculator

Virtuoso Visualization and Analysis XL Calculator is a tool that you use to perform calculations on signals and datasets generated by a simulator. You can also use the

Calculator to build expressions on simulation data and can save these expressions for future use.



You can run the Calculator from Virtuoso Visualization and Analysis XL tool or from the Analog Design Environment (ADE). When you run the Calculator from Virtuoso Visualization and Analysis XL tool, you can open the Calculator either in the MDL or SKILL mode. In both the modes, you use the Results Browser to access simulation results. The simulation results are the signals generated after the simulation is run in a simulation environment, such as ADE and Spectre. To analyze these signals for a particular condition set, you can build expressions in the Calculator and evaluate them.

For example, if you want to calculate the difference between the output signal (`Vout`) and the input signal (`Vin`) for a simulation, you can build an expression `Vout-Vin` by using the Calculator. To build this expression, you can get the expressions for `Vin` and `Vout` using the Results Browser.

However, if you are working in ADE and want to build expressions for the latest simulation run to analyze design specifications, you can open the Calculator directly from ADE. You then build the required expressions in the Calculator and import the expressions to ADE before you run the simulation. The expressions are evaluated immediately after you run the simulation. The generated output helps you to modify the simulation circuit.

In ADE, you can use the Calculator to build expressions, which are then evaluated in ADE after the simulation is run. When you run the Calculator from Virtuoso Visualization and Analysis XL tool, you can use the Calculator to both build and evaluate expressions.

When you run the Calculator from Virtuoso Visualization and Analysis XL tool, you work on signals, whereas when you run the Calculator from ADE, you typically work on schematic objects, such as nets and terminals. In addition, when you run the Calculator from ADE, you can open the Results Browser from the Calculator and work on the saved simulation results, if required.

Calculator works in two modes—RPN Mode and Algebraic Mode. The two Calculator modes provide different sets of operations for building and evaluating expressions. The default Calculator type is set to RPN Mode.

Calculator contains various dockable assistants, a menu bar, a Buffer, and toolbars that you can use to build and evaluate expressions. For more details about these GUI elements, see Using the Calculator Graphical User Interface (GUI) on page 304.

Opening the Calculator Window

This section describes the different methods that you can use to open the Calculator.

- Opening the Calculator from Virtuoso Visualization and Analysis XL Tool on page 301
    - Opening the Calculator from Virtuoso Visualization and Analysis XL in SKILL Mode on page 301

Opening the Calculator from Virtuoso Visualization and Analysis XL Tool

Depending upon the simulation environment you are working in, you can open the Calculator in the SKILL or MDL mode from the Virtuoso Visualization and Analysis XL tool. When you run the Virtuoso Visualization and Analysis XL tool, the Results Browser appears. In the Results Browser, you select the saved simulation results on which you want to build and evaluate expressions.

Opening the Calculator from Virtuoso Visualization and Analysis XL in SKILL Mode

Perform the following steps to open the Calculator from Virtuoso Visualization and Analysis XL in the SKILL mode:

1. Start the Virtuoso Visualization and Analysis XL tool by typing the following command in a terminal window:

   ```
   viva -expr skill &
   ```

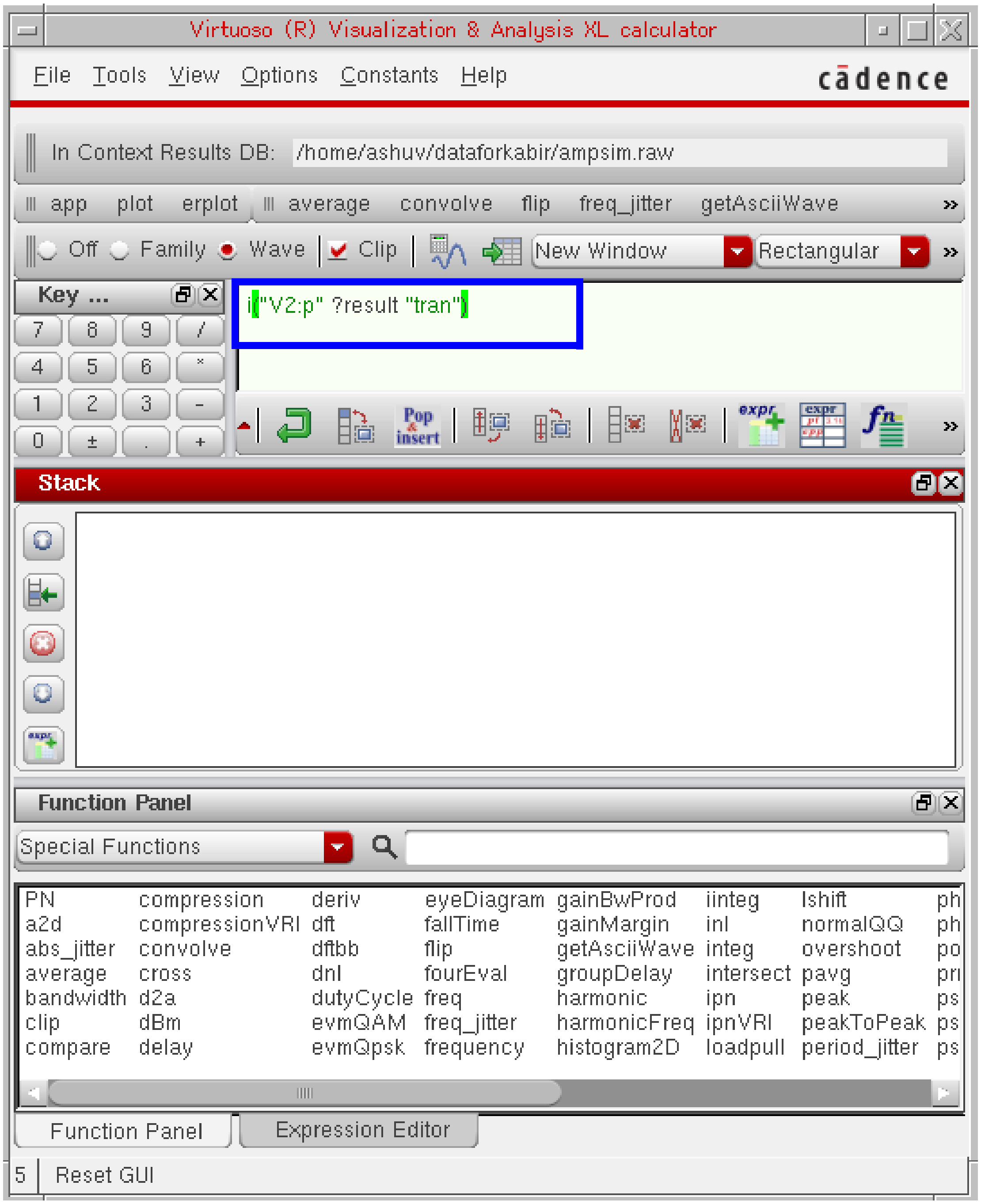
   The Virtuoso Visualization and Analysis XL window appears. This window consists of Results Browser as an assistant.

   Note: If you type only `viva &` in the terminal window, the default mode is SKILL.

2. In the Results Browser, select the required signal and do one of the following:
   - Right-click the signal and choose *Calculator*.
   - Choose *Tools – Calculator*.

   The Calculator window appears. The signal you have selected in the Results Browser appears in the Buffer.

See Selecting Signals and Waveforms to Build Expressions on page 332 to know how to select a signal in the Results Browser.



Note: To verify which result data directory is currently in use, view the *In Context Results DB* field in the Calculator window.

Opening the Calculator from Virtuoso Visualization and Analysis XL in MDL Mode

You can open the Calculator in MDL mode only if you are using the Spectre simulation environment. Perform the following steps to open the Calculator in the MDL mode:

1. Start the Virtuoso Visualization and Analysis XL tool by typing the following command in a terminal window:

   ```
   vivamdl &
   ```

   The Virtuoso Visualization and Analysis XL window appears. This window consists of Results Browser as an assistant pane.

2. Select a signal for which you want to build expressions and do one of the following:

   a. Right-click the signal and choose *Calculator*.

   b. Choose *Tools – Calculator*.

   The Calculator window appears. The signal selected in the Results Browser appears in the Buffer. For more information about the Buffer, see Stack on page 322.

   See Selecting Signals and Waveforms to Build Expressions on page 332 to know how to select a signal in the Results Browser.

Note: In the IC 6.1.5 release, if you open Calculator in the MDL mode, it continues to open the Java-based Calculator. For information about using the Calculator in the MDL mode, see *WaveScan User Guide*, version 5.1.41.

Opening the Calculator from CIW in Stand-Alone Mode

To open the Calculator in the stand-alone mode, perform the following:

➡ In the CIW, choose *Tools – Calculator*.

The Calculator window appears. Click the  button to open the Results Browser and select the signal for which you want to build expressions.

Opening the Calculator from ADE

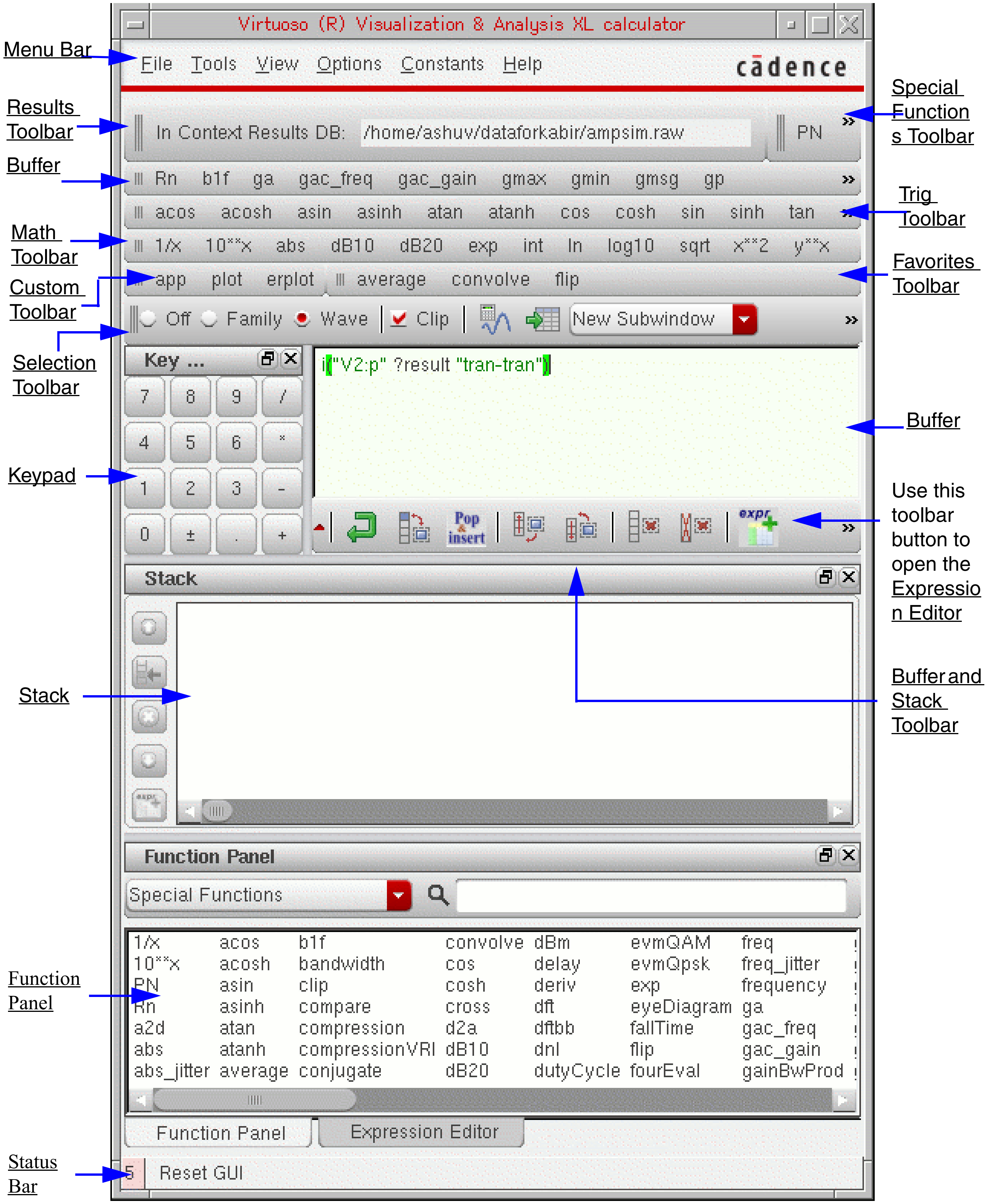
The Calculator can be opened in ADE L or ADE XL to build expressions to analyze simulation output. For detailed information about how to open and work with the Calculator in ADE, see Working with the Calculator in ADE on page 348.

Using the Calculator Graphical User Interface (GUI)

The Calculator graphical user interface (GUI) consists of various toolbars, a menu bar, a Buffer, a status bar, toolbars, and dockable assistants—Expression Editor, Function Panel, Stack, and Keypad— that you use to build and evaluate expressions. You can show, hide, move, and place the dockable assistants and toolbars anywhere inside or outside the main Calculator window. For example, by default, the Keypad is placed in the lower-right corner of the Calculator window. However, if required, you can drag the Keypad to place it next to the Buffer. When you perform drag-and-drop operations on assistants and toolbars, they are automatically positioned according to the available space.

Choose *File – Reset GUI* anytime during the session to restore the default placement of toolbars and assistants.

Note: In the IC 6.1.5 release, the improved version of the Calculator helps you open more than one assistant simultaneously.

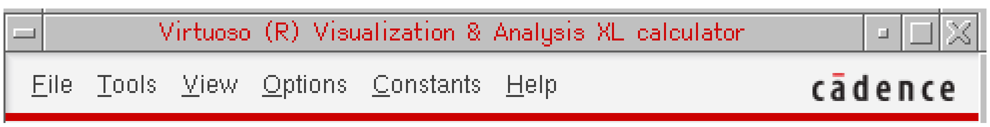

Menu Bar
Results Toolbar
Buffer
Math Toolbar
Custom Toolbar
Selection Toolbar
Keypad
Stack
Function Panel
Status Bar
Special Functions Toolbar
Trig Toolbar
Favorites Toolbar
Buffer
Use this toolbar button to open the Expression Editor
Buffer and Stack Toolbar
Virtuoso (R) Visualization & Analysis XL calculator
File Tools View Options Constants Help
cadence
In Context Results DB: /home/ashuv/dataforkabir/ampsim.raw
PN
Rn b1f ga gac_freq gac_gain gmax gmin gmsg gp
acos acosh asin asinh atan atanh cos cosh sin sinh tan
1/x 10**x abs dB10 dB20 exp int ln log10 sqrt x**2 y**x
app plot erplot average convolve flip
Off Family Wave Clip
New Subwindow
Key ...
i("V2:p" ?result "tran-tran")
Stack
Function Panel
Special Functions
Function Panel
Expression Editor
Reset GUI

This topic describes the following components of the Virtuoso Visualization and Analysis XL Calculator.

- Menu Bar on page 306
- Toolbars on page 309
- Buffer on page 318
- Assistants on page 322

Menu Bar

The menu bar displays the various menus that contain commands for working with the Calculator.



The table below lists a description of the various Calculator menus.

Menu Name	Commands	Description	For More Information, See
File			
	Reset GUI	Resets the Calculator GUI to default settings	
	Close	Closes the Calculator window	
Tools			
	Plot	Plots the signal in the selected graph window	Chapter 3, "Working with Graphs."
	Table	Opens the Virtuoso Visualization and Analysis XL Table.	Working with the Calculator in ADE on page 348
	Browser	Opens the Results Browser window if you want to work on the saved simulation data in ADE	

Menu Name	Commands	Description	For More Information, See
	Send to ADEXL Test	Displays the expression created in the Buffer on the ADE XL *Output Setup* section	
View			
	Display Results Dir	Displays the *In Context Results DB* field on the Result Directory toolbar	Results Toolbar on page 310
	Show Keypad	Displays the Keypad in the lower-right corner of the Calculator window	
	Show Stack	Displays the Stack assistant, which lists expressions that you pushed into the Stack assistant from the Expression Editor and Buffer assistants	
	Math Toolbar	Displays the Math toolbar that includes mathematical functions such as ln and exp	
	Trig Toolbar	Displays the Trigonometry toolbar that includes trigonometric functions such as sin and cos	
	Custom Toolbar	Displays the Custom toolbar that includes buttons to select the mode in which you want to display the output	Custom Toolbar on page 313
	Schematic Selection Toolbar	Displays the Schematic Selection toolbar in ADE L and ADE XL	Schematic Selection Toolbar on page 314
	Configure Schematic Selection Toolbar	Displays the Configure Schematic Selection toolbar form, in which you select the types of simulation analyses, such as tran, ac, dc, and so on. The function buttons for the selected simulation analyses are displayed on the Schematic Selection toolbar.	
Options			

Menu Name	Commands	Description	For More Information, See
	Mode	Sets the Calculator mode *RPN Mode*—Sets the Calculator mode to RPN. Turning this option off changes the Calculator mode to Algebraic *Algebraic Mode*—Sets the Calculator mode to algebraic. Turning this option off changes the Calculator mode to RPN	Building Expressions in the Buffer on page 335
	Notation	Specifies the notation type to be used for displaying data. You can specify one of the following three notation types: *Engineering* - Displays data by using the engineering notation *Suffix* - Displays data by using the suffix notation *Scientific* - Displays data by using the scientific notation	
	Significant Digits	Sets the number of significant digits for the results displayed in the table	
	Edit color	Sets the colors for the various types of Buffer contents, including Results Dir, DataSet name, Calc Function, Node names, Data Access Functions, Number, Brackets, Operators, and Strings	
Constants		Displays a list of constants, such as boltzmann, charge, degPerRad, epp0, pi, twopi, and sqrt2. These constants are used while building expressions.	Adding Constants to Expressions on page 341
Help			
	Contents	Displays the *Virtuoso Visualization and Analysis Tool User Guide*	
	Cadence Online Support	Displays the Cadence customer support website in your default Web browser	

Menu Name	Commands	Description	For More Information, See
	Online User Forum (cdnusers.org)	Displays the online users' forum website in your default Web browser	
	Known Problems and Solutions	Displays *Virtuoso Visualization and Analysis Tool Known Problems and Solutions*	
	What's new	Displays *Virtuoso Visualization and Analysis Tool What's New*	
	About Visualization and Analysis	Displays the version number of the Virtuoso Visualization and Analysis XL tool	

Toolbars

Do one of the following to display the Calculator toolbars that are currently hidden:

➡ From the *View* menu, choose the toolbar you want to display.

➡ Right-click any toolbar and choose the toolbar you want to display.

Calculator has the following toolbars:

- Results Toolbar on page 310
- Test Toolbar on page 314
- Special Functions Toolbar on page 312
- Buffer on page 318
- Trig Toolbar on page 312
- Math Toolbar on page 313
- Custom Toolbar on page 313
- Favorites Toolbar on page 314
- Schematic Selection Toolbar on page 314

- Selection Toolbar on page 315
- Buffer and Stack Toolbar on page 317

Results Toolbar

The Results toolbar displays the *In Context Results DB* field.

In Context Results DB

You build expressions using the data files stored in the results directory. A data file contains output signals that are generated after a simulation is run.

The *In Context Results DB* drop-down list box contains the results directories that are currently open in the Results Browser. By default, the results directory that is set as in-context in the Results Browser is displayed on the top of the list.

You may need to build expressions by using data from one results directory and evaluate the same expression against other results directories. In this case, you can directly open a new results directory from the Calculator by clicking the  (Open Results) button displayed on the right of the Results toolbar. When you click this button, the *Select Waveform Database* form appears. In this form, browse and select the results directory that you want to open. For more information about this form, see Opening a Results Directory on page 38. Note that the results directory that you open from the Calculator becomes the in-context result directory in the Results Browser and Calculator. This new in-context results directory is now used to build or evaluate the expressions.

Tip

Alternatively, you can also type the complete results directory path name in this field and press the `Enter` or the `Tab` key.

When you change the in-context directory in the Results Browser, the *In Context Results DB* field on the Result toolbar displays the path of the changed in-context directory.

It is recommended that you set the database context to the directory you want to use to build expressions because expressions created using any other result directory may result in an error when evaluated.

Note: The database context in the ADE L results toolbar does not change when you open a new in-context results directory from the Calculator.

Important

It is recommended that you set the database context to the directory you want to use to build expressions because expressions created using any other result directory may result in an error when evaluated.

If you do not specify a database context directory, the results directory from which you select the first signal becomes the context results directory, and the directory path is displayed in the *In Context Results DB* field on the Result toolbar.

You can open multiple results directories in the Results Browser, but the *In Context Results DB* field displays the path of the database context directory. For example, if the database context directory is set to the `psf` results directory, this field displays the following value:

```
/home/ashuv/psf
```

Now, if you select a signal from the `psf` results directory, the corresponding expression in the Buffer does not include the results directory name.

```
v(I7.net7" ?result "tran)
```

However, if you select a signal from a results directory that is not the database context directory, the corresponding expression in the Buffer includes the results directory name.

```
v(\\7.net7" ?result "tran" ?resultsDir "./viva_psf)
```

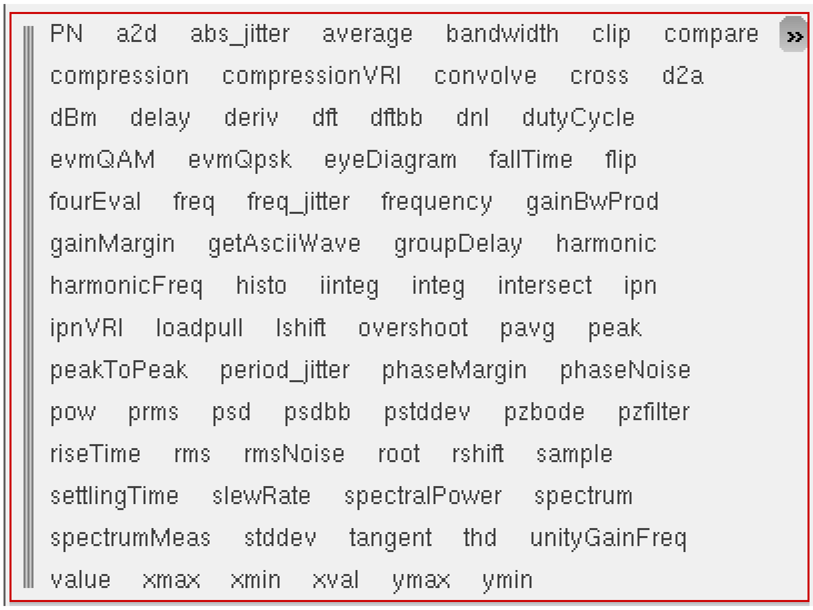
Here, `?resultsDir "./viva_psf"` is the name of the results directory from which the signal is selected.

Note: If the Calculator is opened from ADE, the *In Context Results DB* `value` is determined by the results directory currently selected in ADE.

Important

Starting IC6.1.5ISR3, when you create expressions, the expressions include alias name for the analysis to display the results directory name. For example, the expression `v(I7.net7" ?result "tran)` is now displayed as `v(I7.net7" ?result "tran).`

Special Functions Toolbar



The Special Functions toolbar displays all special functions, such as `a2d`, `average` and `bandwidth`. By default, this toolbar is hidden and when you select to show the toolbar, it is displayed next to the Result toolbar. To view the complete list of functions, click the arrow button on the toolbar.

The Function Panel displays the special functions by default.

RF Functions Toolbar

This toolbar displays the set of RF functions, such as Rn, that can be used while building expressions in the Buffer. This toolbar is hidden by default.

Trig Toolbar

This toolbar displays the set of trigonometric functions, such as acos, that can be used while building expressions in the Buffer.

The Trig toolbar is hidden by default. You can set the trigToolBar .cdsenv variable to hide or show the toolbar.

Math Toolbar

This toolbar displays the set of mathematical functions, such as log, that can be used while building expressions in the Buffer.

The Math toolbar is hidden by default. You can set the mathToolBar .cdsenv variable to hide or show the toolbar.

Custom Toolbar

This toolbar displays the modes in which the results generated by evaluating expressions can be plotted. With the Custom toolbar, you can also work on the expressions that are saved in Stack or Expression Editor. This toolbar is hidden by default.

app

When you click this button, the Buffer contents are appended to the expression table in the Expression Editor. The Buffer is populated with the latest entry in the Stack. To plot or calculate an expression listed in the expression table, select the expression and click the *Eval* button.

plot

When you click this button, the expression in the Buffer is plotted in the append mode in the selected graph window.

The *plot* command is also available in the *Tools* menu.

Note: This command overrides the option selected in the *plot destination* list box in the signal selection toolbar.

erplot

When you click this button, the Buffer contents are plotted in the selected graph in the replace mode.

Note: This button overrides the *plot destination* selected on the Selection toolbar.

Favorites Toolbar

The Favorites toolbar displays the functions that are set as favorite functions in the Function Panel. By default, this toolbar is displayed next to the Custom toolbar and does not contain any function. You can add frequently used functions to this toolbar for quick access.

To add a function to the Favorites toolbar:

➤ Right-click the function in the Function Panel and choose *Add to Favorites*.

To delete a function from the Favorites toolbar:

➤ Right-click the function in the Function Panel and choose *Delete From Favorites*.

When you add a function as a favorite, it is simultaneously added to the Favorites toolbar and to the *Favorites category* in the Function Panel.

Test Toolbar

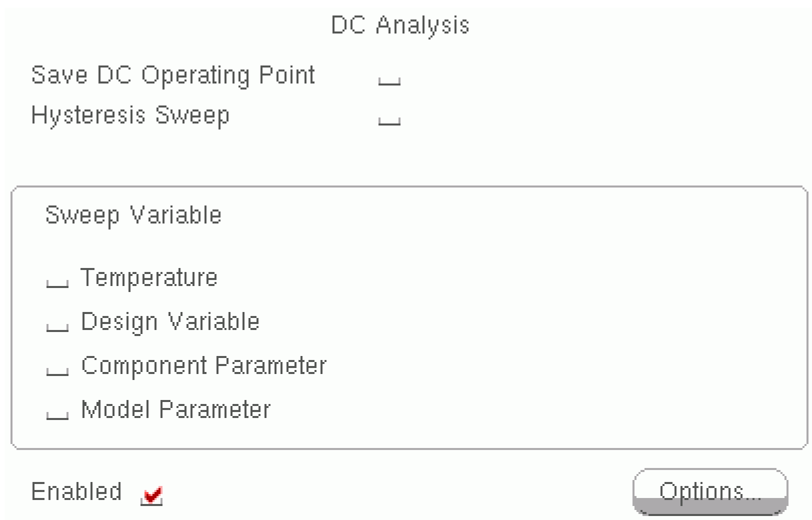
The Test toolbar is available only when the Calculator is opened from ADE L or ADE XL. For more information, see Test Toolbar on page 357.

Schematic Selection Toolbar

Note: The Schematic Selection toolbar is available only when the Calculator is opened from ADE L or ADE XL.

This toolbar displays a set of function buttons, such as *vt* (voltage transient), that you can apply to schematic objects to build expressions. For more information about the Schematic Selection toolbar, see Schematic Selection Toolbar on page 355.

Following are the four cases that describe how *vdc* and *vs* work based on the fields selected in the DC Analysis form in ADE L:



- Case 1—When the *Save DC Operating Point* check box is selected and a Sweep Variable is set, `vdc("node_name")` returns the voltage of the DC operating point, which is a scalar value, and `vs("node_name")` returns a waveform of the DC sweep.
- Case 2 and 3—When the *Save DC Operating Point* check box is enabled or disabled and no variable is swept, `vs("node_name")` returns the voltage of the DC operating point, which is a scalar value, and `vdc("node_name")` also returns a scalar value.
- Case 4—When *Save DC Operating Point* check box is disabled and a Sweep Variable is set, `vdc("node_name")` returns a waveform of the DC sweep and `vs("node_name")` also returns a waveform.

Selection Toolbar

You can use the Selection toolbar to select a signal from the Results Browser or the graph window. The selected signal is displayed in the Buffer. You then use the tools available on the Selection toolbar to evaluate the selected signal and to define the simulation output format.

The Selection toolbar includes the following options:

Off

If selected, the Calculator does not import the following to the Buffer:

- Signals selected in the Results Browser
- Nets selected in the schematic design
- Traces selected in the waveform graph window

Family

If selected, it prompts you to select a signal from a parametric dataset in the Results Browser or a trace from a set of parametric leaf waveforms. This option is used to build expressions for the entire signal family.

Wave

If selected, it prompts you to select a signal in the Results Browser or a trace in the graph window. This option is used to build expressions for the selected trace.

Clip

If selected, the Calculator works on the visible x-axis range of the trace. When you plot the trace after selecting *Clip*, the graph window displays only the clipped part of the trace. If you want the Calculator to work on the complete trace, clear the *clip* check box before importing the signal to the Buffer.

The *Clip* check box is selected by default. However, you can use the `clipSelectionMode .cdsenv` variable to change the default behavior of the check box.

Note: All these options are selected by default. However, you can set the `signalSelection .cdsenv` variable to disable the options.

 *Evaluate & Plot Expression*

This button is used to evaluate the expression displayed in the Buffer. If the result is scalar, it is displayed in the Buffer. If the result is a waveform, it is plotted in the graph window.

 *Send to Table*

This button is used to display the results of the evaluated expressions in a table.

Plot Destination

You can use this drop-down list box to specify the window used to plot the result. The list has the following destination options:

- *Append* – Adds the result to the selected graph window
- *Replace* – Replaces the graph in the selected graph window (or subwindow) with the result
- *New Subwindow* – Plots the result in a new subwindow in the selected graph window
- *New Window* – Plots the result in a new graph window

Graph Type

You can use this drop-down list box to specify the type of graph to be used to plot the result. The list has the following graph types available:

- *Rectangular*
- *Polar*
- *Impedance*
- *Admittance*

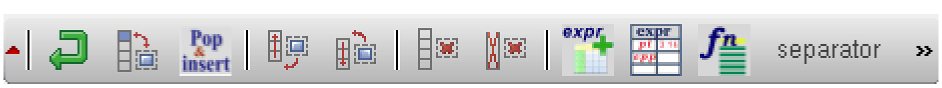
Open Browser

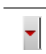
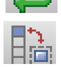
You can use this button to reopen the Results Browser.

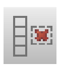
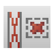
Buffer and Stack Toolbar

This toolbar contains the buttons that help to perform operations on the Buffer and Stack contents. By default this toolbar is displayed below the Buffer and next to the keypad.

The table below describes all the toolbar buttons.



Button	Description
	Displays the Stack assistant
	Pushes the Buffer contents into the Stack
	Swaps the Buffer contents and the first Stack item

Button	Description
	Pops the selected Stack item and appends it to the Buffer contents
	Moves the Buffer contents to the bottom of the Stack and moves the first Stack item to the Buffer
	Moves the Buffer contents to the top of the Stack and moves the last Stack item to the Buffer
	Clears the Buffer
	Clears the Buffer and the Stack
	Adds the expression displayed in the Buffer to the Expression Editor
	Opens the Expression Editor
	Displays the Function Panel
	Undo the last command. The maximum number of commands that can be undone is 8, which is set by the `undoStackSize .cdsenv` variable.
	Redo the last undo command.

Buffer

The Buffer provides an area where you can create or edit expressions that are used to analyze the output data or signals generated after a simulation is run. Buffer is a fixed assistant and is displayed next to the Keypad. When you select a signal from the Results Browser, the signal appears in the Buffer. You can now use this signal to build an expression.

While creating or editing expressions in the Buffer, you can use the Keypad, Stack, Function Panel, and Expression Editor assistants or use the keyboard.

After you have created or evaluated the expressions, you can save the expressions by moving them from the Buffer to the Stack or Expression Editor.

- To push the Buffer contents to the Stack, press *Enter* or select  on the Buffer and Stack toolbar.

- To move the Buffer contents to the Expression Editor, click the  button on the Buffer and Stack toolbar.

After you move the Buffer contents to the Stack or Expression Editor, the Buffer still displays those contents. When you provide a new input to the Buffer to build another expression, the Buffer displays the new entry.

To delete a single character in the Buffer, press the *Backspace* or *Delete* key.

If you want to edit an expression saved in the Stack, move the expressions to the Buffer. When you double-click an expression in the Stack, the expression is moved to the Buffer replacing the previous Buffer contents. You can also drag expressions from Stack or Expression Editor to the Buffer.

Expressions saved in the Expression Editor are edited in the expression table.

The Buffer supports the following features:

- Function Names Auto-Completion on page 319
- Multiline Expressions on page 319
- Color Coding on page 320
- Dependent Expressions on page 321
- Incomplete Parentheses, Quotation Marks, and Expressions on page 321

Function Names Auto-Completion

The Buffer supports auto-completion of function names. When you type the first three letters of a function name in the Buffer, all function names starting with these letters are displayed. You select the required function to complete the function name.

To display a list of functions starting with a particular letter, type the letter in the Buffer and press *Ctrl+E* from the keyboard.

Multiline Expressions

Long expressions that cannot fit in one line in the Buffer can be created as multiline expressions. To create multiline expressions, place the insertion point before the character where you want to start a new line and press *Alt+Enter* from the keyboard.

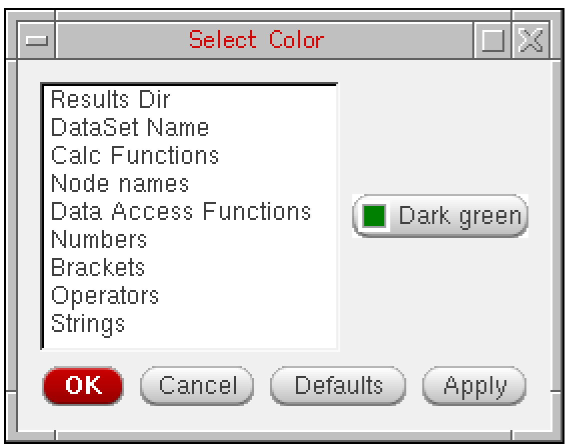
Multiline expressions are moved to the Stack or Expression Editor as a single expression. They are saved and maintained in multiple lines but are treated as a single expression when run.

Color Coding

To differentiate between the various parts of an expression in the Buffer, you can specify different display colors for different parts of the expression, such as results directory, dataset names, functions, node names, data access functions, numbers, brackets, operators, and strings.

1. To set the Buffer color coding, click *Options – Edit Buffer Color*.

   The *Select Color* form appears.



2. Select the category for which you want to set the color.
3. Click the color button on the right to display the available color set.
4. Click the ellipses (...) button to expand the color set.
5. Select the required color.
6. Click *Apply.*

If you do not specify the color settings, the expression is displayed in the default color set.

Dependent Expressions

Dependent expressions are the expressions that contain more than one expression. For example, E2=4+E1, where E1=2+3. In this case E2 is a dependent expression because to evaluate the value of E2, you first need to calculate expression E1.

You can create dependent expressions using the Buffer and can store them in the expression table in the Expression Editor.

Note: To avoid errors, ensure that all expressions required to calculate a dependent expression exist in the expression.

For more information about how to work with dependent expressions, see Building Dependent Expressions on page 341.

Incomplete Parentheses, Quotation Marks, and Expressions

When you build an expression in the Buffer, the Calculator checks for missing parentheses, incomplete quotation marks, and incomplete expressions. For example, when you type the opening (left) parenthesis, it is highlighted in red to indicate that it needs to be closed. After you type the closing (right) parenthesis, the parenthesis pair is highlighted in green or in a color that you have configured for parentheses. If you do not correctly close a pair of parenthesis, the Buffer considers the expression as incomplete.

Similarly, expressions that do not contain the closing quotation mark are also considered incomplete, and the entire string after the opening quotation mark, including the quotation mark, is displayed in red. In addition, expressions ending with an operator are also considered incomplete.

You cannot push incomplete expressions to the Stack or Expression Editor. If you press *Enter* to move the incomplete expression in the Buffer to the Stack, the insertion point is placed in the next line in the Buffer assuming that you are building a multiline expression.

To check if all parenthesis pairs in an expression are complete, place the insertion point after each opening parenthesis and observe the color of the highlight—if the pair is complete, it is highlighted in green.

Note: While an expression is being built in the Buffer, the Buffer background color is set to light grey. This indicates that the expression in the Buffer is not yet complete and cannot be moved to the Stack. After the expression is complete, the background color changes to the default color.

Video

The Using Calculator Buffer - New Features video demonstrates the new features of the Calculator Buffer.

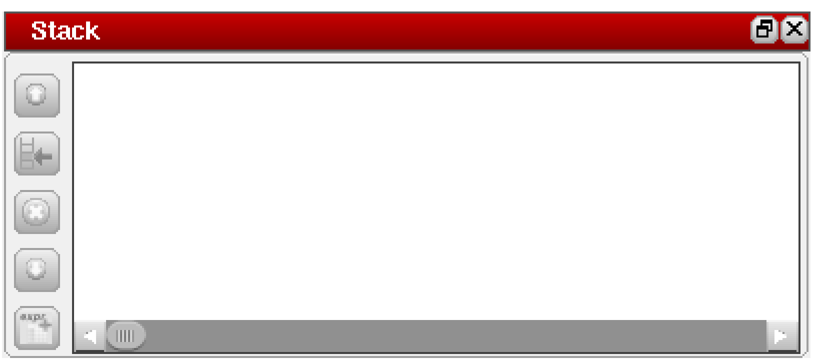
Assistants

Calculator contains the following assistants, which help in building and evaluating expressions:

- Stack on page 322
- Function Panel on page 324
- Expression Editor on page 328
- Keypad on page 330
- Status Bar on page 331

Stack

Stack is an area where you can store the expressions created in the Buffer.



The table below describes the buttons available to the left of the Stack window.

Button	Description
	Moves the selected item(s) up in the Stack.
	Moves the selected item to the Buffer. Alternatively, you can double-click the selected Stack item to move it to the Buffer.

Button	Description
	Deletes the selected item(s) from the Stack. Alternatively, you can also use the *Delete* key to delete the Stack items.
	Moves the selected item(s) down in the Stack.

Pushing Expressions onto the Stack

After you make an entry into the Buffer, you can move the contents to the Stack. To push the current expression onto the Stack click the Enter key or the button.

Displaying the Stack

Click on the Buffer and Stack toolbar to show or hide the Stack.

Clearing Stack and Buffer

There are several ways to clear the Buffer and Stack:

- To remove an item from the Stack without deleting the item from the Buffer, press the *Delete* Key.
- To clear the Buffer without deleting the item from the Stack, click the button.
- To delete an item from both the Buffer and the Stack, click the button.

For more information about the buttons that help you perform Stack and Buffer operations, see the Buffer and Stack Toolbar on page 317.

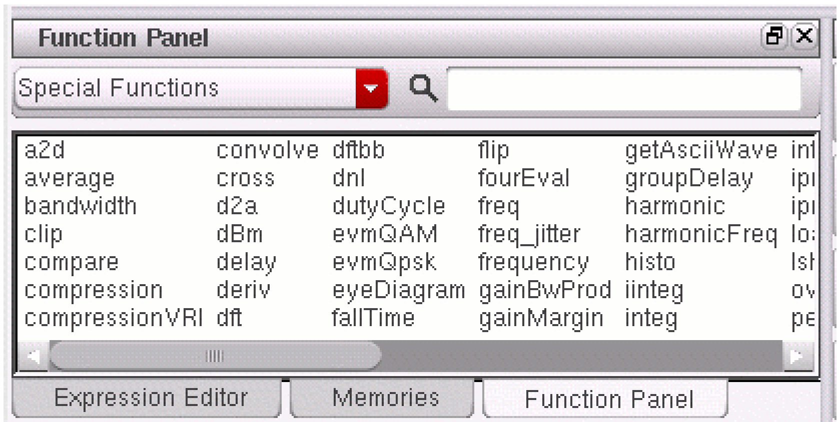
The `showStack .cdsenv` variable determines whether the Stack is displayed by default.

The default number of items stored in the Stack is 8. However, you can set the `StackSize .cdsenv` variable to define the Stack size.

Note: To differentiate between items in the Stack, alternate Stack rows are coded in different colors.

Function Panel

This assistant displays a list of functions. The functions displayed are determined by the function category selected in the Function Panel drop-down list box.



Function Categories in the SKILL Mode

The following table lists the function categories available in the SKILL mode.

Category	**Functions Available**
All	Displays all the available functions
Favorites	Displays the functions that you specify as favorites. To add a function to the *Favorites* list, right-click the required function in the Function Panel and choose *Add to Favorites.*
Math	1/x, 10**x, abs, dB10, dB20, exp, int, ln, log10, sqrt, x**2, and y**x
Modifier	db10, db20, imag, mag, phase, real, phaseDeg, phaseDegUnwrapped, phaseRad, phaseRadUnwrapped, and conjugate
RF Functions	Rn, b1f, ga, gac_freq, gac_gain, gmax, gmin, gmsg, gp, gpc_freq, gpc_gain, gt, gumx, kf, lsb, nc_freq, nc_gain, nf, nfmin, rn, s11, s12, s21, s22, and ssb

Category	Functions Available
Special Functions	average, bandwitdth, clip, compare, compression, compressionVRI, convolve, cross, dBm, delay, deriv, dft, dftbb, dnl, dutyCyle, evmQAM, evmQpsk, eyeDiagram, falltime, flip, fourEval, freq, freq_jitter, frequency, gainBwProd, gainMargin, getAsciiWave, groupDelay, harmonic, harmonicFreq, histogram2D, iinteg, integ, intersect, ipn, ipnVRI, lshift, overshoot, peak, peakToPeak, period_jitter, phaseMargin, phaseNoise,pow, psd, psdbb, pzbode, pzfilter, riseTime, rms, rmsNoise, root, rshift, sample, settlingTime, slewRate, spectralPower, stddev, tangent, thd, unityGainFreq, value, xmax, xmin, xval, ymax, and ymin
Spectre RF Functions	ifreq, ih, it, pir, pmNoise, pn, pvi, pvr, spm, totalNoise, vf, vh, vt, ypm, and zpm
Trigonometric	acos, acosh, asin, asinh, atan, atanh, cos, cosh, sin, sinh, tan, and tanh
AWD Programmed Keys	f1, f2, f3, f4, rf1, rf2, rf3, rf4, rf5, rf6, rf7, and rf8
SKILL User Defined Functions	Includes the functions you define. For more information, see Appendix D, "Defining New SKILL Functions,".
Expressions	Displays the Expression Editor where you can save and build expressions. For more information, seeExpression Editor on page 328.

For more information about the listed above, see Appendix B, "Calculator Functions."

The category displayed by default in the Function Panel is *Special Functions.* You can change the default category by using the `defaultCategory .cdsenv` variable.

Using Functions in the Function Panel

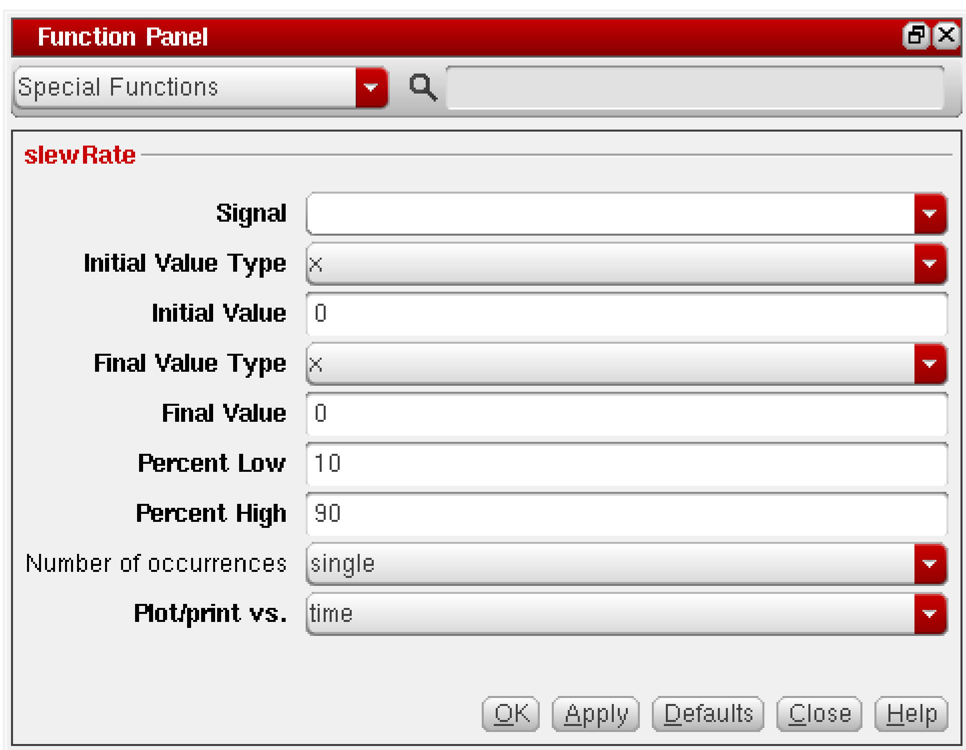
The Function Panel has two types of functions:

1. Single parameter Function
2. Multi-parameter Function

If you select a single-parameter function, it can be applied directly to the Buffer contents. For example, *average* is a single parameter function. When you select this function, it is displayed in the Buffer and the Buffer contents become the function arguments.

If you select a multi-parameter function, the parameter panel for the function appears.

For example, if you select the multi-parameter function, `slewRate`, the various parameters that the function requires appear in the parameter panel, grouped under the function name as shown in the figure below. You can now specify the various parameter values. By default, the parameters contain the values that are set as defaults for that function.



The *Signal* drop-down list maintains a record of all previously selected signals in the current session. It also includes *Buffer* as one of the options. If you select *Buffer*, the Buffer contents are displayed as the *Signal* parameter value.

The default number of signals stored in the drop-down list is *8*. However, you can set the `signalHistorySize .cdsenv` variable to change the number of signals stored.

After you specify the parameter values for the selected function, you can do one of the following:

Action	Description
Click *OK*	Creates the expression, adds the expression to the Buffer, and again displays the list of functions in the Function Panel
Click *Apply*	Creates the expression, adds the expression to the Buffer, and keeps the parameter list for the same function open
Click *Defaults*	Discards the changes that you have made and resets the parameters to the default values
Click Close	Closes the parameter panel and displays the list of functions in the Function Panel
Click *Help*	Displays more information about the selected function

Searching for a Function

You can use the *Search* field in the Function Panel to search for any function.

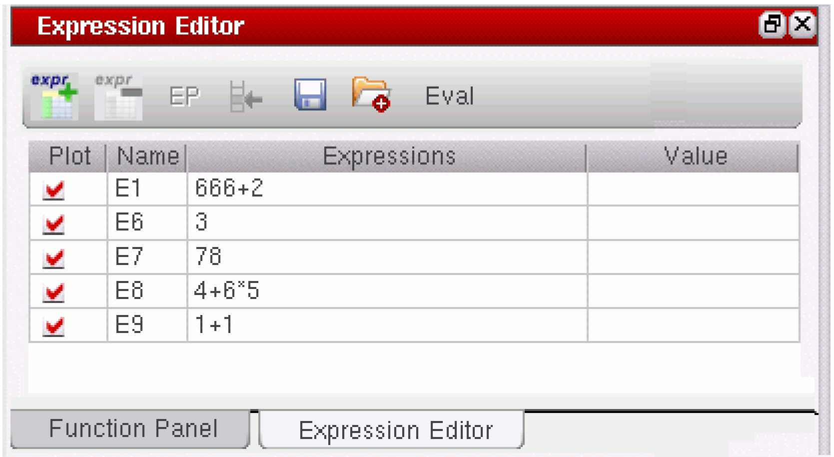
The function search supports category-based search. For example, if you select *Math* in the function category drop-down list and search for a specific function, the search is run only on the *Math* category. To search for a function all the categories, select *All*.

The function search also supports regular-expressions-based search, for example, you can specify *a* to search for the function names that contain the letter *a* or *A* and specify *^a* to search for function names that begin with the letter *a.*

To view the online help for a function, right-click the required function in the Function Panel and choose *Help.*

Expression Editor

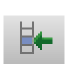
You can move expressions from the Buffer or Stack assistants to the Expression Editor and can store them for future use. The Expression Editor stores expressions in an expression table.



To open Expression Editor, do one of the following:

- ➡ Click the EE button on the Buffer and Stack toolbar.
- ➡ Select *Expressions* from the list of functions categories in the Function Panel.

For information about the buttons in the Expression Editor, see the table below:

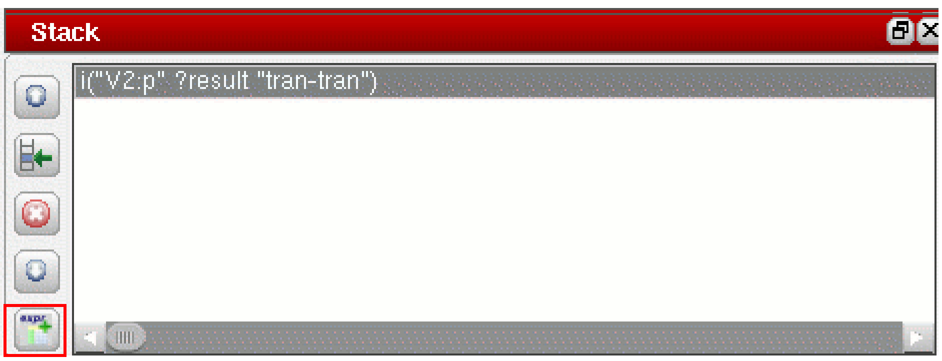
Button	Description
	Adds the expression created in the Buffer to the Expression Editor
	Deletes the selected expressions from the expression table
	Copies the expression selected in the expression table to the Buffer
	Saves the expressions selected in the Expression Editor to a file
	Restores expressions from a file

Button	Description
`Eval`	Evaluates the expression selected in the expression table. **Note:** To cancel the evaluation, press *Ctrl+C*

Adding Expressions to the Expression Editor

To add an expression to the Expression Editor, do one of the following:

- From the Buffer. click the button on the Buffer and Stack toolbar.
- From the Stack, select the required expression and click the *E+* button on the left to move the selected expression to the Expression Editor.



Note: You can also drag expressions from the Stack to the Expression Editor.

When you add an expression to the expression table, the Expression Editor associates a name (signal alias) with the expression—the default expression names are E1, E2, and E3, and so on. If required, you can modify the default expression names. Expression names support alphanumeric values and the first letter of an expression name must be an alphabet.

In Expression Editor, you can also plot circular graphs by selecting the required circular graph type from the *Graph Type* drop-down list on the Selection toolbar.

Editing Expressions in the Expression Editor

To edit an expression in the Expression Editor, double-click the row you want to edit.

Deleting Expressions from the Expression Editor

To delete an expression from the Expression Editor, select the expression you want to delete by selecting the check box associated with the expression and click the E- button or press the *Delete* key.

For more information about how to work with expressions in the Expression Editor, see Working with Expressions on page 332.

Keypad

The Keypad contains buttons for numbers and simple arithmetic functions. If you define any function buttons, they too are displayed on the Keypad.

To define a function button and associate a function with it, type the following command in the CIW:

```
userButton=envSetVal(“viva.calculator” “userButtonX” ‘string
“button_name;function_name)
```

Here,

userButtonX defines button number that you want to map to a function. You can define a maximum of 12 function buttons. Therefore, the value of X can be a number from 1 through 12

button_name is the name of the function button that you want to display on the Keypad.

function_name is the name of the function that you want to associate with the button.

For example, if you want to display the *average* function on the Keypad with the button name *MyAvg*, in the CIW type:

```
userButton=envSetVal(“viva.calculator” “userButton1” ‘string
“MyAvg;average)
```

The defined function button is displayed on the Keypad.

The Keypad is displayed by default. You can change the default behavior by using the `showKeypad .cdsenv` variable.

Status Bar

The status bar displays the status of the action performed in the Calculator window. It also displays warning and error messages. An example is shown in the figure below.

Setting Calculator GUI Using .cdsenv Variables

The variables in the `.cdsenv` file determine the default behavior of the Calculator. Thereafter, the local defaults override the `.cdsenv` variables if `usePreviousGuiSettings .cdsenv` is set to true:

Following are the environment variables that you use to define the settings for Calculator assistants:

- `showKeyPad`
- `showStack`
- `rpnMode`
- `clipSelectionMode`
- `signalSelection`
- `plotStyle`
- `defaultCategory`
- `mathToolBar`
- `TrigToolBar`
- `schematicToolBar`
- `schematicAnalyses`

- `userButton`
- `xLocation`
- `yLocation`
- `width`
- `height`

Note: If `usePreviousGuiSettings` is set to false, the `.cshrc` settings override the local defaults.

Working with Expressions

You can build expressions in the Calculator in the RPN and algebraic modes. After the expressions are created, you can edit, save, load, or delete them using the Expression Editor, Stack, and Buffer.

This section contains the following topics which explain how you can work with expressions:

Selecting Signals and Waveforms to Build Expressions

To build and evaluate an expression, you need to select a signal in the Results Browser or in the graph window. This section describes the following two ways that you can use to select signals:

Selecting Signals from the Results Browser

1. Ensure that the *Family*, `which` is a set of parametric leaf waveforms, or *Wave* option on the Selection toolbar is selected.
2. Choose *Tools – Browser*.

   The Results Browser window appears.
3. In the left panel of the Results Browser, double-click the required results directory.

   The associated datasets appear in the list view to the right.

   Note: You can double-click the directory to expand (display the subdirectories) or collapse (hide the subdirectories) it.
4. Double-click the required dataset.

   The associated signals are displayed.
5. Right-click the required signal and choose *Calculator*.

   The signal is displayed in the Buffer along with the dataset name. For example:

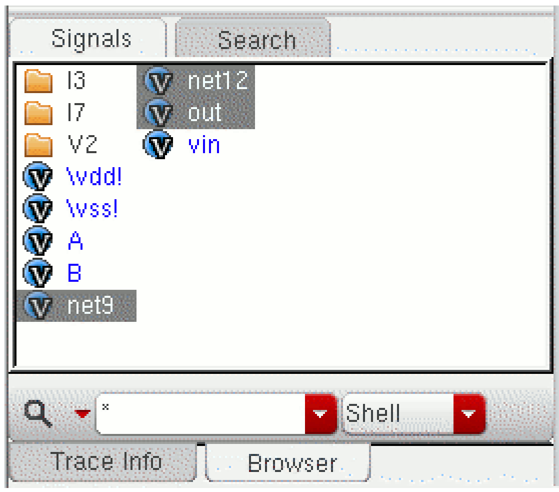
   ```
   v("vin" ?result "tran")
   ```

   Here, `vin` is the name of the signal.

   `?result "tran` is the dataset from which the signal is selected. This `tran` dataset contains the signals generated during the transient analysis of a simulation.

When you send a signal from the Results Browser to the Calculator, the selected signal is displayed in the Buffer. Now, when you send another signal from the Results Browser to the Calculator, the previous signal is pushed onto the stack and the new signal is displayed in the Buffer.

When you select multiple signals in the Result Browser and send them to the Calculator window, the recently selected signal is displayed in the Buffer and all other signals are pushed to the Stack.



For example, if you select three signals in the order—`net9, net12`, and `out`— in the Results Browser (as shown in the figure), the signal `out` is displayed in the buffer and the signals `net9` and `net12` are pushed to the stack.

Selecting Traces from the Graph Window

Perform the following steps to select a trace:

1. Ensure that the *Wave* option on the Selection toolbar is selected.
2. In the graph window, select the trace for the signal you want to add to the Calculator. By default, the Calculator works on the visible X-axis range of a zoomed-in trace. If you want the Calculator to work on the complete trace, deselect the *clip* check box before importing the signal to the Buffer.

   An expression for the selected trace appears in the Buffer.

   - ❑ If the selected trace is a leaf parametric waveform, an expression for the leaf is added to the Buffer.
   - ❑ In the ADE, if the selected trace is the result of an evaluated expression, the SKILL function name for that trace is displayed in the Buffer.

To select a family of traces:

1. Ensure that the *Family* option on the Selection toolbar is selected.

2. In the graph window, select a parametric trace.

   An expression for the parametric family appears in the Buffer.

Building Expressions in the Buffer

You can build expression in the Calculator in the following two modes:

- RPN Mode
- Algebraic Mode

To set the Calculator to work in RPN or Algebraic mode, choose *Options – Mode – RPN Mode/Algebraic Mode*.

This section describes how to build expressions in the RPN and Algebraic modes, how to use functions from the Function Panel while building expressions, and how to add constants or design variables to the expressions.

- Building Expressions in the RPN Mode on page 335
- Building Expressions in the Algebraic Mode on page 336
- Examples of Building Expressions on page 337
- Adding Constants to Expressions on page 341
- Adding Design Variables to the Buffer on page 341

Building Expressions in the RPN Mode

RPN is the default Calculator mode, which can also be set by using the `rpnMode` `.cdsenv` variable.

In the RPN mode, you write expressions without using parenthesis to define priorities for evaluating operators.

To understand this better, consider the following expression:

```
(3 + 5) * (7 - 2)
```

In this expression, the parentheses tell you to first add 3 to 5, then subtract 2 from 7, and finally multiply the two results. In the RPN mode, you list the numbers and operators one after the other to form a stack, instead of using parentheses. For example, the above expression is written as follows in the RPN mode:

```
3 5 + 7 2 - *
```

To build this expression, you perform the following Buffer and Stack operations:

1. Add 3 to the Buffer and press *Enter* to push it to the Stack.
2. Add 5 to the Buffer.
3. Apply the + operation. This pops the topmost item, 3, from the Stack and moves it to the Buffer. The Buffer now has the following expression: 3+5. Now when you press any key from the keypad, this expression is pushed to the Stack.
4. Add 7 to the Buffer and push it to the Stack. The Stack contains: 7, 3+5
5. Add 2 to the Buffer.
6. Apply the - operation. This pops the topmost item, 7, from the Stack and moves it to the Buffer. The Buffer now contains the expression: 7-2.
7. Apply the * operation. This pops the topmost item, 3+5, from the Stack and moves it to the Buffer and the following expression is created in the Buffer: (3+5)*(7-2).

Important

In the RPN mode, if the Buffer contains a complete expression and you click a number button on the Keypad, the expression is pushed to the Stack. However, if you press any key from the keyboard, the character associated with the key is appended to the expression in the Buffer.

Note: It is must to add parenthesis to the expression that you want to fetch from the Buffer.

Building Expressions in the Algebraic Mode

To change the Calculator mode to algebraic, choose *Options – Algebraic Mode*.

In the Algebraic mode you build expressions from left to right. When you click an operator or a function, the operator or function is added to the Buffer to the right of the cursor.

To build an expression in the Algebraic mode:

1. Ensure that the *Wave* button on the Selection toolbar is selected.
2. Select a signal. For more information, see Selecting Signals and Waveforms to Build Expressions on page 332
3. Click the function you want to use.
    - ❑ If you select a single-parameter function, it appears in the Buffer.

- ❑ If you select a multi-parameter function, the parameter panel for the function appears. Enter the required information and click *OK*.

  The expression is displayed in the Buffer. For more information, see Using Functions in the Function Panel on page 325.

4. On the Selection toolbar, click the button to evaluate the expression.

   If the expression evaluates to a scalar, the result appears in the Buffer. If the expression evaluates to a signal, the graph window appears with a trace for the evaluated expression.

Examples of Building Expressions

This section describes how you can build expressions in the RPN and Algebraic mode.

Example of Building Expressions in the RPN Mode

Suppose you want to build an expression in the RPN mode that multiplies two signals:

1. Select the first signal, `signal1`, from the Results Browser. The signal is displayed in the Buffer.
2. Select another signal, `signal2`, from the Results Browser. This pushes `signal1` to the Stack and the Buffer displays `signal2`.
3. Click `*` on the Keypad to multiply `signal1` and `signal2`. This pops `signal1` from the Stack and the following expression is created in the Buffer:

   ```
   signal1*signal2
   ```
4. Click on the Selection toolbar to evaluate the expression. If the result is a scalar, it is displayed in the Buffer. If the result is a waveform, it is plotted in the graph window.

Example of Building Expressions in the Algebraic Mode

To build the same expression in the Algebraic mode, you perform the following steps:

1. Select the first signal, `signal1`, from the Results Browser. The signal is displayed in the Buffer.
2. Press *Enter* to move `signal1` to the Stack.
3. Select another signal, `signal2`, from the Results Browser. The signal is displayed in the Buffer.

4. Click `*` on the Keypad to multiply `signal1` and `signal2`. The Buffer now contains the following expression: `Signal2*`
5. Pop `Signal1` from the Stack. This creates the following expression in the Buffer: `signal2*signal1`
6. Click  on the Selection toolbar to evaluate the expression. If the result is a scalar, it is displayed in the Buffer. If the result is a waveform, it is plotted in the graph window.

Example of Building Expressions Using Functions in Function Panel

Suppose you want to build an expression for measuring the delay between an input and output signal by using the `delay` function. This example assumes you are working with the SKILL Calculator in the RPN mode:

1. Ensure that the *Wave* option on the selection toolbar is selected.
2. In the Results Browser window, right-click the required input signal and choose *Calculator*. The following signal is displayed in the Buffer.

   ```
   v("vin" ?result "tran")
   ```

3. In the Function panel, select the `delay` function from the *Special Functions* category. The parameter panel for `delay` appears.

   In the parameter panel, `v("vin" ?result "tran")`, which is the last selected signal, appears in the *Signal1* field.

4. In the Results Browser, right-click the `out` signal and choose *Calculator*. The following signal appears in the Buffer:

   ```
   v("out" ?result "tran")
   ```

5. In the Function Panel, do the following:

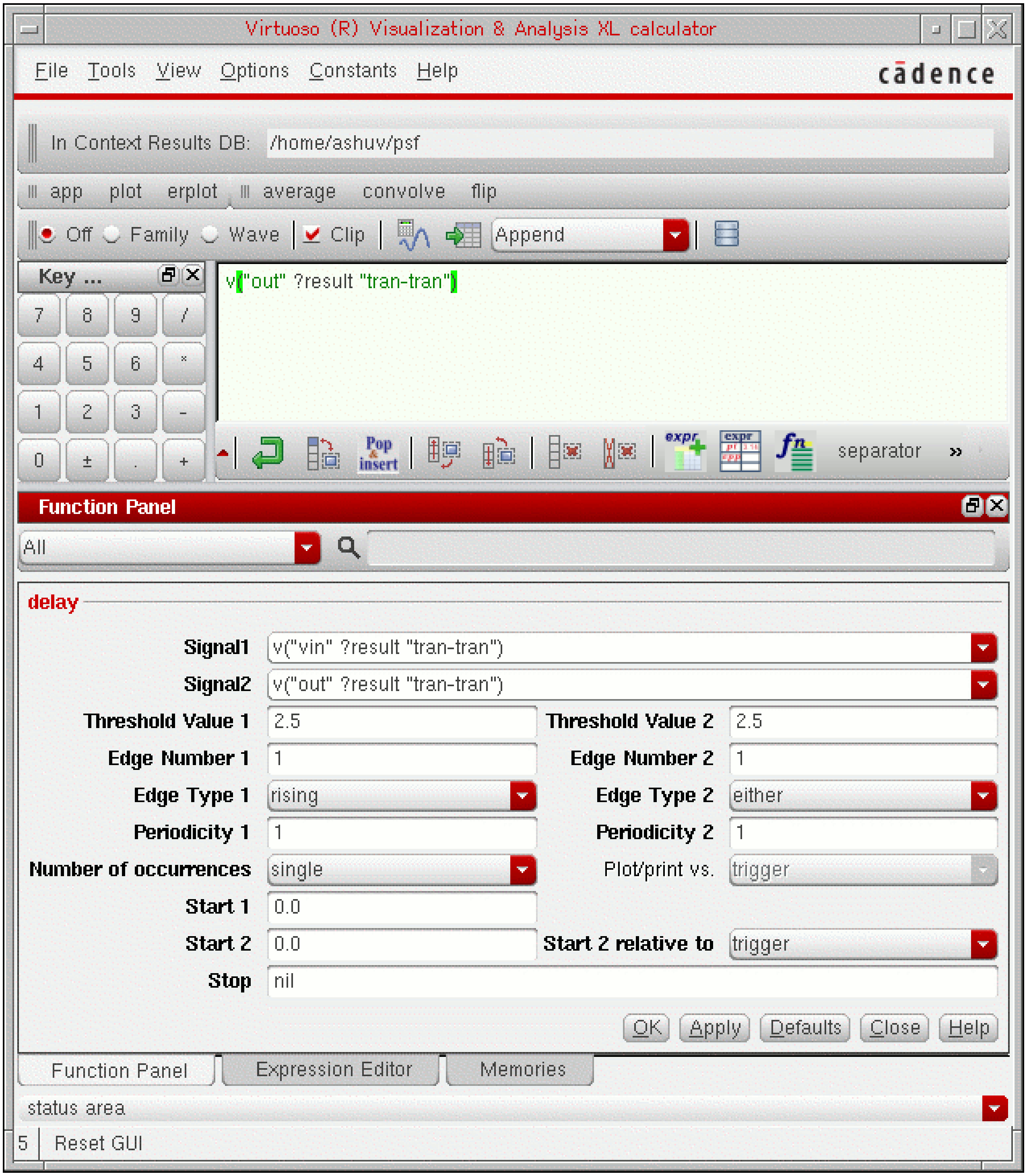
   a. In the *Signal 2* drop-down list, select *Buffer*. The Buffer content, which is the `out` signal that you selected, appears in *Signal2*.

   b. In the *Edge Type1* field, choose *rising*

   c. In the *Edge Type 2* field, choose *rising*.

Ensure that the following values appear in the parameter list:

Field Name	Value	Field Name	Value
Signal1	`vin`	*Edge Type 1*	*rising*
Signal2	`out`	*Threshold Value 2*	`2.5`
Threshold Value 1	`2.5`	*Edge Number 2*	`1`
Edge Number 1	`1`	*Edge Type 2*	*rising*



6. Click *Apply*. The following expression appears in the Buffer:

```
delay(v("vin" ?result "tran") 2.5 1 "rising" v("out" ?result
"tran") 2.5 1 "rising" 0 0 nil nil)
```

7. Click on the Selection toolbar to evaluate the expression. If the result is a scalar, it is displayed in the Buffer. If the result is a waveform, it is plotted in the graph window.

Adding Constants to Expressions

To add a constant to an expression in the Buffer:

➤ From the *Constants* menu, select a constant. The constant appears in the Buffer.

For example, if you want to add the *Boltzmann* constant to a signal displayed in the Buffer, select *Bolzmann* from the *Constants* menu. *Boltzmann* appears in the Buffer and the signal is pushed to the top of the Stack. Apply the + operation. This pops the signal from the top of the Stack and builds the following expression in the Buffer:

```
v("\\vss! " ?result "tran")+boltzmann
```

For more information about constants, see Appendix C, “Constants.”

Adding Design Variables to the Buffer

You can use design variables in the expressions when the Calculator is opened in the SKILL mode. When you open a simulation result in the Results Browser, the Results Browser displays a directory named *variables* that contains design variables for the simulation. To select design variables, do the following:

1. Select the *Wave* option on the Selection toolbar.
2. In the Results Browser, select the required design variable from the *variables* directory.

The selected design variable appears in the Buffer as shown in the figure below.

Building Dependent Expressions

Dependent expressions are the expressions that contain more than one expression.

Suppose you want to compare the clip results of input and output signals that are generated during transient analysis. To create a single expression for this task is a tedious process because the task is complex and requires you to perform multiple operations. Therefore, you can divide the task into subtasks and create separate expressions for each subtask. You can then use the names associated with the expressions for the various subtasks to create the final expression.

1. Select the input and output signals from the Results Browser. The expressions for the signals are displayed in the Buffer. Move the expressions for the signals from the Buffer to the Expression Editor and save them with names *sig1* and *sig2* in the expression table.

   sig1=v("in_p" ?result "tran")

   sig2=v("out" ?result "tran")

2. Next, in the Buffer, create expressions to apply the *clip* function on both the signals, *sig1* and *sig2*. Store the expressions created for the *clip* function in the Expression Editor with names *cl_s1* and *cl_s2*.

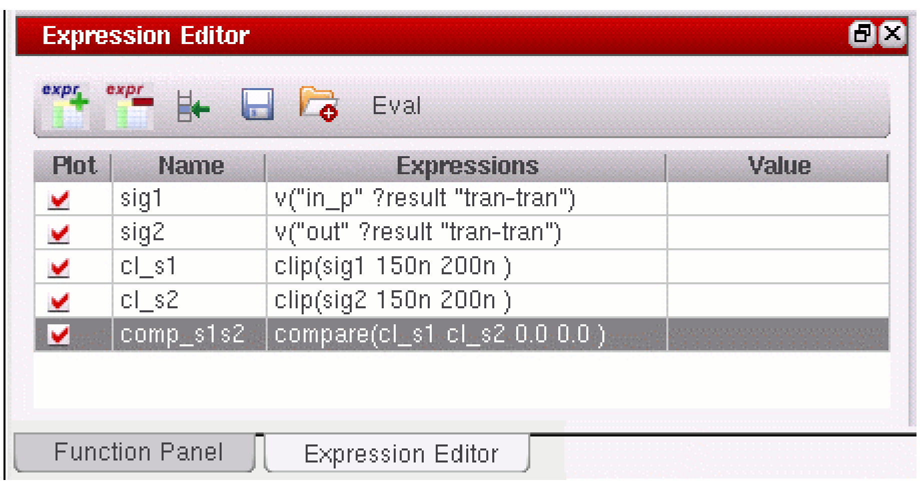
   cl_s1=clip(sig1 150n 200n)

   cl_s2="clip(sig2 150n 200n)

3. Finally, in the Buffer, create an expression to compare the clip results for *sig1* and *sig2* by using the expression names *cl_s1* and *cl_s2*. Move this expression from the Buffer to the Expression Editor and save it with the name *comp_s1s2.*

   comp_s1s2=compare(cl_s1 cl_s2 0.0 0.0)

The expressions created in the expression table for each step are shown in the figure below.



From this example, you can see how dependent expressions simplify the task of creating long and complex expressions.

Note the following when you create dependent expressions:

- Expressions can be created in any order, irrespective of the dependencies. In the above example, it is not necessary to define expression *E1* before expression *E2*.
- An expression can be dependent on multiple expressions. For example, *E2=E1+E3*. Similarly, multiple expressions can be dependent on the same expression. For example, *E2=5*E1* and *E3=2+E1*. Here, both *E2* and *E3* are dependent on E1.
- While creating dependent expressions, ensure that there is no cyclic dependency between the expressions. The following are expressions with cyclic dependency because the expressions are dependent on each other:

  E1=2+E2,

  *E2=5*E1*
- While evaluating or plotting a dependent expression, it is not required to evaluate or plot all expressions on which the dependent expression is based.
- Error messages and warnings, if any, are displayed in the CIW.

By default, the *plot* check box in the expression table is selected for all expressions, indicating that the expressions have already been evaluated and plotted. If the expression is scalar, the result is displayed in the expression table and if the expression is a waveform, the result is plotted in the graph window.

To plot an expression again or to edit an expression, move the expression from the Expression Editor to the Buffer. To do this, deselect the plot check box and click the *Copy Expression to Calculator Buffer* button in the Expression Editor. This moves the expression name (signal alias) to the Buffer. To move the entire expression to the Buffer, drag the expression from the Expression Editor to the Buffer.

To evaluate expressions in the expression table, click the *Eval* button. The corresponding result column displays the results generated. If you do not want to evaluate any expression, clear the corresponding *plot* check box.

Important

Ensure that all expressions that are used in calculating a dependent expression exist in the expression table else an error occurs.

Video

The Building Dependent Expressions Using the Expression Editor video demonstrates how to create and use dependent expressions using the Expression Editor.

Saving Expressions

This section describes how you can save expression created in the Buffer to a file and to the Expression Editor. This section covers the following topics:

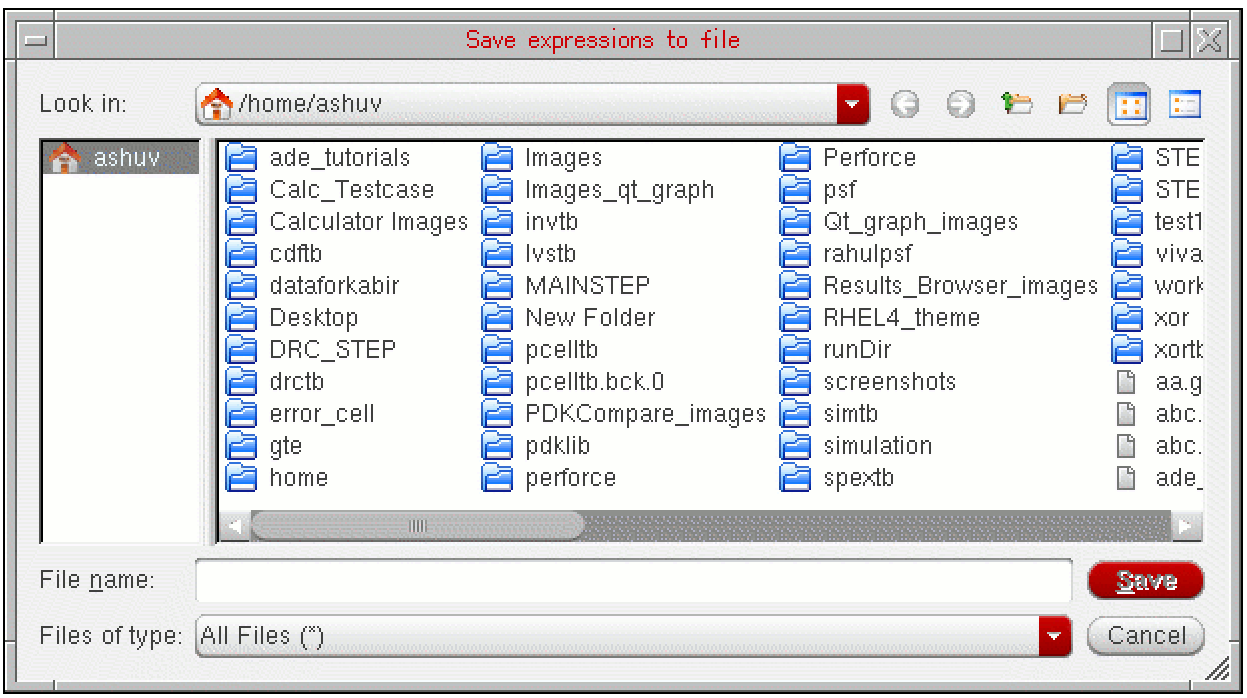
- Saving Expressions to a File on page 344
- Saving Expressions to the Expression Editor on page 345

Saving Expressions to a File

To save an expression in the Expression Editor to a file, perform the following steps:

1. In the Expression Editor, click the  button.

   The *Save expressions to file* form appears.



2. In the *Look in* field, select the directory where you want to save the expression file.
3. Do one of the following:
   - ❑ To overwrite an existing expression file, select that file from the list box below the *Look in* field.
   - ❑ To create a new file, in the *File name* field, type a name for the expression file that you want to create.

4. Click *Save*.

The Virtuoso Visualization and Analysis XL tool saves the expression list to the file.

To automatically save the expressions in the current session to the `defaultVarFileName` file when you close the Calculator or exit the Virtuoso Visualization and Analysis XL tool, set the `writeDefaultVarFileOnExit` `.cdsenv` variable to `true`.

Saving Expressions to the Expression Editor

You can save expressions created in the Buffer to the Expression Editor. To add an expression from the Buffer to the Expression Editor, click the button on the Buffer and Stack toolbar. For more information, see Expression Editor on page 328.

Dragging Expressions across Buffer and Assistants

Calculator supports the following drag-and-drop operations:

- You can drag expressions from the Expression Editor and place them into any text area. If you drag expressions outside the Calculator window without specifying any destination text file, you are prompted to provide a file name and its location. You can create a new text file or you can append expressions to an existing text file.
- You can drag expressions from the Expression Editor to the Buffer.
- You can drag expressions from the Stack to the Expression Editor to add these expressions to the expression table in the Expression Editor. However, you cannot drag expressions from the Expression Editor to add them to the Stack. To drag expressions from the Expression Editor to the Stack, you need to first drag the expressions from the Expression Editor to the Buffer. You can then press *Enter* to push the expressions to the Stack.
- If you drag multiple expressions at a time from the Expression Editor to the Buffer, the first expression is added to the Buffer and the remaining expressions are moved to the Stack.
- When you drag expressions from the Expression Editor or Stack to the Buffer, the expressions are appended to the current expression in the Buffer.

You cannot perform drag-and-drop operation on expressions in the Buffer. To move expressions from the Buffer to the Expression Editor or Stack, use Buffer and Stack toolbar buttons.

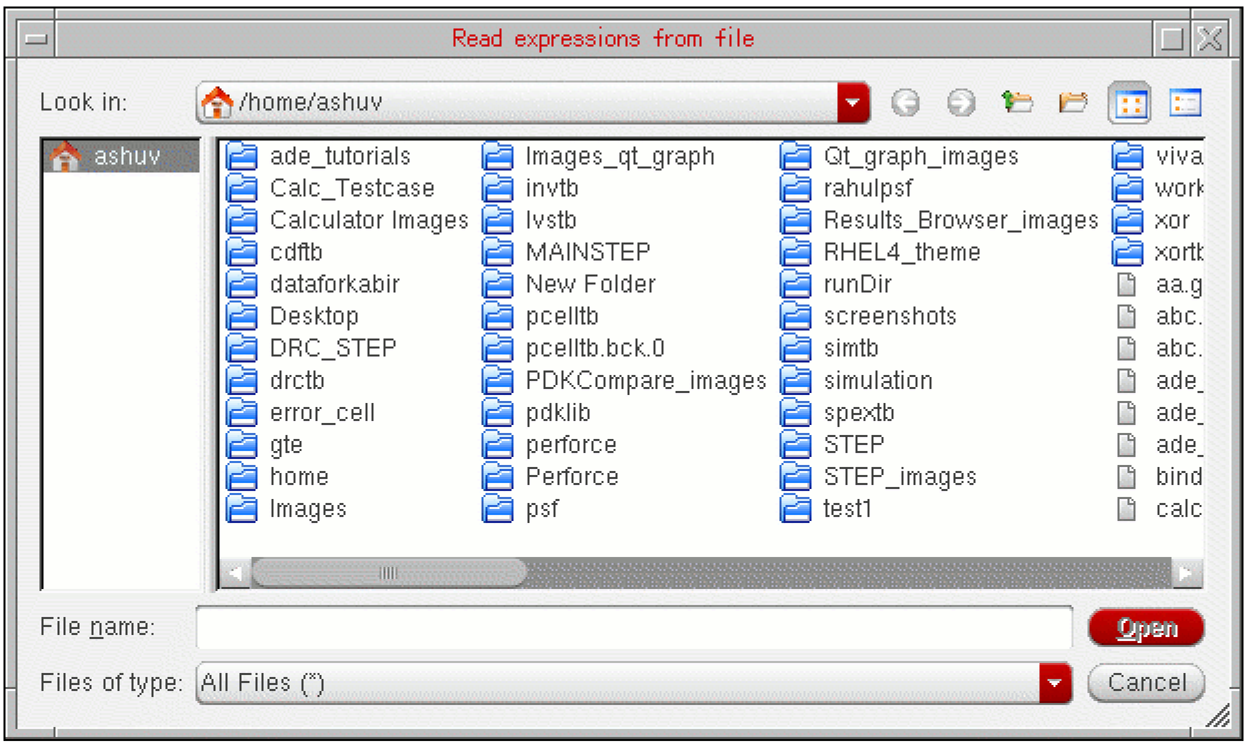
Loading Expressions from an Expression File

You can load an expression file to the expression table in the Expression Editor. The file that you load replaces the current set of expressions in a session, and you lose any unsaved information.

To load an expression file to the Expression Editor, perform the following steps:

1. In the Expression Editor, click  .

   The *Read expressions from file* form appears.



2. In the *Look in* field, select the directory which contains the required expression file.
3. Do one of the following:
   - ❑ From the list box below the *Look in* field, select the expression file you want to open.
   - ❑ In the *File name* field, type the name of the file you want to open.
4. Click *Open*.

   The Virtuoso Visualization and Analysis XL tool loads the specified expression file.

If the `readDefaultVarFileOnStartup` `.cdsenv` variable is set to `true`, The Virtuoso Visualization and Analysis tool automatically loads the `defaultVarFileName` file.

Evaluating and Plotting Expressions

You can evaluate or plot the expressions in the Buffer and the Expression Editor. When you evaluate or plot expressions in the Buffer and if the result is a scalar, it is displayed in the Buffer. If the result is a waveform, it is displayed in the graph window.

When you evaluate or plot expressions in the Expression Editor and if the result is a scalar, it is displayed in the *Value* column in the expression table. If the result is a waveform, it is displayed in the graph window.

To evaluate and plot an expression in the Expression Editor,

1. In the Expression Editor, select the expressions you want to evaluate and plot by selecting the *plot* check box corresponding to the expression.
2. Click the 

button.

   The Calculator evaluates and plots the selected expressions.

Displaying Results in a Table

You can display the results in a table after evaluating expressions.

On the Selection toolbar, you can use the following options to select the destination table where the results are displayed:

- *Append*—Adds the result to an existing table
- *Replace*—Replaces the existing table with the result
- *New Window*—Displays the result in a new table
- *New Subwindow*—Displays the result in a new table

You can display the expressions or results evaluated by the Buffer in a table.

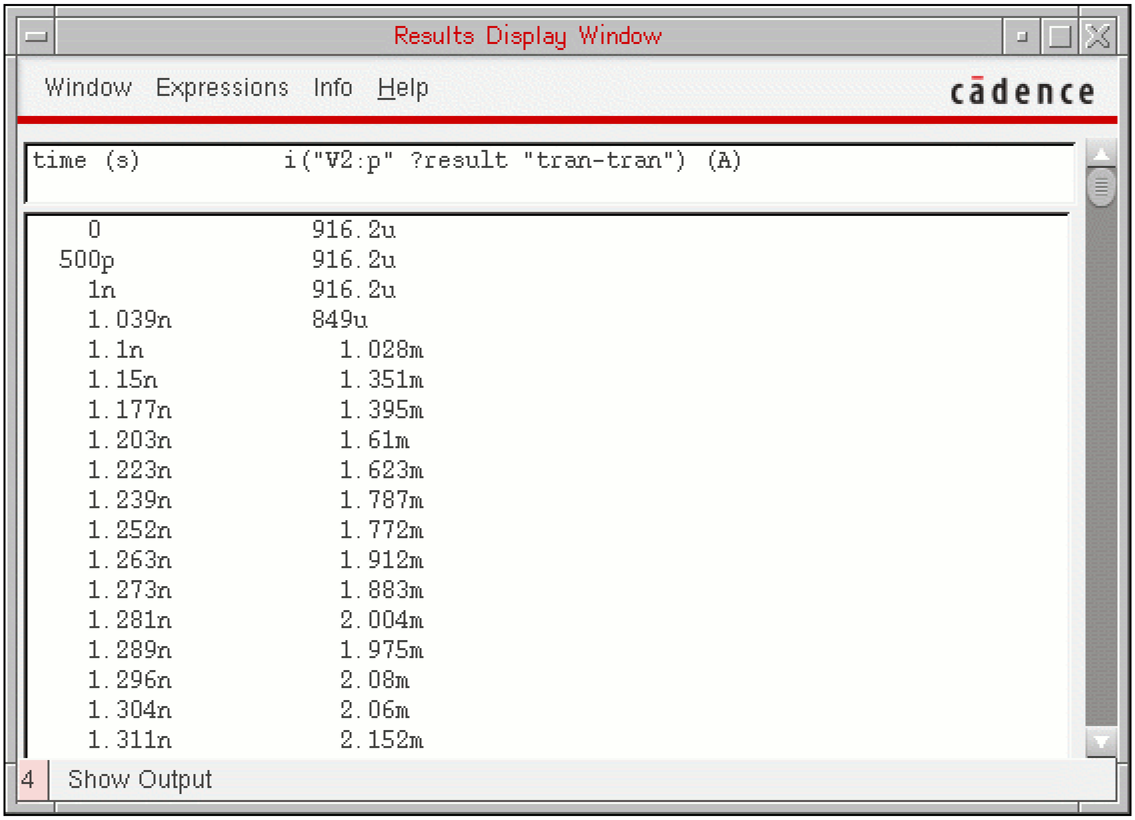
To display results in a table, do one of the following

➡ Click the  button on the Selection toolbar.

The *Results Display Window* appears.

➡ Choose *Tools – Table.*

The *Results Display Window* appears.



Working with the Calculator in ADE

While working in ADE, if you want to analyze simulation results, you can directly build expressions by opening the Virtuoso Visualization and Analysis XL Calculator from the ADE window and build expressions by using the simulation data generated during the latest run.

This section covers the following topics:

- Opening the Calculator using ADE on page 349
- Additional Features in ADE on page 355
- Working with Expressions in ADE on page 358
- Example of Building an Expression in ADE L on page 359

Opening the Calculator using ADE

When you open the Calculator from ADE, it displays some additional GUI components, such as Schematic Selection toolbar, Test toolbar. The Schematic Selection toolbar contains a set of standard functions, such as `vt` for voltage transient, that can be applied on the schematic objects, such as nets, terminals, and flip-flops, to build expressions. The Test toolbar contains the path of the simulation directories available for the current simulation run.

When you select a function on the Schematic Selection toolbar, the schematic design view opens. In the schematic design view, select a schematic object on which you want to perform this function. The complete expression for the selected function and the selected schematic object is created in the Buffer. For more information about how to select a schematic object, see Working with Expressions in ADE on page 358.

While working in the ADE environment, you can open the Calculator by using any of the following two methods:

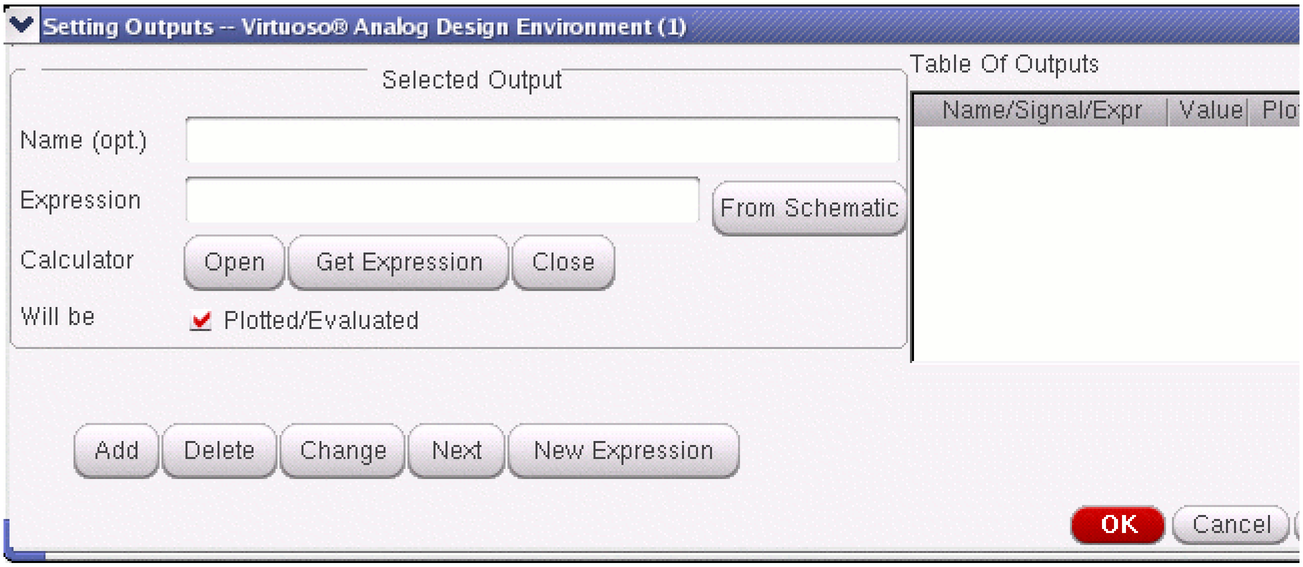
- Opening the Calculator from ADE L on page 349
- Opening the Calculator from ADE XL on page 352

Opening the Calculator from ADE L

To open the Calculator from the ADE L, open ADE L and do one of the following:

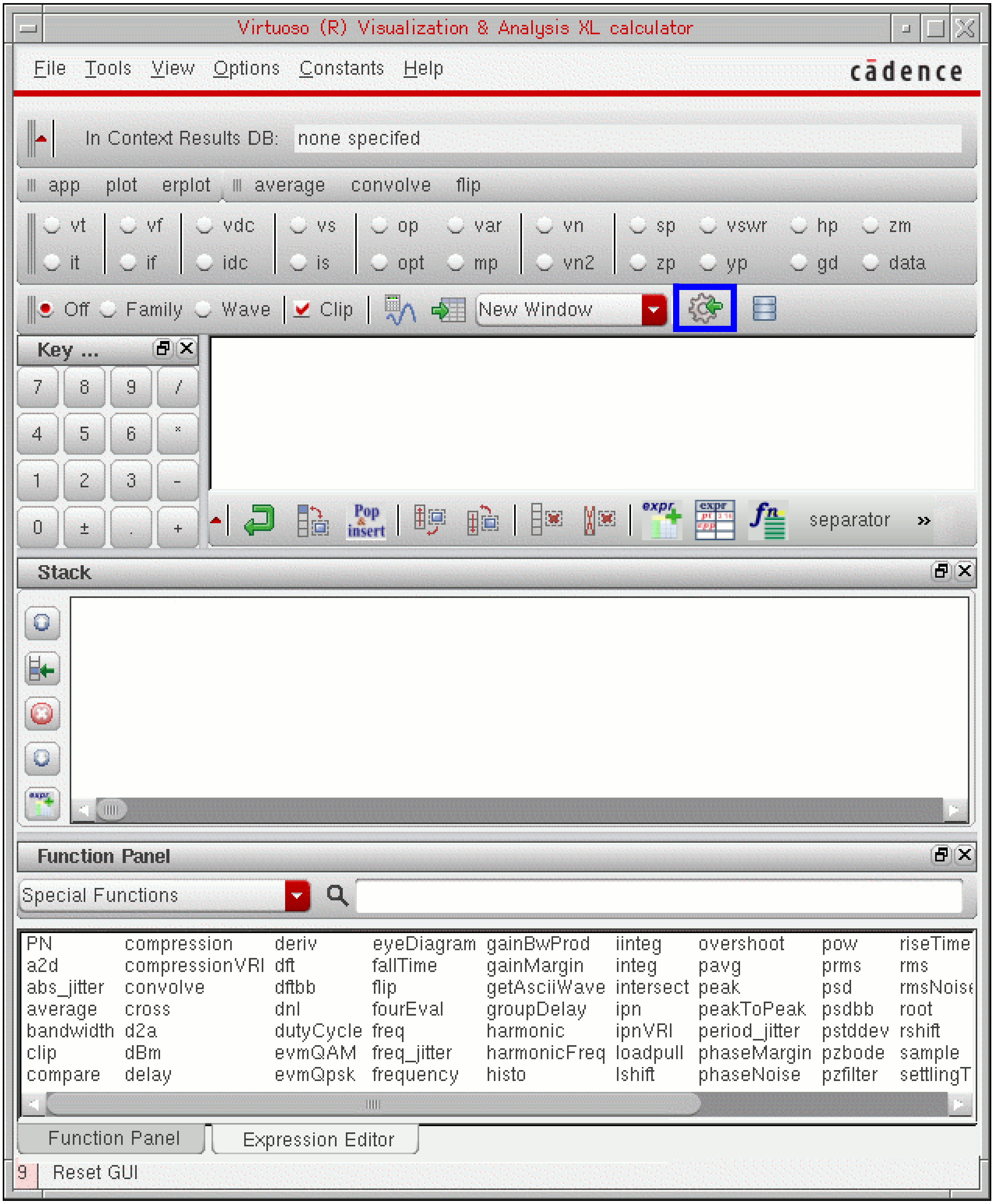
- ➡ Choose *Tools – Calculator*
- ➡ In the ADE L Output Setup window:

a. Right-click anywhere in the *Name/Signal/Expr* column and choose *Edit*. The *Setting Outputs* window appears.



b. Click *Calculator – Open.*

The Calculator window appears. Notice the Schematic Selection toolbar and the Send to ADE button on the Selection toolbar in the figure below, which appear when the Calculator is opened from ADE L.


Virtuoso (R) Visualization & Analysis XL calculator
File Tools View Options Constants Help
cādence
In Context Results DB: none specifed
app plot erplot average convolve flip
vt vf vdc vs op var vn sp vswr hp zm
it if idc is opt mp vn2 zp yp gd data
Off Family Wave Clip
New Window
Key ...
7 8 9 /
4 5 6 *
1 2 3 -
0 ± . +
Pop & insert
separator
Stack
Function Panel
Special Functions
PN compression deriv eyeDiagram gainBwProd iinteg overshoot pow riseTime
a2d compressionVRI dft fallTime gainMargin integ pavg prms rms
abs_jitter convolve dftbb flip getAsciiWave intersect peak psd
average cross dnl fourEval groupDelay ipn peakToPeak psdbb root
bandwidth d2a dutyCycle freq harmonic ipnVRI period_jitter pstddev rshift
clip dBm evmQAM freq_jitter harmonicFreq loadpull phaseMargin pzbode sample
compare delay evmQpsk frequency histo lshift phaseNoise pzfilter
Function Panel
Expression Editor
Reset GUI

To move the expression created in the Buffer to ADE L, click *Get Expression* in the *Setting Outputs* window of ADE L. The expression is displayed in the *Expression* field of the *Setting Outputs* window.

For more information about the Schematic Selection toolbar, see “Schematic Selection Toolbar” on page 314.

For more information about Send to ADE toolbar button, see Send to ADE Button on page 358.

For information about the Calculator menu and toolbar options, see “Using the Calculator Graphical User Interface (GUI)” on page 304.

Opening the Calculator from ADE XL

To open the Calculator in ADE XL, open the ADE XL window and do one of the following:

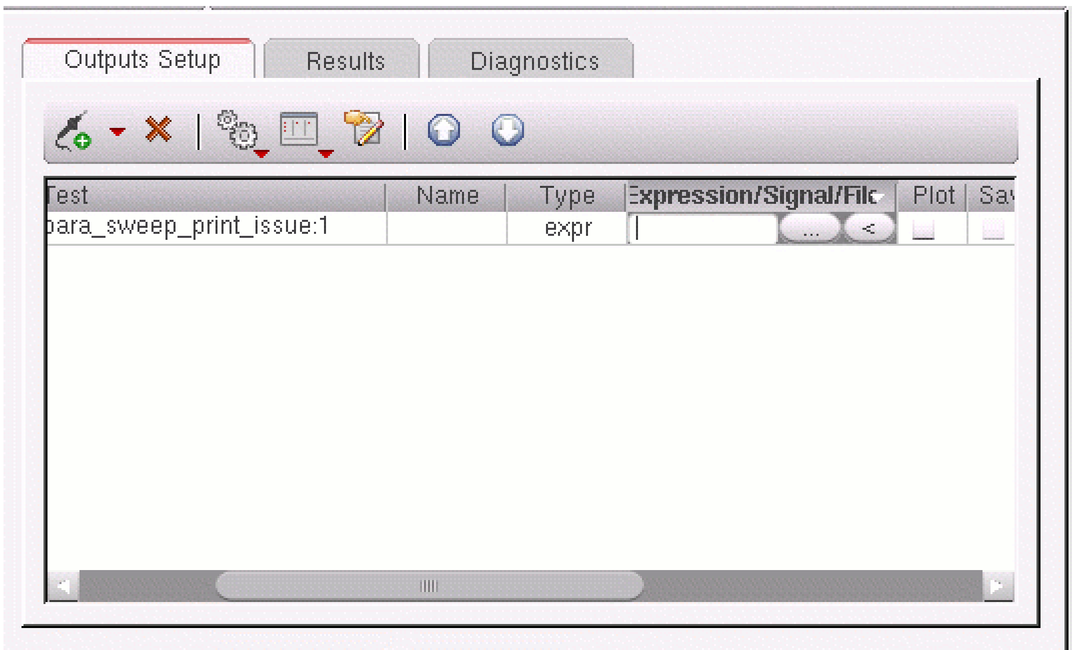
➡ Choose *Tools – Calculator*

Alternatively, you can open the Calculator window by clicking the  button in the ADE XL window.

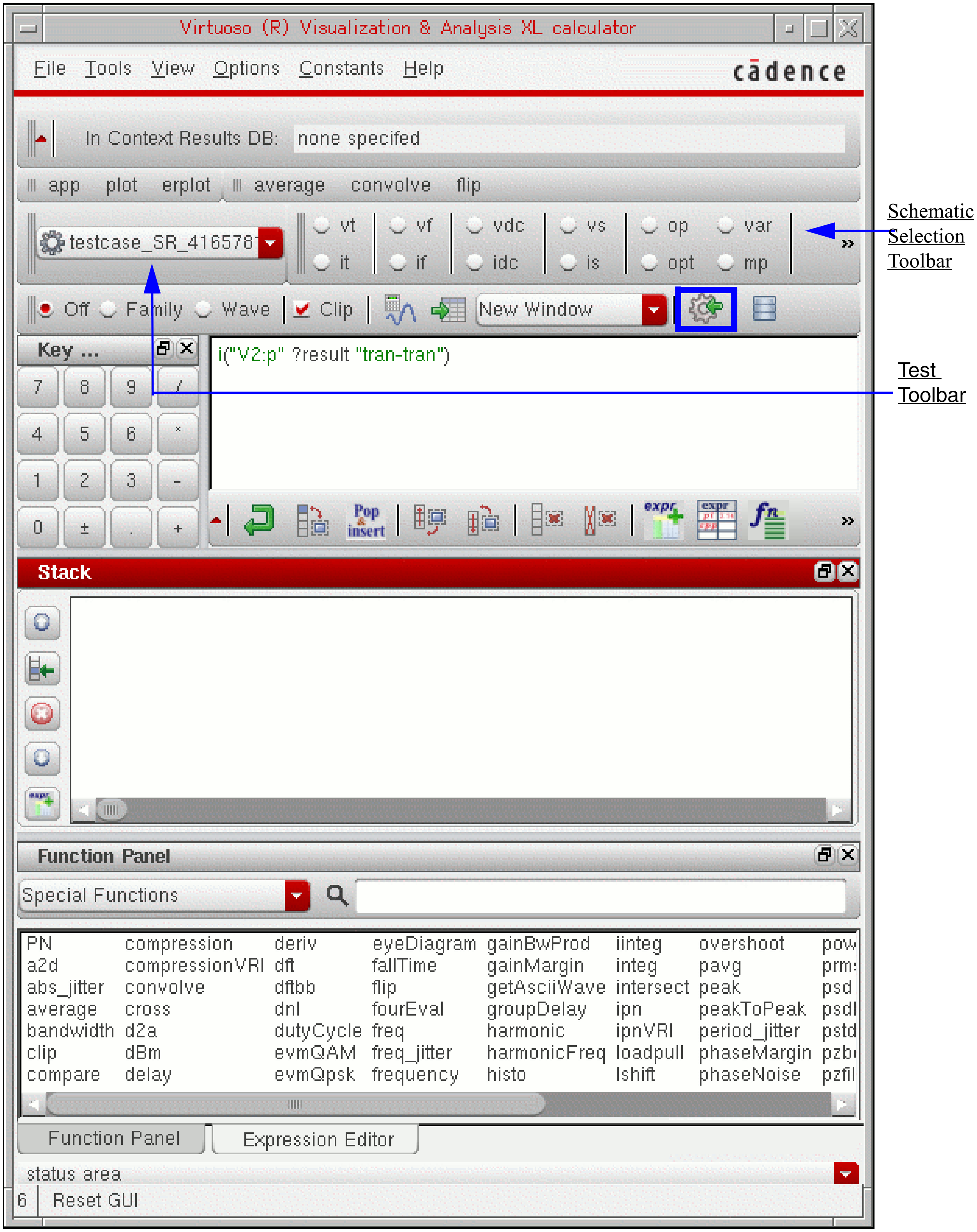
➡ On the *Outputs Setup* tab:

a. Select *Type* as *expr.*

b. Double-click anywhere in the *Expression/Signal/File* column and then click the ellipses (*...*) button.



The Calculator window appears. Notice the Schematic Selection toolbar and the Test directory displayed on the Result toolbar in the figure below. Also, note an additional toolbar button on the Selection toolbar that can be used to send expressions displayed in the Buffer to the ADE XL Outputs Setup. These appear when the Calculator is opened from ADE XL. For more information, see Additional Features in ADE on page 355.


Virtuoso (R) Visualization & Analysis XL calculator
File Tools View Options Constants Help
cadence
In Context Results DB: none specifed
app plot erplot average convolve flip
testcase_SR_416578
vt vf vdc vs op var
it if idc is opt mp
Off Family Wave Clip
New Window
Key ...
i("V2:p" ?result "tran-tran")
Stack
Function Panel
Special Functions
PN compression deriv eyeDiagram gainBwProd iinteg overshoot
a2d compressionVRI dft fallTime gainMargin integ pavg
abs_jitter convolve dftbb flip getAsciiWave intersect peak psd
average cross dnl fourEval groupDelay ipn peakToPeak
bandwidth d2a dutyCycle freq harmonic ipnVRI period_jitter
clip dBm evmQAM freq_jitter harmonicFreq loadpull phaseMargin
compare delay evmQpsk frequency histo lshift phaseNoise
Function Panel
Expression Editor
status area
Reset GUI
Schematic Selection Toolbar
Test Toolbar

After you have created the required expression in the Buffer, you can move the expression to ADE XL for evaluation. To move the expression to ADE XL, in the Calculator window, choose *Tools – Send to ADEXL Test*. The expression is displayed in the *Output Setup* window of ADE XL and is evaluated after you run the simulation.

In ADE XL, you can simultaneously work on multiple test directories in a session. To select a test directory, select it from the *Test* drop-down list in the Calculator window.

For more information about the Schematic Selection toolbar, see Schematic Selection Toolbar on page 355.

For more information about the Calculator menus and toolbar options, see "Using the Calculator Graphical User Interface (GUI)" on page 304.

Additional Features in ADE

When you open the Calculator in ADE, the Calculator GUI displays the following additional features:

- Schematic Selection Toolbar on page 355
- Test Toolbar on page 357
- Send to ADE Button on page 358

Schematic Selection Toolbar

This toolbar displays a set of standard functions that can be applied on the schematic objects to build expressions. When you click a function button on the toolbar, the schematic design view appears. In the schematic design view, select a schematic object on which you want to apply the selected function. When you select the schematic object, an expression is created and displayed in the Buffer.

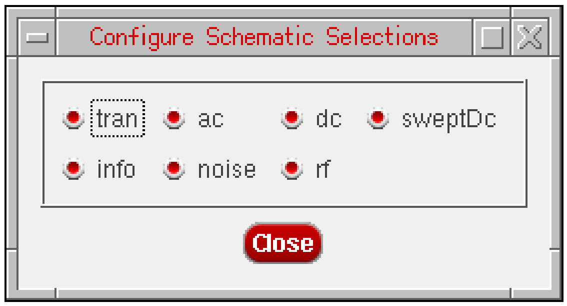
By default, the toolbar displays the function buttons for all simulation analyses types. However, you can configure the Schematic Selection toolbar to display function buttons for the selected analyses types.

To display the function button for a particular analysis type, do one of the following:

➡ Right-click in the toolbar area and choose *Configure selections*.

➡ Choose *View – Configure Schematic Selection Toolbar*.

The *Configure Schematic Selections* form appears. Select the simulation analyses types for which you want to display function buttons on the Schematic Selection toolbar.



Note: You can click the button shown in the Result toolbar to hide the Schematic Selection toolbar.

The following table lists the available function buttons:

Button	**Description**
vt	Transient voltage
it	Transient current
vf	Frequency voltage
if	Frequency current
vdc	DC voltage
idc	DC terminal current
vs	Source sweep voltage
is	Source sweep current (I compared to V graphs)
op	DC operating point
opt	Transient operating point
var	Design variables (ADE design variables). For information about using design variables with Calculator functions, see Adding Design Variables to Expressions on page 358

mp	Model parameter
vn	Noise voltage
vn2	Noise voltage square
sp	Scattering parameters
zp	Impedance parameters
vswr	Voltage standing wave ratio
yp	Admittance parameters
hp	H-parameters
gd	Group delay
zm	Input impedance if all other ports are matched
data	Plots a previous analysis

The Schematic Selection toolbar can also be displayed using the `schematicToolbar .cdsenv` variable.

The function buttons for all analyses types are displayed by default which can be changed by setting the `schematicAnalyses .cdsenv` variable.

Test Toolbar

This field is displayed on the left of the Schematic Selection toolbar if you open the Calculator from ADE XL.

The *Test* field lists the different set of results directories available in the current session, which you can use to build expressions. The results directory in which you are working currently is displayed on the *In Context Results DB* field. If you select another test directory from the *Test* list, the *In Context results DB* field displays continues to display the results directory for which the context in ADE XL is set.

The `displayContext.cdsenv` variable controls the display of the *Test* field.

Send to ADE Button

This toolbar button is displayed on the Selection toolbar.

You can use this toolbar button to move expressions that you create in the Buffer by using the simulation data for the current run to the ADE Outputs section for further analysis.

Working with Expressions in ADE

This section describes how to select a schematic object to build an expression and how to evaluate expressions in ADE.

Selecting Schematic Objects

Before you build an expression in the Calculator, you first need to select a schematic object in the schematic design view. After you select the schematic object, the corresponding expression is created in the Buffer.

Perform the following steps to select a schematic object:

1. On the Schematic Selection toolbar, click the required schematic function button.

   The schematic design view appears.

2. Select an object in the schematic design view. The function selected in the Calculator is applied to the selected schematic object and an expression is created in the Buffer.

   For some function, such as *var*, a form with additional options appears before the object is sent to the Buffer.

Adding Design Variables to Expressions

In ADE, you can add design variables from the schematic to the expression that you build in the Calculator.

To add a design variable to the expression, perform the following steps:

1. On the Schematic Selection toolbar, select the *var* option.

   The Virtuoso Schematic Editing window opens and a form named *Select an Instance* appears.

2. Select an instance on the schematic.

   The design variables for the instance are displayed in the *Select an Instance* form.

3. Select the design variable that you want to add to your expression and click *OK*.

   The selected design variable appears in the Buffer as follows:

   `VAR("CAP")`

Evaluating Expressions

You can load the expressions created using the Calculator into the ADE *Outputs Setup* window. The purpose of Virtuoso Visualization and Analysis XL Calculator in ADE is to build expressions for the current simulation run.

When the Calculator is opened from ADE, the expressions are based on the schematic objects; therefore the expressions cannot be evaluated in the Buffer. Instead the expressions are evaluated in ADE immediately after the simulation is run and the results are displayed in the ADE *Outputs Setup* window. You can use the results to analyze the circuit design. For more information about how to import results to the ADE, see Opening the Calculator using ADE on page 349.

Reloading Expressions using ADE

You can reload a saved expression in the Calculator while running simulation in ADE L and XL.

For example, if you run a simulation in ADE and create expressions in the Expression Editor, such as `E0 = VT("/vin")` and `E1 = E0 + 1`, where expression `E1` is dependent on `E0` and evaluate these expressions by clicking the *Eval* button in the Expression Editor. The output signals are plotted in a graph window.

Now, if you run the simulation again with different dataset values, the new values for `E0` and `E1` are generated in the ADE Outputs section. To update these expression values in the Virtuoso Visualization and Analysis XL graph window, choose *File – Reload*. The updated values for `E1` and `E0` are displayed in the graph window.

Example of Building an Expression in ADE L

Suppose you want to build an expression to calculate `dB20` of the magnitude of the frequency voltage of a schematic object, such as `net3`. For this, you perform the following steps:

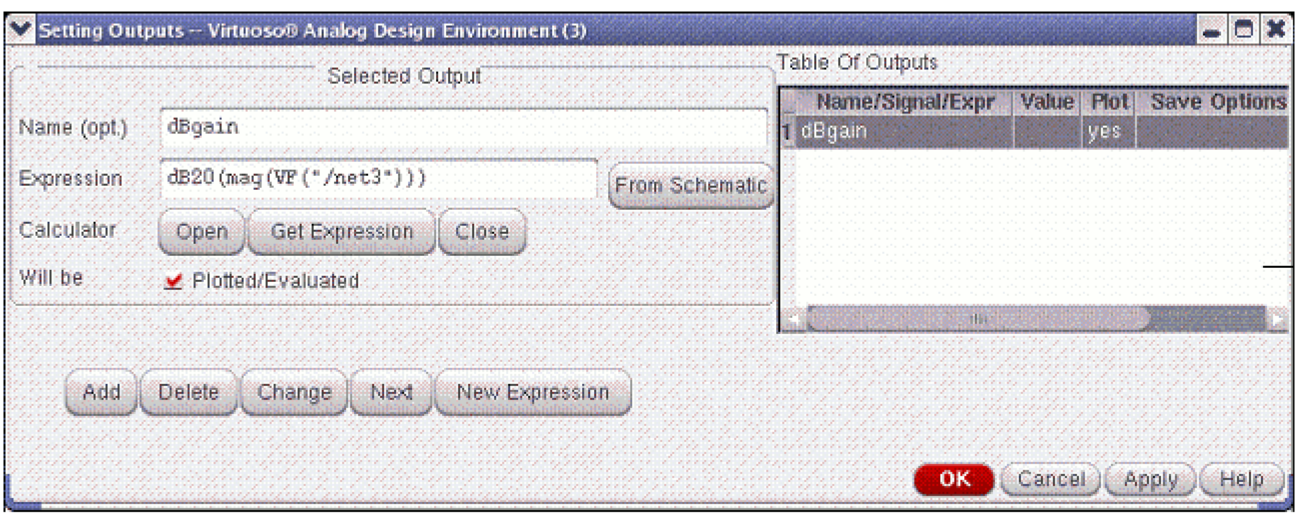
1. In the ADE L *Setting Outputs* window, click *Calculator – Open*. The Virtuoso Visualization and Analysis XL Calculator appears.
2. In the Calculator window, select `vf` on the Schematic Selection toolbar. The schematic design view appears.
3. Select a schematic object, such as `net3`, in the schematic design view. The following expression is displayed in the Buffer:

   ```
   VF("/net3")
   ```
4. To calculate the magnitude, select the single parameter `mag` function from the *ALL* function category in the Function Panel. The Buffer now has the following expression:

   ```
   mag(VF("/net3"))
   ```
5. To calculate `dB20`, select the single parameter `dB20` function in the *ALL* function category in the Function Panel. The expression in the Buffer is updated to:

   ```
   dB20(mag(VF("/net3")))
   ```
6. To move this expression to ADE L, click *Get Expression* in the ADE L *Setting Outputs* window. The expression is displayed in the *Expression* field in this window.

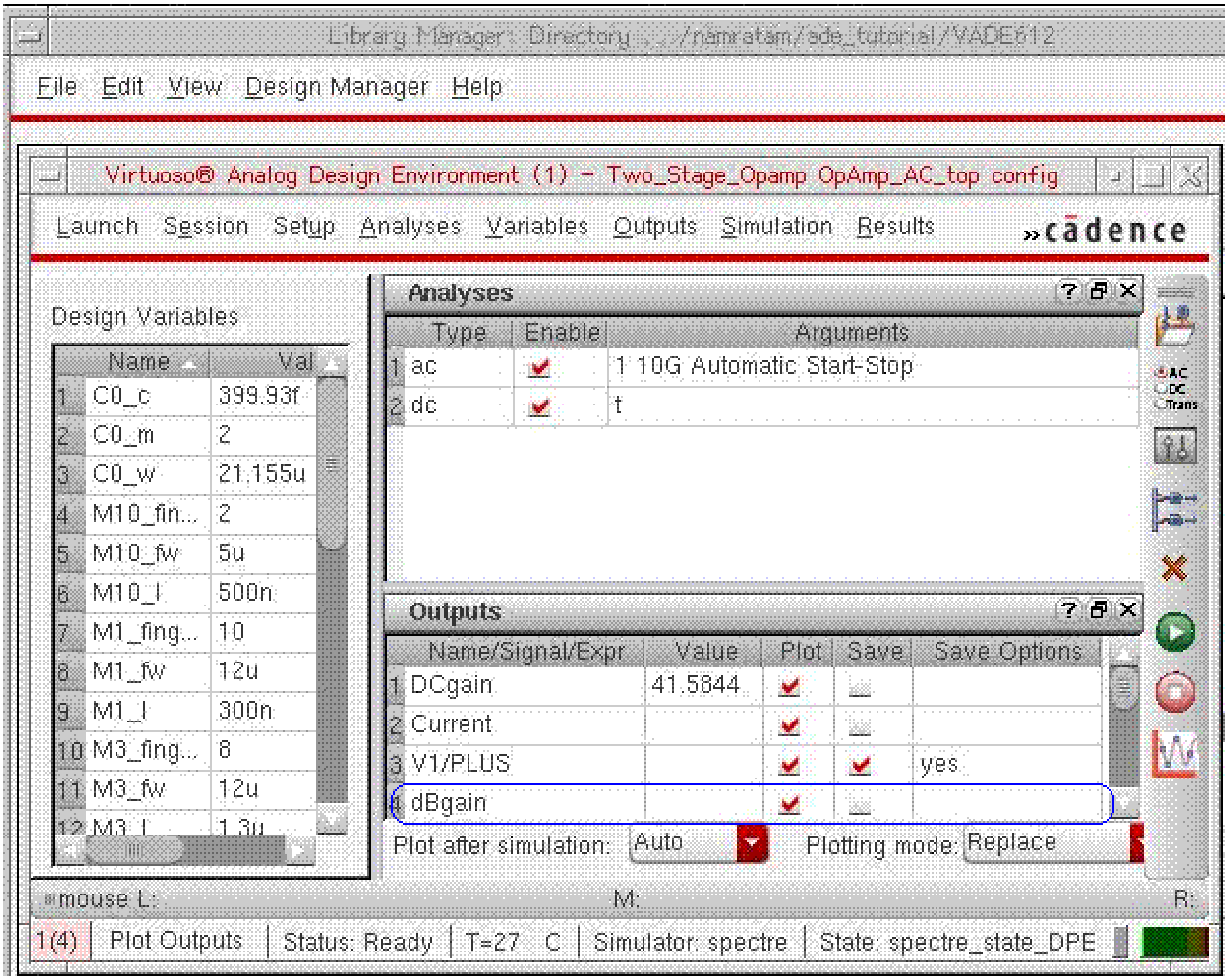
   Note: You can also save the expression in the Expression Editor that you have created. The expression is saved with an expression name. If the expression name already exists in ADE L, a warning message is displayed when you send the expression to ADE L.



7. Enter a name for the expression in *Name (opt)* field and click *Add*. The expression is added to the *Table of Outputs*. The expression is also displayed in the *Outputs* section of the ADE L window.

Note: Follow the same sequence of steps to build an expression in the ADE XL.

You can similarly build more expressions. These expressions are added to the *Outputs* section of the ADE L window. After you run the simulation, all expressions are evaluated and the results are displayed in the *Value* column.



5

Working with Virtuoso Visualization and Analysis XL Table

Virtuoso Visualization and Analysis XL Table (Table) displays data for the selected traces or signals in a table for easy analysis.

This chapter includes the following topics:

Opening Table

You can open Table by using any of the following three methods:

- Opening Table from the Graph Window on page 364
- Opening Table from the Results Browser on page 366
- Opening Table from the Calculator on page 367

Opening Table from the Graph Window

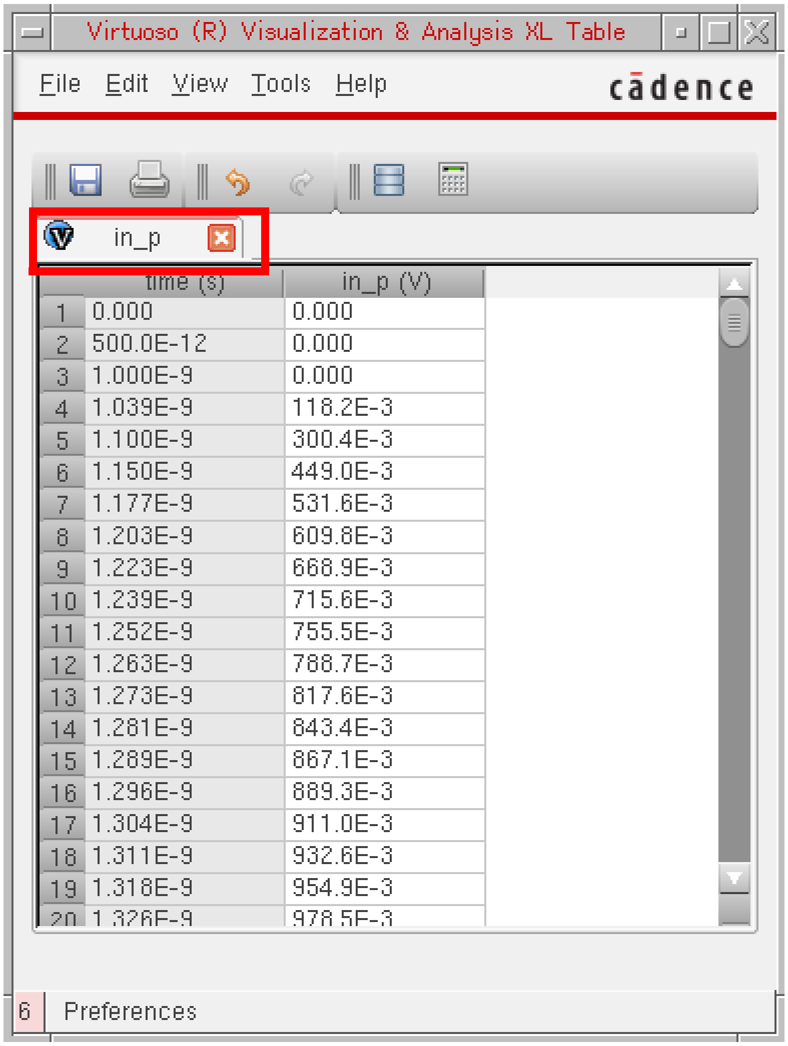
To open Table from the graph window:

- Select the required traces in the trace legend area.
- Right-click a selected trace and choose *Send To – Table – New Window/Append/ Replace*.

  The *Virtuoso (R) Visualization & Analysis XL Table* window opens, as shown in the figure above.

  - If you select the destination as *New Window*, the trace data is displayed in a new table that is displayed on a new tab.
  - If you select the destination as *Append*, the trace data is appended to the table displayed on the active tab.
  - If you select the destination as *Replace*, the Table on the active tab is replaced with a new table.

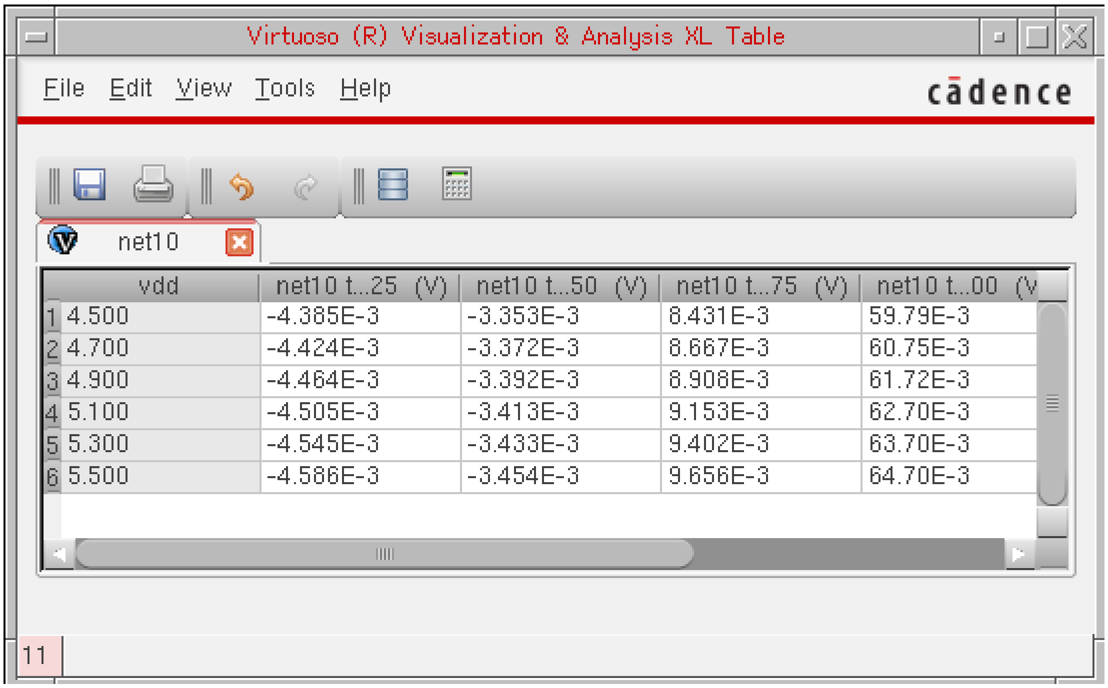
Notice that the tab name contains the names of all the traces for which data is displayed in Table. You can also close the tabs if required.



Note: If the X-axis values of a trace that you are appending to the table does not match the X-axis values of the trace already displayed in the table, the X-axis values for the new trace are added to a separate column. This happens when you display traces from different analyses, such as `AC` (contains frequency on the X-axis) and `tran` (contains time on the X-axis) analyses, in the same table. Parametric, Monte Carlo, and corner data, containing different X-axis values, is also displayed in Table by using the same method.

Viewing Results for Sweep Data

The traces from parametric sweep data display trace data for each sweep value. The figure below displays the trace results for sweep data.

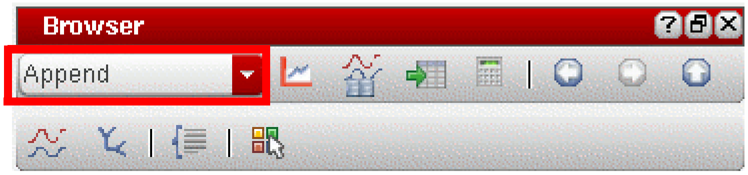


Opening Table from the Results Browser

To open Table from the Results Browser:

- Select the signals for which you want to display the data in Table. To select more than one signal, hold down the `Ctrl` key while you click each signal. After you have selected the signals, do one of the following:
  - Right-click a selected signal and choose *Table*.
  - Click the  button on the Results Browser toolbar.
- From the drop-down list box on the Results Browser toolbar, as shown in the figure below, select the destination table where you want to display the signal data:
  - *Append*—Adds the result to an existing table
  - *Replace*—Replaces the existing table with the result
  - *New Window*—Displays the result in a new table

- ❑ *New Subwindow*—Displays the result in a new table



The *Virtuoso (R) Visualization & Analysis XL Table* window opens with the signal data displayed in a table, a specified.

Note: If the *Virtuoso (R) Visualization & Analysis XL Table* window is already open, the signal data is appended to the table displayed on the active tab.

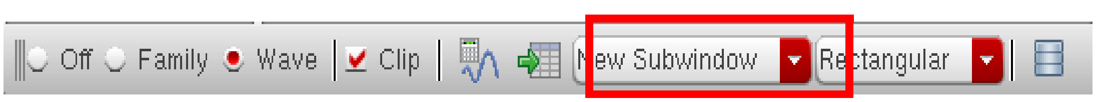
Opening Table from the Calculator

While using the Calculator, you can view in Table the data for the following:

- A signal contained in an expression in the Calculator Buffer
- An output signal obtained after evaluating an expression

To open Table from the Calculator:

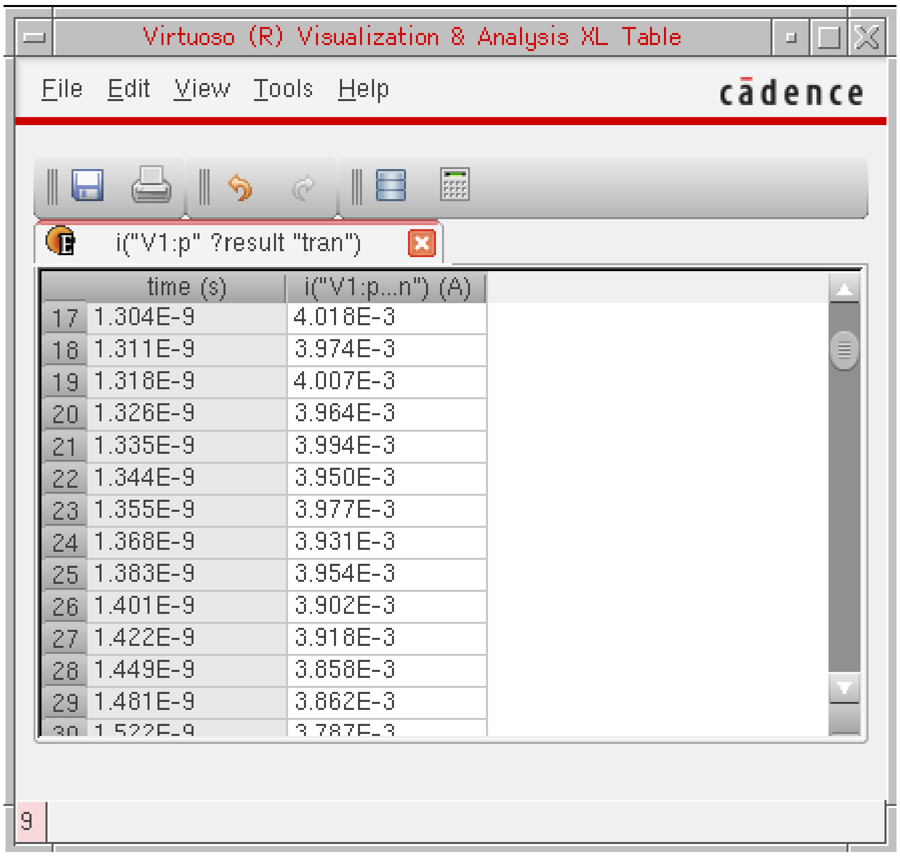
1. From the drop-down list box on the Selection toolbar, as shown in the figure below, select the destination table where you want to display the signal data:
   - ❑ *Append*—Adds the result to an existing table
   - ❑ *Replace*—Replaces the existing table with the result
   - ❑ *New Window*—Displays the result in a new table
   - ❑ *New Subwindow*—Displays the result in a new table



2. To display signals contained in the Calculator Buffer:
   - ❑ Choose *Tools – Table.*
   - ❑ Click the  button from the Selection toolbar.

   The *Virtuoso (R) Visualization & Analysis XL Table* window appears.

Note: To display the signals contained in the Expression Editor in the Table, you first need to move them to the Calculator Buffer.



For more information about Virtuoso Visualization and Analysis XL Calculator, see *Chapter 4, Working with the Calculator* in the *Virtuoso Visualization and Analysis XL User Guide*.

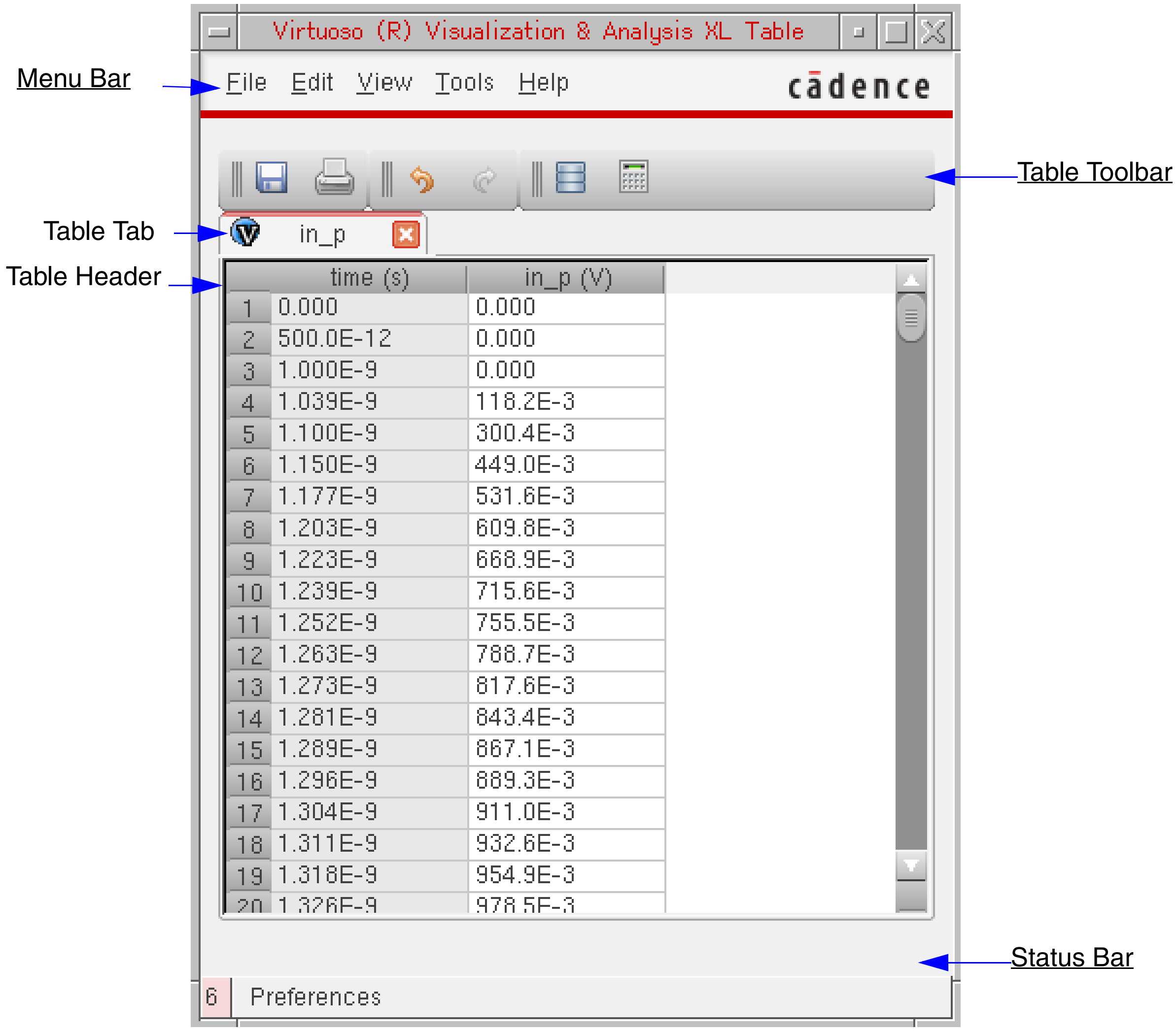
Table Graphical User Interface (GUI)

The Table graphical user interface (GUI) comprises a menu bar, a toolbar, and the tabs on which signal or trace data is displayed.

This section consists of the following topics:

- Menu Bar on page 369
- Table Toolbar on page 371
- Status Bar on page 373

- Shortcut Menu on page 373



Menu Bar

The following menus are available:

- *File*—Includes the following commands:

Command	Description
Save As CSV	Saves the table contents in the CSV format. For more information, see Saving Tables on page 374.

Command	Description
Print	Opens the *Print* form that is used to print the contents of the Table window. For more information, see Printing Tables on page 383.
Close	Closes the Table window.

- *Edit*—Includes the following commands:

Command	Description
Undo	Undoes the operations performed in the selected table.
Redo	Redoes the operations performed in the selected table. This coomand is available only when you perform an undo operation.
Preferences	Specifies the number of operations you want to undo and redo. You can specify upto maximum of 10 operations.

- *View*—Includes the following commands:

Command	Description
Hide	Hides the selected column. For more information, see Hiding and Displaying Columns on page 377.
Reveal	Displays the selected column. For more information, see Hiding and Displaying Columns on page 377.
Format	Sets the format attributes for the selected column. For more information, see Formatting Columns on page 378.
Sort	Sorts the selected column in the ascending or descending order. For more information, see Sorting Table Columns on page 379.
Transpose	Transposes the Table rows and columns. For more information, see Transposing Table Columns and Rows on page 380.

Command	Description
Default Column Order	Sets the default column order, in which the columns that contains X-axis data is displayed to the left of the columns containing the Y-axis data.

- *Tools*—Includes the following commands:

Command	Description
Browser	Opens the Results Browser window
Calculator	Opens the Calculator window

- *Help*—Includes the following commands:

Command	Description
Contents	Displays the *Virtuoso Visualization and Analysis* XL *Table* help
Cadence Online Support	Displays the Cadence customer support website in your default Web browser
Online User Forum (cdnusers.org)	Displays the online users' forum website in your default Web browser
Known Problems and Solutions	Displays *Virtuoso Visualization and Analysis XL Known Problems and Solutions*
What's New	Displays *Virtuoso Visualization and Analysis XL What's New*
About Visualization and Analysis	Displays the version number of the Virtuoso Visualization and Analysis XL

Table Toolbar

The Table has the following toolbars:

- File
- Edit
- Tools

File Toolbar

The following table describes the buttons available on the File toolbar:

Button	Name	Description
	Open CSV File	Opens the CSV file
	Save as CSV	Saves the table contents in the CSV format
	Printer	Prints the contents of the selected Table window

Edit Toolbar

The following table describes the buttons available on the Edit toolbar:

Button	Name	Description
	Undo	Undo the actions performed in the selected Table window
	Redo	Redo the actions performed in the selected Table window.

Tools Toolbar

The following table describes the buttons available on the Tools toolbar:

Button	Name	Description
	Results Browser	Opens the Result Browser as an assistant in the graph window
	Calculator	Opens the Calculator window

To hide or display the toolbars, right-click anywhere on the toolbar or status bar and select the toolbars you want to display.

To view signal names or expression for traces, right-click the Table toolbar or the status bar and choose *Signals/Expressions*. The *Signal/Expressions* window appears on the left, displaying the signal names or expressions for which data is displayed in Table.

Note: You can also use this window to switch between the different tabs.

To rename a Table window tab name, double-click in it and specify a name. Drag the tabs to change their display order. To change the display order of columns, drag the column headers.

Status Bar

The status bar shown at the bottom of the table displays the actions you perform in the Table.

Shortcut Menu

The following commands are available on the shortcut menu that appears when you right-click a column in Table:

Command	Description
Hide Column	Hides the selected column. For more information, see Hiding and Displaying Columns on page 377.
Reveal Column	Shows the hidden columns. For more information, see Hiding and Displaying Columns on page 377.
Format	Sets the format attributes for the selected column. For more information, see Formatting Columns on page 378.
Sort	Sorts the selected column. For more information, see Sorting Table Columns on page 379.
Transpose	Transposes the Table columns and rows. For more information, see Transposing Table Columns and Rows on page 380.

Command	Description
Default Column Order	Sets the default column order, in which columns that contains X-axis data is displayed on the left of the columns containing the Y-axis data.
Change Column Color	Sets the background color of the selected column. For more information, see Changing Column Color on page 380.
Rename Header	Sets the header name of the selected column. For more information, see Renaming Column Headers on page 380.
Show Grid	Shows or hides the grids in the Table.
Copy to ClipBoard	Copies the content of the selected column to clipboard.
Merge X	Combines the selected columns that contain X-axis data. For more information, see Merging Columns on page 380.
Merge All X	Combines all the columns that contain the X-axis data. For more information, see Merging Columns on page 380.
Apply Range	Displays the values that fall in the specific range. For more information, see Filtering Table Data on page 381.
Value At	Displays the value at a particular point in the column. For more information, see Filtering Table Data on page 381.
Sample Values	Displays the rows X-axis values that match the specified set of sample values. For more information, see Filtering Table Data on page 381.
Resize Columns To Contents	Sets the width of all columns to ensure that the contents displayed in them are fully visible. For more information, see Resizing Columns and Rows on page 382.
Reset To Default Cell Size	Restores the default column and row size. This option does not restores the default column order. For more information, see Resizing Columns and Rows on page 382.

Saving Tables

You can save the table contents in the CSV as well as in the HTML format.

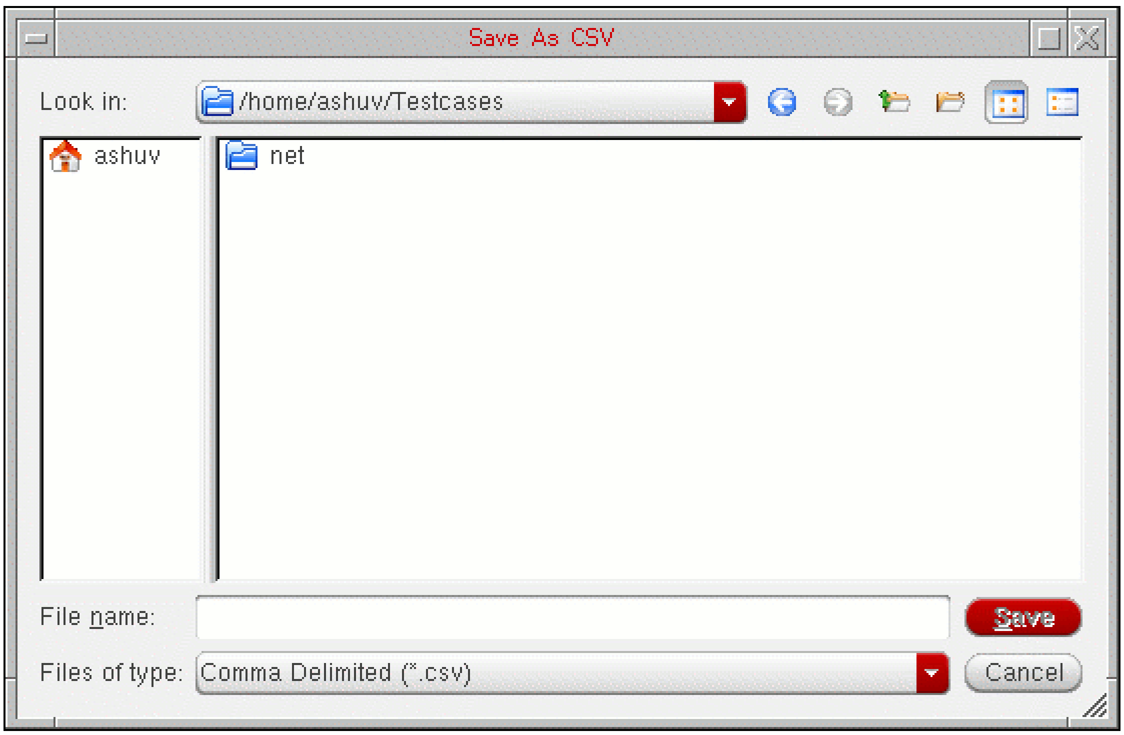
Saving Tables in CSV Format

You can save a table as a text file in the CSV format and import it into spreadsheets, such as Microsoft Excel.

To save a table in CSV format, do one of the following:

- Choose *File – Save As CSV*.

  The *Save As CSV* form appears.



  a. In the *Look in* field, select the directory where you want to save the text file.

  b. In the *File name* field, specify a name for the file to which you want to save the table contents.

  c. In the *Files of type* field, `.csv` is selected by default.

  d. Click *Save*.

  The table is saved as a text file, in the string format. The first line in the text file contains the column headers separated by commas. Each subsequent line contains a table row with the trace data values separated by commas.

Only those values that are displayed in the table are saved to the text file. For example, if you hide a column, and then save the table, the data in the hidden column is not saved in the text file.

The Table tab name is not saved when you save a table in the CSV format.

Selecting Columns

You can select any column in the table by clicking on it. To select multiple columns, hold down the `Control` or `Shift` key and click the columns that you want to select.

To de-select a column, hold down the `Ctrl` key and click a selected column.

To select the entire Table, click the corner button available at the left-most corner of the Table header.

To de-select the Table selection, click the corner button available at the bottom of the scroll bar.

Performing Undo and Redo

You can undo and redo a specific number of actions performed in a selected Table.

To undo an action, do one of the following:

- Choose *Edit – Undo*.
- Click the  button on the Edit toolbar.

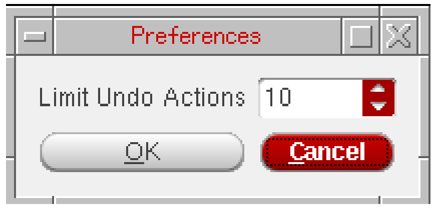
To redo an action, do one of the following:

- Choose *Edit – Redo*.
- Click the  button on the Edit toolbar.

Do the following to specify a limit for the number of actions that you want to undo and redo:

- Choose *Edit – Preferences*.

The *Preferences* form appears.



- In the *Limit Undo Actions* list box, select the a limit upto which you want to undo and redo the actions. The maximum limit that you can specify is 10.
- Click *OK*.

  If you select 3 in this list box, you can undo upto the last 3 actions perfromed in the selected Table.

Note: The redo command becomes avaialble only when you undo the latest action in the Table.

Hiding and Displaying Columns

To hide columns, do one of the following:

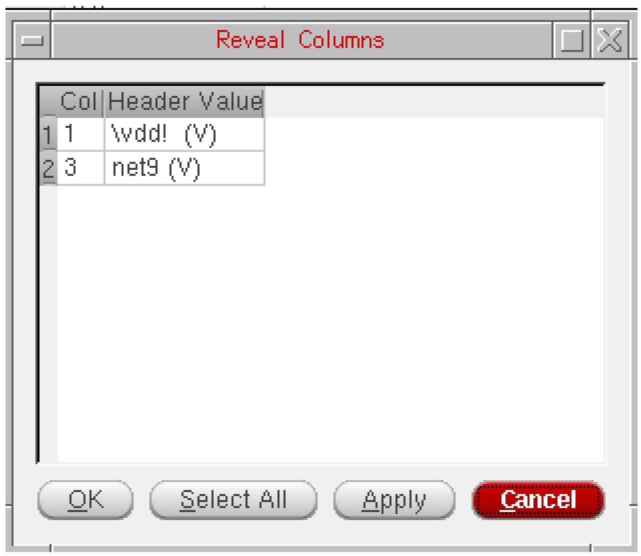
- Select the columns and choose *View – Hide*.
- Right-click a column and choose *Hide Column*.

  The selected columns are hidden.

To display columns that are hidden, do one of the following:

- Select any column and choose *View – Reveal*.
- Right-click any column and choose *Reveal Column*.

The *Reveal Columns* form appears. This form displays a list of all the columns that are hidden in the Table.



- ❑ Select the columns that you want to show. To show all the columns, click Select All.
- ❑ Click *OK*.

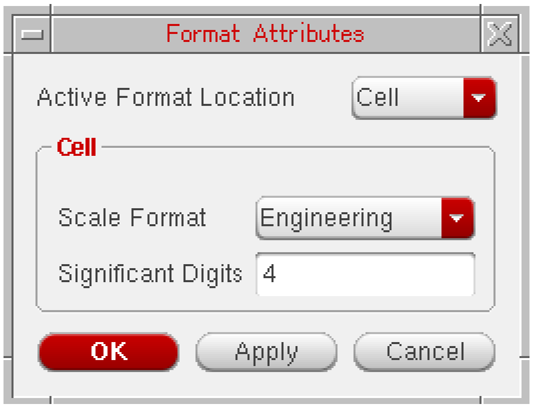
  The columns are displayed in the table.

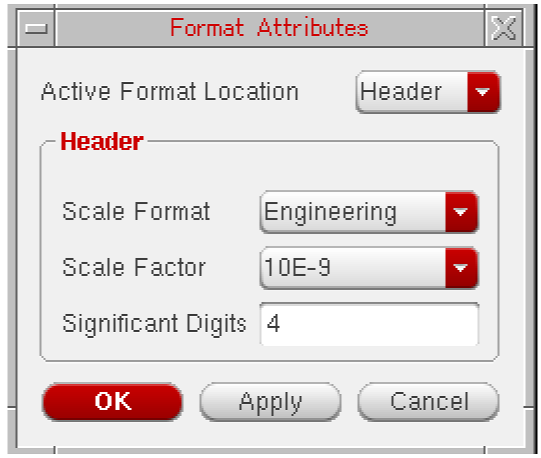
Formatting Columns

To format a column, do one of the following:

- Select a column and choose *View – Format*.
- Right-click a column and choose *Format*.

The *Format Attributes* form appears.





- ❑ In the *Active Format Location* list*,* select the location as `Cell` or `Header`.
- ❑ In the *Scale Format* list*,* select `Scientific`, `Engineering`, or `Suffix`.

Note: The selected value is applied to all the cells in the column.

- ❑ In the *Scale Factor* field, specify the scaling factor, such as`10E-9`.

Note: You need to specify this value only if you select `Header` in step 3.

- ❑ In the *Significant Digit*s field, specify the number of significant digits to be displayed for the data. The maximum number of significant digits that you can specify is `16`.
- ❑ Click *OK*.

Sorting Table Columns

When you sort a column the first time, it is sorted in an ascending order. The next time you sort the same column, it is displayed in the descending order.

To sort the columns in a Table window, do one of the following:

- Select a column and choose *View – Sort*.
- Right-click the column you want to sort and choose *Sort*.

  When the column is sorted, an arrow key appears at the column header. You can press this arrow key to change the sorting order of the column.

Transposing Table Columns and Rows

To transpose Table columns and rows, do one of the following:

- Choose *View – Transpose*.
- Right-click anywhere in the Table and choose *Transpose*.

  The table columns and rows are transposed, which means the column information is displayed in rows and vice versa.

  Note: You cannot format a table after the rows and columns are transposed.

Changing Column Color

To change the background color of a column:

➡ Right-click a column and choose *Change Column Color*.

A drop-down list appears, from which you can choose the color that you want to apply to the background of the selected column.

Renaming Column Headers

Perform the following steps to rename the a column heading:

1. Right-click a column and choose *Rename Header*.

   The *Rename Header* form appears.

2. Type the new name that you want to set as the column header.
3. Click *OK*.

Merging Columns

Perform the following steps to merge two or more columns that contain the similar X-axis data:

1. Select the columns that you want to merge.
2. Right-click the selected columns and choose *Merge X*. This option is available only when you select two or more columns that contain similar X-axis data.

The selected columns are merged into a single column in the Table window. The merged column includes of both sets of X-axis values. The corresponding columns that contain Y-axis data get either interpolated or extrapolated.

To merge all columns that contain similar X-axis data:

➡ Right-click one of these columns and select *Merge All X*.

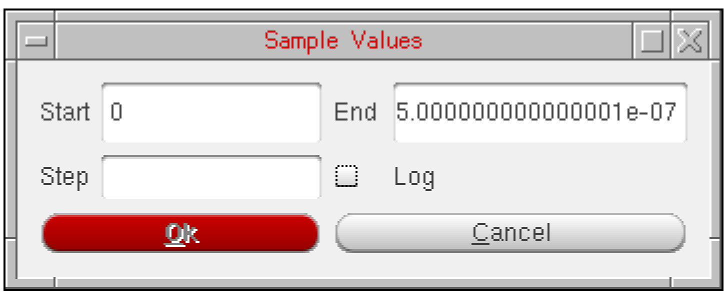
Note: This option is available if you have more than one X-axis data column in the Table window.

Filtering Table Data

To display in the table the data that matches a given set of sample values:

➡ Right-click a column that contains X-axis data and choose *Sample Values*.

The *Sample Values* form appears.



Specify the following values and click *OK*:

- ❑ *Start*—The starting value for samples
- ❑ *End*—The end value for samples
- ❑ *Step*—The step size
- ❑ *Log*—Select this check box to include the logarithmic values

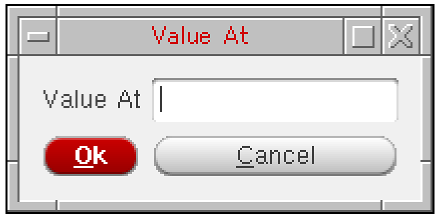
All the X-axis values that match the specified set of sample values are displayed in the table.

Note: Do not specify the step size in the *Step* field when the *Log* check box is selected.

To find the Y-axis value at a particular point:

➡ Right-click a column that contains X-axis data and choose *Value At*.

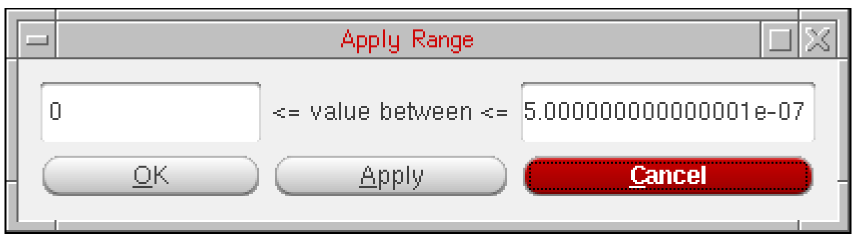
The *Value At* form appears. Specify the X-axis value for which you want to find the Y-axis value.



To display all the rows and columns in a table that fall within the specific range of values:

➡ Right-click a column that contains X-axis data and choose *Apply Range*.

The *Apply Range* form appears. Specify the start and end values for the range.



Displaying Complex Data

To change the format of the complex data:

➡ Right-click the table column and choose *Display Complex As – Real/Imaginary/Real and Imaginary/Magnitude.*

Resizing Columns and Rows

To resize columns:

➡ Drag the right edge of a column header to the left or right to make the column narrower or wider respectively.

To resize rows:

➡ Drag the top or the bottom edges of rows in the first column to make the entire row narrower or wider respectively.

To set the column width equivalent to the data displayed in the column:

➡ Right-click anywhere in the table and choose *Resize Columns To Contents*.

To reset the column width and row height to their default values:

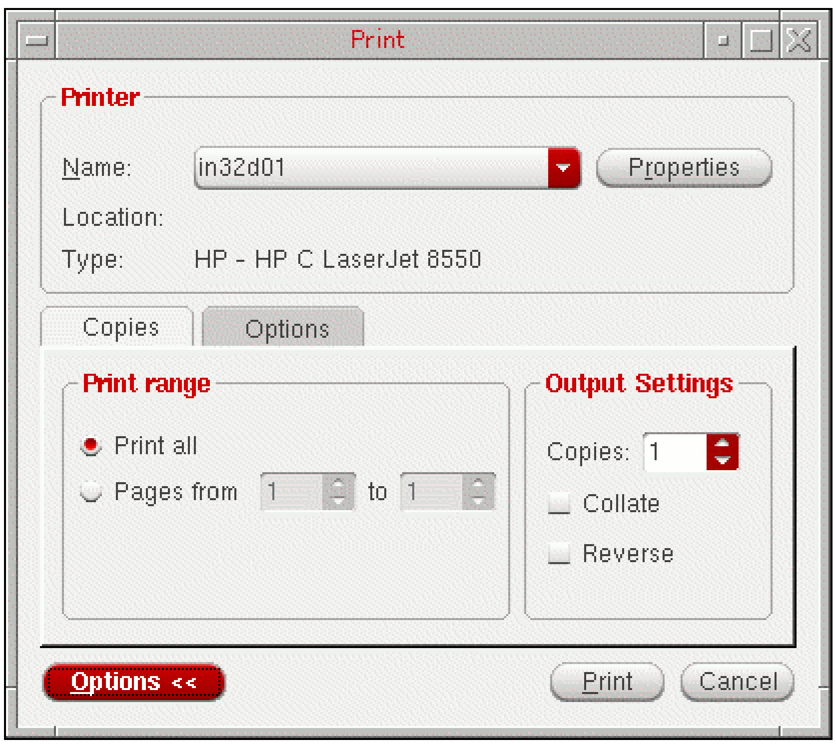
➡ Right-click anywhere in the table and choose *Reset To Default Cell Size*.

Printing Tables

Perform the following steps to print a selected table displayed in the Table window.

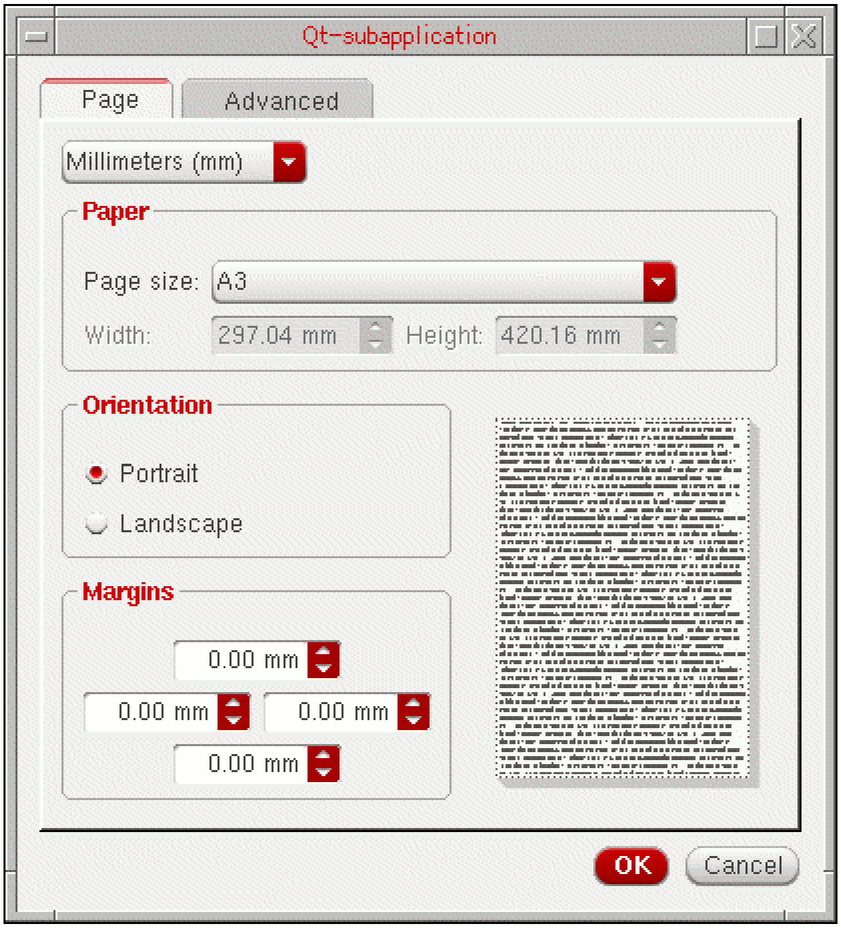
1. In the Table window, do the following
   - ❑ Choose *File – Print*.

   The *Print* form appears.



2. In the *Printer* group box, do the following:
   - ❑ In the *Name* drop-down list box, specify the printer name.
   - ❑ To set properties for the print job, click the *Properties* button next to the *Name* field.

The *Qt-subapplication* form appears. This form has two tabs—*Page* and *Advanced*. The *Advanced* tab is currently not available.



- On the *Page* tab, specify the following fields:
  - Select the units, such as *Centimeters (cm),* or *Millimeters (mm)*, in which you want the paper size to be displayed.
  - In the *Paper* group box, select the required page size in the *Page size* drop-down list box. The *Width* and *Height* fields display the default paper settings for that page size in the specified units.

  Note: The default settings are defined by the `LC_ALL` environment variable on your computer.

  - In the *Orientation* group box, select the print orientation as *Portrait* or *Landscape*.
  - In the *Margin* group box, select the top, left, right, and bottom margins.

  - Click *OK* to save the settings.

Note: If you have specified a printer name in *Destination*, this field becomes disabled.

- Click the *Options* button to view the other printing options. When you click this button, two tabs are displayed in the *Print* form—C*opies* and *Options.*
  - On the *Copies* tab, do the following:
    - In the *Print range* group box, specify the range of pages that you want to print. By default, all pages are printed.
    - On the Output Settings tab, specify the number of copies you want to print. In addition, specify the type in which you want the pages to be printed—*Collate* or *Reverse*.
  - On the *Options* tab, do the following:
    - Specify *Duplex Printing* as *None, Long side*, or *Short side*.
    - Specify *Color Mode* as *Color* or *Grayscale*.
- Click *Print*.

A

Virtuoso Visualization and Analysis XL Tool Environment Variables

The variables and values that specify the basic behavior of the components of the Virtuoso Visualization and Analysis XL are part of the `.cdsenv` file. For information about the order in which the tool reads the `.cdsenv` file, see Creating the .cdsenv File in Chapter 10 of the *Cadence Design FrameworkII User Guide*.

This appendix describes the Virtuoso Visualization and Analysis XL tool variables in the `.cdsenv` file. In each entry, the first column is the tool, the second column is the variable, the third column is the data type, and the fourth column contains the value to be used.

In the SKILL mode, you can use the `envGetVal` and `envSetVal` functions to retrieve and set the `.cdsenv` variables in the CIW. For more information on the CIW or the `envGetVal` and `envSetVal` functions, see Chapter 2 in the *Cadence User Interface SKILL Functions Reference*.

Note: The `envSetVal` settings work for the expressions that are plotted from ADE or from Virtuoso Visualization and Analysis XL when opened in the stand-alone mode. These settings also work for the signals that are plotted when Virtuoso Visualization and Analysis XL is opened in the stand-alone mode. On the other hand, the signals plotted from ADE follow the `display.drf` settings.

The environment variables for Results Browser and Calculator are included in the .cdsenv file at the following location:

```
./tools.lnx86/dfII/samples/wavescan/.cdsenv
```

The environment variables for graph are included in the .cdsenv file at the following location:

```
./tools.lnx86/dfII/samples/viva/.cdsenv
```

This chapter includes the following sections that decsribes the environement variable for Virtuoso Visualization and Analysis XL:

- Graph Variables on page 389

Graph Variables

This section describes the environment variables that are used to set the graph properties:

Font String

The font values for Virtuoso Visualization and Analysis XL is displayed in the following format:

`Default,10,-1,5,50,0,0,0,0,0`

The various fields in the font string are as follows:

<Font family>, <Point size>, <Pixel size>, <Style hint>, <Weight>, <Style>, <Underline>, <Strikeout>, <Fixed pitch>, <Raw mode>

Point size—Specifies point size of the font. If the font size is specified using pixels, it should be -1.

Pixel size—Specifies pixel size of the font. The font size value is -1 if it is specified using pixels. The pixel size makes the font device dependent. If you want to set the size of the font in a device independent You can use Point size to set the size of the font in a device independent manner.

Style hint—Specifies the style hint that is used by the font matching algorithm to find an appropriate default family if a selected font family is not available in the system. Style hints are not supported on X11 since this information is not provided by the window system. The value of '5' represents the enumeration:

QFont—Any Style. Any value other than `5` is ignored on X11 based systems, which means on(i.e., essentially on most unix systems).

Weight—Weighting scale for font rendering, which ranges from 0 to 99. This is similar to the scales used in Windows or CSS. A weight of `0` is ultralight and `99` is black.

Following values can be used for this aspect:

Enumeration	Value
Light	25

Enumeration	Value
Normal	50
DemiBold	63
Bold	75
Black	87

Style—This aspect describes the different styles of glyphs that are used to display text. Following values are supported for this aspect:

Enumeration	Value	Description
StyleNormal	0	Normal glyph used in text with no style.
StyleItalic	1	Italic glyph designed for representing the italicized text.
StyleOblique	2	Italic glyph that are typically based on the graphs with no style. However, these glyphs are not fine-tuned for representing italicized text.

Underline—Set the value to `1` if you want to underline the text. Set the value to `0` if you do not want to underline the text.

Strikeout—Set the value to `1` if you want to strikeout the text. Set the value to 0 if you do not want to strikeout the text.

Fixed pitch—Set the value to `1` if you want to set the for the fixed pitch preference. Set the value to `0` if you do not want to set the fixed pitch.

Raw mode—Set the value to 1 if you want to use raw mode for font name matching. Otherwise, set the value to 0. This option is applicable only for the X11 systems. If the raw mode is enabled, the X font with a complete font name that matches with the family name is searched and all other values set for QFont are ignored. If the font name matches multiple fonts, the first font returned by X window system is used.

Important

When you set the font-related environment variables in the .cdsinit using the `envSetVal` command, you first need to load the `viva` context before setting the variable. For this, you can add the following command in your .cdsinit file:

```
loadContext(strcat(prependInstallPath("etc/context/") "viva.cxt"))
callInitProc("viva")
```

Now, you can use the `envSetVal` command to set the font variable in the .cdsinit file.

Graph Frame Variables

- `viva.graphFrame width string "1000" nil`
- `viva.graphFrame height string "800" nil`
- `viva.graphFrame autoTraceSelect string "true" nil`
- `viva.graphFrame rightMouseZoom string "false" nil`
- `viva.graphFrame selectBySweep string "false" nil`
- `viva.graphFrame useSplitter string "true" nil`
- `viva.graphFrame title string "Window" nil`
- `viva.graphFrame graphMinWidth string "200" nil`
- `viva.graphFrame graphMinHeight string "140" nil`
- `viva.graphFrame graphLayoutType string "Auto" nil`
- `viva.graphFrame useSpacer 'string "true"`

width

Controls the width of the Graph Window.

Syntax

viva.graphFrame width string "*width_pixels***"**

Values

width_pixels	Width of the graph window. Default: Valid values: A positive integer

Example

```
viva.graphFrame width string "1000" nil
```

height

Controls the height of the Graph Window.

Syntax

`viva.graphFrame height string "`*`height_pixels`*`"`

Values

`height_pixels`	Height of the graphwindow. Default: Valid values: A positive integer.

Example

```
viva.graphFrame height string "800" nil
```

autoTraceSelect

Specifies whether the tool selects the trace closest to the system cursor.

Syntax

`viva.graphFrame autoTraceSelect string "true" | "false"`

Values

`true`	The tool automatically selects the trace closest to the system cursor. This is the default value.
`false`	The tool does not select a trace automatically.

Example

```
viva.graphFrame autoTraceSelect string "true" nil
```

rightMouseZoom

Specifies whether you can use the right mouse button to zoom your graph.

Syntax

`viva.graphFrame rightMouseZoom string "true" | "false"`

Values

`true`	Use the right mouse button to zoom your graph.This is the default value.
`false`	Use the middle mouse button to zoom your graph. This enables the right mouse pop-up menu in the Graph Window.

Example

```
viva.graphFrame rightMouseZoom string "false" nil
```

selectBySweep

Specifies whether traces from parametric sweeps are selected by family or by individual leaf.

Syntax

`viva.graphFrame selectBySweep string "true" | "false"`

Values

`true`	The *Trace – Select by Family* command is selected.
`false`	The *Trace – Select by Family* command is not selected. This is the default value.

Example

```
viva.graphFrame selectBySweep string "false" nil
```

useSplitter

Syntax

`viva.graphFrame useSplitter string "true" | "false"`

Values

`true`	This is the default value.
`false`	

Example

```
viva.graphFrame useSplitter string "true" nil
```

title

Specifies the title of the graph window.

`viva.graphFrame title string "`*`window_title`*`"`

Values

`window_title`	The title for the graph window.

Example

```
viva.graphFrame title string "Window" nil
```

graphLayoutType

Specifies the subwindow display layout type of the graph window.

`viva.graphFrame graphLayoutType string "`*`Auto`*`" nil`

Default Value

`Auto`

Valid Values

- `Card`
- `Horizontal`
- `Vertical`

useSpacer

Adds an empty spacer widget to make the number of open subwindows even.

`viva.graphFrame useSpacer 'string "true"`

Default Value

`true`

Valid Values

- `true`—Adds an empty spacer widget.
- `false`—Restores the legacy behavior of not adding an empty spacer widget.

Graph Environment Variables

- `viva.graph titleFont string "Default,10,-1,5,50,0,0,0,0,0" nil`
- `viva.graph subTitle string "" nil`
- `viva.graph displayDate string "true" nil`
- `viva.graph useDefaultTitle string "true" nil`
- `viva.graph defaultSubtitle string "true" nil`
- `viva.graph traceMarkerOn string "true" nil`
- `viva.graph traceMarkerSignificantDigits string "4"`
- `viva.graph snapOn string "snapOff" nil`

  The other valid value for `SnapOn` environment variable is: `snapToData`.
- `viva.graph selectByFamily string "false" nil`
- `viva.graph useCurrentContext string "true" nil`
- `viva.graph colorByParameter string "trace" nil`
- `viva.graph symbolByParameter string "family" nil`
- `viva.graph stripByParameter string "VDD" nil`
- `viva.graph colorBank string "default"`

  You can use this variable to customize the set of trace colors in a Virtuoso Visualization and Analysis XL session. The default value of this environment variable is the default color bank that is specified in the `display.drf` file.

  Note: This variable does not change the colors defined in the `display.drf` file. Also, when you plot simulation results from ADE, the color bank from the `display.drf` file is honored.

  The color names should be separated by a comma, a semicolon, or a space. The color name can be a list of colors defined in the SVG color keyword, such as steelblue, azzure, or the RGB hexadecimal value of the color in the `#RRGGBB` format.

  ```
  viva.graph colorBank string "red; blue; green; yellow"
  ```

  ```
  viva.graph colorBank string "#ff0000, #0000ff, #008000, #ffff00"
  ```

  You can specify any number of colors names using this variable. However, it is recommended that you should not specify more than 18 colors because lot of colors might lead to confusion.

Rectangular Graphs Environment Variables

- `viva.rectGraph foreground string "white" nil`
- `viva.rectGraph background string "black" nil`
- `viva.rectGraph useGradient string "false" nil`
- `viva.rectGraph traceMarkerDisplay string "(%X, %Y)" nil`
- `viva.rectGraph referenceLinesOn string "true" nil`
- `viva.rectGraph enableEdgeMeasurement string "false" nil`
- `viva.rectGraph legendPosition string "left" nil`

  Note: When you specify the `legendPosition` option as `left`, the trace legend is displayed on the left of the graph or strip. To move the trace legend inside the graph, specify `inside`. To move the trace legend to be displayed at the top of each strip, specify `above`.

- `viva.rectGraph showDeltaChildLabels string "false" nil`

  When you set this variable to `false`, the labels are not displayed for the point markers that are part of a delta marker.

- `viva.rectGraph showZoomBar string "false" nil`

  When this variable is set to `false`, the zoom bar is not displayed on graph. To display the zoom bar, set this variable to `true`.

Strip Environment Variables

- `viva.rectGraph stripChartOn 'string "true" nil`

  When you set this variable, all the new signals are plotted in a new strip.

- `viva.rectGraph stripChartOn string "false" nil`
- `viva.rectGraph stripByFamily string "false" nil`
- `viva.rectGraph stripHeight string "50" nil`
- `viva.rectGraph minStripHeight string "32" nil`
- `viva.rectGraph activeStripCue string "Vertical Bar" nil`
- `viva.rectGraph activeStripCueColor string "yellow" nil`

Digital Strip Environment Variables

- `viva.rectGraph digitalStripHeight string "32" nil`
- `viva.rectGraph maxDigitalStripHeight string "40" nil`
- `viva.rectGraph minDigitalStripHeight string "32" nil`

Circular Graph Environment Variables

- `viva.circGraph background string "white" nil`
- `viva.circGraph foreground string "black" nil`
- `viva.circGraph useGradient string "false" nil`
- `viva.circGraph gridTypestring "Polar" nil`
- `viva.circGraph traceMarkerDisplay string "(R=%R, I=%I) (M=%M, P=%P)" nil`
- `viva.circGraph characteristicImpedance string "50" nil`
- `viva.circGraph normalizeSmithValues string "true" nil`
- `viva.circGraph plotToSmithView string "false" nil`

  This variable affects the initial plotting of a Smith Chart. By default, this variable is set to `false`, which means that the Smith chart fits to data when opened for the first time. If you set this variable to `true`, the Smith chart is fitted to the extent of the standard Smith grid. If a compressed Smith grid is required to show data, the initial plot shows the full extent of the compressed Smith regardless of the value of `viva.circGraph.plotToSmithView` variable.

- `viva.circGraph traceMarkerAlwaysVisible string "false" nil`

  The `traceMarkerDisplay` environment variable is used to display values for the tracking marker (trace marker). The following formats are supported to display the trace marker values:

  - `%C` —Displays the real and imaginary cartesian values
  - `%Z`—Displays the impedance values, such as resistance and reactance
  - `%A`—Displays the admittance values, such as conductance and susceptance
  - `%R`—Displays the reflection coefficients, such as mag and angle
  - `%P`—Displays the polar values, such as mag and angle
  - `%F`—Displays the frequency value, which includes the independent axis data

Axis Environment Variables

- `viva.axis majorGridsOn string "true" nil`
- `viva.axis minorGridsOn string "true" nil`
- `viva.axis majorGridForeground string "gray" nil`
- `viva.axis minorGridForeground string "lightGray" nil`
- `viva.axis autoAxisColor string "true" nil`
- `viva.axis foreground string "white" nil`
- `viva.axis background string "black" nil`
- `viva.axis font string "Default,10,-1,5,50,0,0,0,0,0" nil`
- `viva.axis majorTicsOn string "true"nil`
- `viva.axis minorTicsOn string "true" nil`

Dependent Axis Environment Variables

- `viva.depAxis baseAndToplineReferenceHint string "0.0,5.0" nil`
- `viva.depAxis threshold string "20,80" nil`
- `viva.depAxis logScale string "false" nil`
- `viva.depAxis forceOrigin string "false" nil`
- `viva.depAxis notation string "suffix" nil`
- `viva.depAxis showUnits string "true"`

  The `showUnits` variable is used to show or hide the units on the Y-axis

Independent Axis Environment Variables

- `viva.indepAxis logScale string "false" nil`
- `viva.indepAxis forceOrigin string "false" nil`
- `viva.indepAxis notation string "suffix" nil`
- `viva.indepAxis showUnits string "true"`

  The `showUnits` variable is used to show or hide the units on the X-axis.

String Independent Axis Environment Variables

- `viva.stringIndepAxis traceStyle string "points"`

  This variable controls the default line style for X-axis with strings or corners.

Trace Environment Variables

- `viva.trace hiliteColor string "lime" nil`
- `viva.trace useGlow string "false" nil`
- `viva.trace lineThickness string "fine" nil`
- `viva.trace lineStyle string "solid" nil`
- `viva.trace depModifier string "Magnitude" nil`
- `viva.trace indepModifier string "Magnitude" nil`
- `viva.trace symbolsOn string "false" nil`
- `viva.trace symbolStyle string "plus" nil`
- `viva.trace toplineOn string "true" nil`
- `viva.trace midlineOn string "true" nil`
- `viva.trace baselineOn string "true" nil`
- `viva.trace dToAHiVoltage string "5.0" nil`
- `viva.trace dToALoVoltage string "0.0" nil`
- `viva.trace dToAXVoltage string "(vhi + vlo)/2" nil`
- `viva.trace dToAUnit string "V" nil`
- `viva.trace autoReferenceLines string "true" nil`
- `viva.trace baseAndtoplineReferenceHint string "5.0" nil`
- `viva.trace threshold string "20_80" nil`

Trace Legend Environment Variables

- showVisColumn
- font
- printSaveImageFont

showVisColumn

Controls the display of the column containing the visibility buttons for each signal plotted on the graph.

Syntax

```
viva.traceLegend showVisColumn string "true"
```

Default Value

`true`

Valid Values

- `true`—Shows the visibility buttons by default.
- `false`—Hides the visibility buttons by default.

font

Sets the font for the trace legend.

Syntax

```
viva.traceLegend font string "" nil
```

Default Value

The font currently in use.

Valid values

```
viva.traceLegend font string "Default,18,-1,5,55,0,0,0,0,0"
```

For more details, see Font String on page 390.

printSaveImageFont

Sets the font for saving and printing the graph image.

Note: This environment variable does not work if you choose the *Render exactly as screen* option in the *Save Image* form.

Syntax

```
viva.traceLegend printSaveImageFont string "" nil
```

Default Value

The font currently in use.

Valid Values

```
Default,18,-1,5,55,0,0,0,0,0
```

For more details, see Font String on page 390.

Digital Trace Environment Variables

- `viva.digitalTrace foreground string "green" nil`

Digital Bus Trace Environment Variables

- `viva.digitalBusTrace radix string "hex" nil`
- `viva.digitalBusTrace foreground string "green" nil`

Histogram Environment Variables

- `viva.histogramTrace densityEstimator string "true" nil`

  Plots a curve that estimates the distribution concentration. By default, this variable is set to `true`.

- `viva.histogramTrace deviationLines string "true" nil`

  Shows the standard deviation lines in the graph indicating the mean, (mean - standard deviation), and (mean + standard deviation) values. Note that the standard devitaion is a sample standard deviation. By default, this variable is set to `true`.

- `viva.histogramTrace yieldLines string "false" nil`

  Shows the markers associated with the histogram. By default, this variable is set to `false`.

Horizontal Marker Environment Variables

This section describes the environment variables for horizontal marker:

font

Specifies the font of the horizontal marker.

Syntax

```
viva.horizMarker font string "Default,10,-1,5,50,0,0,0,0,0" nil
```

Valid Values

See Font String on page 390.

notation

Specifies the notation of the horizontal marker.

Syntax

```
viva.horizMarker notation string "suffix" nil
```

Default Value

`Suffix`

Valid Values

- `Scientific`
- `Enginnering`

defaultLabel

Specifies the default label of the horizontal marker.

Syntax

`viva.horizMarker defaultLabel string "%M" nil`

Default Value

- `%M`—Name of the horizontal marker

Valid Values

- `%X`—X-axis coordinates
- `%Y`—Y-axis coordinates
- `%N`—Name of the trace

foreground

Specifies the foreground color of the horizontal marker.

Syntax

`viva.vertMarker foreground string "black" nil`

Default Value

`black`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

interceptStyle

Specifies the intercept style of the horizontal marker. This variable also controls the visibility of axes intercepts.

Syntax

```
viva.horizMarker interceptStyle string "OnWhenHover" nil
```

Default Value

`OnWhenHover`	Displays the intercepts when you hover mouse pointer on the horizontal marker.

Valid Values

`On`	Always displays intercepts on the horizontal marker.
`Off`	Does not display intercepts on the horizontal marker.

lineStyle

Specifies the line style of the horizontal marker.

Syntax

```
viva.horizMarker lineStyle string "dash" nil
```

Default Value

`dash`

Valid Values

- `dash`
- `solid`

- `dot`
- `dashdot`
- `dashdotdot`
- `none`

significantDigits

Specifies the significant digits for the horizontal marker.

`viva.horizMarker significantDigits string "4" nil`

Default value

`4`

Valid Values

Any integer value.

snapPoint

Specifies the snap data point of the horizontal marker.

Syntax

`viva.horizMarker snapPoint string "Data Point" nil`

Default Value

`Data Point`	Shifts the marker to a specific data point on the curve.

Valid Values

`Local Maxima`	Shifts the marker to the maxima value (peak) local to the curve.
`Local Minima`	Shifts the marker to the minima value local to the curve.

`Local Max or Min`	Shifts the marker to either maxima or minima value local to the curve.
`Specific Y value`	Shifts the marker to a specific Y-axis value.
`Specific X value`	Shifts the marker to a specific X-axis value.

Note: The `Specific X value` option is not available for horizontal markers.

`Global Maxima`	Shifts the marker to the maxima value (peak) global to the curve.
`Global Minima`	Shifts the marker to the minima value global to the curve.

Reference Line Marker Environment Variables

This section describes the environment variables for reference line marker:

font

Specifies the font of the Reference Line marker.

Syntax

```
viva.referenceLineMarker font string "Default,10,-1,5,50,0,0,0,0,0" nil
```

Valid Values

See Font String on page 390.

interceptStyle

Specifies the intercept style of the reference line marker.

Syntax

```
viva.referenceLineMarker interceptStyle string "off" nil
```

Valid Values

`off`	Turns off the intercepts on the reference line marker. This is the default value.
`on`	Turns on the intercepts on the reference line marker.

lineStyle

Specifies the line style of the reference line marker.

Syntax

```
viva.referenceLineMarker lineStyle string "Dot" nil
```

Default Value

`Dot`

Valid Values

- `dash`
- `solid`
- `dot`
- `dashdot`
- `dashdotdot`
- `none`

foreground

Specifies the foreground color of the reference line marker.

Syntax

```
viva.referenceLineMarker foreground string "aquamarine" nil
```

Default Value

`aquamarine`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

defaultLabel

Specifies the default label of the vertical marker.

Syntax

```
viva.referenceLineMarker defaultLabel string "%Y" nil
```

Default Value

- `%Y`—Y-axis coordinates

Valid Values

- `%M`—Name of the reference line point marker
- `%X`—X-axis coordinates
- `%N`—Name of the trace

significantDigits

Specifies the significant digits for the reference line marker.

```
viva.referenceLineMarker significantDigits string "4" nil
```

Default value

`4`

Valid Values

Any integer value

Vertical Marker Environment Variables

This section describes the environment variables for vertical marker:

font

Specifies the font of the Vertical marker.

Syntax

```
viva.vertMarker font string "Default,10,-1,5,50,0,0,0,0,0" nil
```

Valid Values

See Font String on page 390.

notation

Specifies the notation of the vertical marker.

Syntax

```
viva.vertMarker notation string "suffix" nil
```

Default Value

`Suffix`

Valid Values

- `Scientific`
- `Enginnering`

defaultLabel

Specifies the default label of the vertical marker.

Syntax

`viva.vertMarker defaultLabel string "%M" nil`

Default Value

- `%M`—Name of the vertical marker

Valid Values

- `%X`—X-axis coordinates
- `%Y`—Y-axis coordinates
- `%N`—Name of the trace

foreground

Specifies the foreground color of the vertical marker.

Syntax

`viva.vertMarker foreground string "black" nil`

Default Value

`black`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

interceptStyle

Specifies the intercept style of the vertical marker. This variable also controls the visibility of axes intercepts.

Syntax

```
viva.vertMarker interceptStyle string "OnWhenHover" nil
```

Default Value

`OnWhenHover`	Displays the intercepts when you hover mouse pointer on the vertical marker.

Valid Values

`On`	Always displays intercepts on the vertical marker.
`Off`	Does not display intercepts on the vertical marker.

lineStyle

Specifies the line style of the vertical marker.

Syntax

```
viva.vertMarker lineStyle string "dash" nil
```

Default Value

`dash`

Valid Values

- `dash`
- `solid`

- `dot`
- `dashdot`
- `dashdotdot`
- `none`

significantDigits

Specifies the significant digits for the vertical marker.

`viva.vertMarker significantDigits string "4" nil`

Default value

`4`

Valid Values

Any integer value

snapPoint

Specifies the snap data point of the vertical marker.

Syntax

`viva.vertMarker snapPoint string "Data Point" nil`

Default Value

`Data Point`	Shifts the marker to a specific data point on the curve.

Valid Values

`Local Maxima`	Shifts the marker to the maxima value (peak) local to the curve.
`Local Minima`	Shifts the marker to the minima value (peak) local to the curve.

`Local Max or Min`	Shifts the marker to either maxima or minima value local to the curve.
`Specific Y value`	Shifts the marker to a specified Y value.
`Specific X value`	Shifts the marker to a specific X-axis value.

Note: The `Specific X value` option is not available for horizontal markers.

`Global Maxima`	Shifts the marker to the maxima value (peak) global to the curve.
`Global Minima`	Shifts the marker to the minima value global to the curve.

Point Marker Environment Variables

This section describes the environment variables for point marker:

font

Specifies the font of the point marker.

Syntax

```
viva.pointMarker font string "Default,10,-1,5,50,0,0,0,0,0" nil
```

Valid Values

See Font String on page 390.

notation

Specifies the notation of the point marker.

Syntax

```
viva.pointMarker notation string "suffix" nil
```

Default Value

`suffix`

Valid Values

- `Engineering`
- `Scientific`

defaultLabel

Specifies the default label of the point marker.

Syntax

```
viva.pointMarker defaultLabel string "%M: %X %Y" nil
```

Default Value

- `%M`—Name of the point marker
- `%X`—X-axis coordinates
- `%Y`—Y-axis coordinates

Valid Values

- `%N`—Name of the trace

foreground

Specifies the foreground color of the point marker.

Syntax

```
viva.pointMarker foreground string "black" nil
```

Default Value

`black`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

background

Specifies the background color of the point marker.

Syntax

```
viva.pointMarker background string "white" nil
```

Default Values

`white`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

significantDigits

Specifies the significant digits for the point marker.

```
viva.pointMarker significantDigits string "4" nil
```

Default value

`4`

Valid Values

Any integer value

snapPoint

Specifies the snap point of the point marker.

Syntax

```
viva.pointMarker snapPoint string "Data Point" nil
```

Default Value

`Data Point`	Shifts the marker to a specific data point on the curve.

Valid Values

`Local Maxima`	Shifts the marker to the maxima value (peak) local to the curve.
`Local Minima`	Shifts the marker to the minima value (peak) local to the curve.
`Local Max or Min`	Shifts the marker to either maxima or minima value local to the curve.
`Specific Y value`	Shifts the marker to a specified Y value.
`Specific X value`	Shifts the marker to a specific X-axis value.

Note: The `Specific X value` option is not available for horizontal markers.

`Global Maxima`	Shifts the marker to the maxima value (peak) global to the curve.
`Global Minima`	Shifts the marker to the minima value global to the curve.

drawCrossHairs

Specifies whether to display the X and Y intercepts for point markers in the graph.

Syntax

```
viva.pointMarker drawCrossHairs string "Dynamic" nil
```

Default Value

- `Dynamic`—Displays the X and Y intercepts when the point marker is clicked.

Valid Values

- `On`—Hides the X and Y intercepts
- `Off`—Shows the X and Y intercepts

circDefaultLabel

Specifies the default label of the point marker.

Syntax

```
viva.pointMarker circDefaultLabel string "%M: %C (%F)" nil
```

Default Value

`%M: %C (%F)`

where,

- `%M`—Name of the marker
- `%C`— Real and imaginary (complex)
- `%F`—Frequency

Valid Values

- `%P`—Gamma (polar)
- `%Z`—Impedance
- `%A`—Admittance
- `%R`—Reflection

Reference Point Marker Environment Variables

This section describes the environment variables for reference point marker:

font

Specifies the font of the reference point marker.

Syntax

viva.refPointMarker font string "`Default,10,-1,5,50,0,0,0,0,0`**" nil**

Valid Values

See Font String on page 390.

notation

Specifies the notation of the reference point marker.

Syntax

viva.refPointMarker notation string "suffix" nil

Default Value

`suffix`

Valid Values

- `Engineering`
- `Scientific`

foreground

Specifies the foreground color of the reference point marker.

Syntax

```
viva.refPointMarker foreground string "black" nil
```

Default Value

`black`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

defaultLabel

Specifies the default label of the reference point marker.

Syntax

```
viva.refPointMarker defaultLabel string "%X %Y" nil
```

Default Value

- `%X`—X-axis coordinates
- `%Y`—Y-axis coordinates

Valid Values

- `%M`—Name of the marker

- `%N`—Name of the trace

circDefaultLabel

Specifies the circular default label of the reference point marker.

Syntax

```
viva.refPointMarker circDefaultLabel string "%C (%F)" nil
```

Default Value

- `%C`—Real and imaginary (complex)
- `%F`—Frequency

Valid Values

- `%P`—Gamma (polar)
- `%Z`—Impedance
- `%A`—Admittance
- `%R`—Reflection

drawCrossHairs

Specifies whether to display the X and Y intercepts for reference point markers in the graph.

Syntax

```
viva.refPointMarker drawCrossHairs string "Dynamic" nil
```

Default Value

- `Dynamic`—Displays the X and Y intercepts when the reference point marker is clicked.

Valid Values

- `On`—Hides the X and Y intercepts

- `Off`—Shows the X and Y intercepts

significantDigits

Specifies the significant digits for the reference point marker.

```
viva.refPointMarker significantDigits string "4" nil
```

Default value

`4`

Valid Values

Any integer value

Specification Marker Environment Variables

This section describes the environment variables for specification marker:

font

Specifies the font of the spec marker.

Syntax

viva.specMarker font string "Default,10,-1,5,50,0,0,0,0,0**" nil**

Valid Values

See Font String on page 390.

passcolor

Specifies the color of the pass region of the spec marker.

Syntax

viva.specMarker passcolor string "#00CC00" nil

Default Value

#00CC00

Valid Values

All values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

Note: Color values are specified in the #RRGGBB format. For example, the cyan color in RGB format is (0,255, 255) and in hexadecimal format, it is written as #00FFFF.

failcolor

Specifies the color of the fail region of the spec marker.

Syntax

```
viva.specMarker failcolor string "#CC0000" nil
```

Default Value

`#CC0000`

Valid Values

All values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

Color values are specified in the #RRGGBB format. For example, the cyan color in RGB format is (0,255, 255) and in hexadecimal format, it is written as #00FFFF.

lineStyle

Specifies the line style of the spec marker.

Syntax

```
viva.specMarker lineStyle string "solid" nil
```

Default Value

`solid`

Valid Values

- `dot`
- `dash`
- `dashDot`
- `dashDotDot`
- `none`

lineThickness

Specifies the line thickness of the spec marker.

Syntax

`viva.specMarker lineThickness string "medium" nil`

Default Value

`medium`

Valid Values

- `fine`
- `thick`
- `extraThick`

lineColor

Specifies the line color of the spec marker.

Syntax

`viva.specMarker lineColor string "white" nil`

Default Value

`white`

Valid Values

- `black`
- `red`
- `blue`
- `green`
- `yellow`

displayMode

Specifies the spec marker region display mode of the spec marker.

Syntax

`viva.specMarker displayMode string "both" nil`

Default Value

`both`

Valid Values

- none—Displays no spec marker on the graph
- `pass`—Displays only pass spec marker region on the graph
- `fail`—Displays only fail spec marker region on the graph
- `thresholdOnly`—Displays threshold region on the graph

significantDigits

Specifies the significant digits for the specification marker.

`viva.specMarker significantDigits string "4" nil`

Default value

`4`

Valid Values

Any integer value

showLabel

Displays the spec marker labels on the graph.

Syntax

`viva.specMarker showLabel string "true" nil`

Default Value

`true`

Valid Values

- `true`—Shows the spec marker labels.
- `false`—Hides the spec marker labels.

rule

Sets the specifictaion type to be used for the spec marker.

Syntax

`viva.specMarker rule string "range" nil`

Default Value

`range`

Valid Values

- `none`
- `minimize`
- `maximize`
- `<`

- `>`
- `tol`
- `info`

type

Specifies whether the specification type of spec waveform is drawn normally or in a sampleHold style.

Syntax

`viva.specMarker type string "line" nil`

Default Value

`line`

Valid Values

- `line`
- `stairStep`

Intercept Marker Environment Variables

This section describes the environment variables for Intercept marker:

- foreground on page 441
- lineStyle on page 441

foreground

Specifies the foreground color of the intercept marker.

Syntax

```
viva.interceptMarker foreground string "black" nil
```

Default Value

`black`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

lineStyle

Specifies the line style of the intercept edge marker.

Syntax

```
viva.interceptMarker lineStyle string "solid" nil
```

Default Value

`solid`

Valid Values

- `dot`
- `dash`
- `dashDot`
- `dashDotDot`
- `none`

Circle Marker Environment Variables

This section describes the environment variables for circular marker:

font

Specifies the font of the circular marker.

Syntax

```
viva.circleMarker font string "Default,10,-1,5,50,0,0,0,0,0" nil
```

Valid Values

See Font String on page 390.

notation

Specifies the notation of the circular marker.

Syntax

```
viva.circleMarker notation string "suffix" nil
```

Default Value

`suffix`

Valid Values

`Engineering`

defaultLabel

Specifies the mnemonic label for the circular markers.

Syntax

```
viva.circleMarker defaultLabel string "%M: %X %Y" nil
```

Default Value

- `%M`—Name of the marker
- `%X`—X-axis coordinates
- `%Y`—Y-axis coordinates

Valid Values

- `%N`—Name of the trace.

foreground

Specifies the foreground color of the circular marker.

Syntax

```
viva.cicleMarker foreground string "black" nil
```

Default Value

`black`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

background

Specifies the background color of the circular marker.

Syntax

```
viva.circleMarker background string "white" nil
```

Default Values

`white`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

Delta Marker Environment Variables

This section describes the environment variables for delta markers:

- foreground on page 446
- notation on page 446
- showChildLabels on page 447
- defaultLabel on page 447
- significantDigits on page 448

foreground

Specifies the foreground color of the delta marker.

Syntax

```
viva.multiDeltaMarker foreground string "black" nil
```

Default Value

`black`

Valid Values

All the values defined at the following location:

http://www.w3.org/TR/SVG/types.html#ColorKeywords

notation

Specifies the notation of the delta marker.

Syntax

```
viva.multiDeltaMarker notation string "suffix" nil
```

Default Value

`suffix`

Valid Values

- `Engineering`
- `Scientific`

showChildLabels

Shows or hides marker labels for the delta markers.

Syntax

```
viva.multiDeltaMarker showChildLabels string "true" nil
```

Values

- `true`—Shows the labels for all markers that combine to form the delta marker.
- `false`—Hides the labels for all markers that combine to form the delta marker.

defaultLabel

Specifies the mnemonic label for the delta markers.

Syntax

```
viva.multiDeltaMarker defaultLabel string "dx:%W dy:%H s:%S" nil
```

Default Value

`dx:%W dy:%H s:%S`

where,

- `dx:%W`—Delta value on X-axis
- `dy:%H`—Delta value on Y-axis

- `s:%S`—Slope (dx/dy)

Valid Values

- `%M`—Name of the marker
- `%N`—Name of the trace

significantDigits

Specifies the significant digits for the delta marker.

```
viva.multiDeltaMarker significantDigits string "4" nil
```

Default value

`4`

Valid Values

Any integer value

Transient Edge Markers Environment Variables

This section describes the environment variables for transient edge markers:

-

significantDigits

Specifies the significant digits for the transient edge marker.

```
viva.transEdgeMarker significantDigits string "4" nil
```

Default value

4

Valid Values

Any integer value

Graph Label Environment Variables

- `viva.graphLabel font string "Default,10,-1,5,50,0,0,0,0,0" nil`
- `viva.graphLabel foreground string "white" nil`
- `viva.graphLabel background string "lightGray" nil`

Probe Environment Variables

- `viva.probe font string Default,10,-1,5,50,0,0,0,0,0" nil`
- `viva.probe foreground string "black" nil`
- `viva.probe background string "white" nil`
- `viva.probe autoTopBaseline string "true" nil`
- `viva.probe topLine string "0.0" nil`
- `viva.probe baseLine string "0.0" nil`
- `viva.probe autoMinMax string "true" nil`
- `viva.probe minValue string "0.0" nil`
- `viva.probe maxValue string "0.0" nil`

Polar Grid Environment Variables

- `viva.polarGrid font string "Default,10,-1,5,50,0,0,0,0,0" nil`
- `viva.polarGrid numCircles string "4" nil`
- `viva.polarGrid circlesForeground string "gray" nil`
- `viva.polarGrid circlesOn string "true" nil`
- `viva.polarGrid circlesLabelsOn string "true" nil`
- `viva.polarGrid numRadials string "2" nil`
- `viva.polarGrid radialsOn string "true" nil`
- `viva.polarGrid radialsLabelsOn string "true" nil`

Smith Grid Environment Variables

- `viva.smithGrid highlightUnitCircle string "true" nil`
- `viva.smithGrid showAxes string "true" nil`
- `viva.smithGrid showOrigin string "true" nil`
- `viva.smithGrid showCurves string "true" nil`
- `viva.smithGrid showMinorCurves string "true" nil`
- `viva.smithGrid font string "Default,10,-1,5,50,0,0,0,0,0" nil`
- `viva.smithGrid showPerimeterLabels string "true" nil`

Results Browser Variables

This section describes the following Results Browser environment variables:

historyLength

Specifies the maximum number of dataset paths saved in the *Location* pull-down in the Results Browser window.

Syntax

```
viva.browser historyLength string "maxDirectories"
```

Values

maxDirectories	Maximum number of data directories that fit in the *Location* field. Default: `10` Valid values: 0–20

plotstyle

Specifies the default plotting style for a new graph. This sets the default value for the plot style pull-down in the top right corner of the Results Browser window.

Syntax

```
viva.browser plotStyle string "Append" | "Replace" | "New SubWin" | "New Win"
```

Values

`append`	Appends the new graph to the current graph.
`replace`	Replaces the current graph with the new graph. This is the default value.
`newsub`	Plots the graph in a new subwindow.
`newwin`	Plots the graph in a new window.

dataDirHome

Specifies the default directory for the Choose Data Directory dialog box.

Syntax

```
viva.browser dataDirHome string "directory"
```

Values

directory	Directory which the Choose Data Directory dialog box defaults to. Default: `./`

Calculator Variables

This section describes the following Calculator environment variables:

usePreviousGuiSettings

Specifies whether the local defaults are to override the .cdsenv settings.

Syntax

`viva.calculator usePreviousGuiSettings string "true" | "false"`

Values

`true`	Local defaults for the following variables to override the `.cdsenv` settings: .`showKeyPad`, `showStack`, `rpnMode`, `clipSelectionMode`, `signalSelection`, `plotStyle`, `defaultCategory`, `mathToolBar`,`TrigToolBar`, `schematicToolBar`, `schematicAnalyses`, `userButton`, `xLocation`, `yLocation`, `width`, and `height`. This is the default value.
`false`	`.cdsenv` variable settings to override the local defaults.

rpnMode

Specifies whether the Calculator is in the rpn mode by default. You can change this setting through the Options menu in the Calculator window (*Options–Set RPN*).

Syntax

```
viva.calculator rpnMode string "true" | "false"
```

Values

`true`	Calculator is in rpn mode. This is the default value.
`false`	Calculator is in algebraic mode.

clipSelectionMode

Controls whether the *Clip* check box is selected by default in the Calculator window.

Syntax

```
viva.calculator clipSelectionMode string "true" |"false"
```

Values

`true`	The *Clip* check box is selected, hence the calculator works on the visible X-axis of a zoomed in trace. This is the default value.
`false`	The *Clip* check box is not selected, hence the calculator works on the complete trace.

displayContext

Specifies whether the test (in ADE XL mode) or results directory (in ADE L mode) is displayed.

Syntax

```
viva.calculator displayContext string "true" | "false"
```

Values

`true`	*Test* and *Results Dir* are displayed in the ADE XL mode. If there is no current test, only the results directory is displayed. In ADE L mode, the results directroy is displayed. This is the default value.
`false`	*Test* and *Results Dir* are not displayed.

plotStyle

Controls the plot destination for graphs from the Calculator window.

Syntax

```
viva.calculator signalselection string "append" |"replace" | "new Subwindow"
     |"New Window"
```

Values

`append`	The trace is appended to an existing graph. This is the default value.
`replace`	The trace replaces the existing trace.
`new Subwindow`	The trace is plotted to a new subwindow.
`new Window`	The trace is plotted to a new window.

signalselection

Controls the *Selection choices* in the Calendar window.

Syntax

`viva.calculator signalselection string "off" |"wave" | "family"`

Values

`off`	Selection choice in the Calculator window is set to off. This is the default value.
`wave`	Selection choice in the Calculator window is set to *wave*.
`family`	Selection choice in the Calculator window is set to *family*.

familyMode

Controls whether the *Family* button is selected by default in the Calculator window.

Syntax

`viva.calculator familyMode string "true" | "false"`

Values

`true`	The *Family* button is selected.
`false`	The *Family* button is not selected. This is the default value.

defaultCategory

Specifies the function category to be displayed.

Syntax

```
viva.calculator displayContext string "All" | "Favorites" | "Math" | "Modifier"
     | "Programmed Keys" | "RF Functions" | "Special Functions" |
     "Trigonometric" | "AWD Programmed Keys" | "SKILL Defined User Functions"
     | "Memories"
```

Values

`All`	Displays all the functions.
`Favorites`	Displays your favorite functions.
`Math`	Displays the math functions.
`Modifier`	Displays the modifier functions.
`Programmed Keys`	Displays the programmed keys.
`RF Functions`	Displays the RF functions.
`Special Functions`	Displays the special functions. This is the default value.
`Trigonometric`	Displays the trigonometric functions.
`AWD Programmed Keys`	Displays the AWD programmed keys.
`SKILL Defined User Functions`	Displays the SKILL defined functions.
`Memories`	Displays the memories you created.

reportVarErrors

Controls whether the Calculator reports errors when validating an MDL expression that contains variables. This variable applies to ViVA only in the MDL mode.

Syntax

```
viva.calculator reportVarErrors string "true" | "false"
```

Values

`true`	Calculator reports errors when validating an MDL expression that contains variables.
`false`	Calculator does not report errors when validating an MDL expression that contains variables. This is the default value.

sizeKeyPad

Controls the size of buttons of numeric keypad. It can be defined in small, medium or large sizes.

Syntax

viva.caculator sizeKeyPad string "small" "medium" "large"

Values

small	Button size will be small. This is default value
medium	Button size will be 1.5 times of small buttons.
large	Button size will be 2 times of small buttons.

stackSize

Controls the maximum number of expressions displayed in the Calculator stack. A scrollable list is displayed after this number is exceeded.

Syntax

```
viva.calculator stackSize string "stack_number"
```

Values

stack_number	Number of expressions displayed in the Calculator. Default: 8 Valid values: 0–20

undoStackSize

Controls the maximum number of commands that can be undone.

Syntax

`viva.calculator undoStackSize string "`*`undo_stack_number`*`"`

Values

`undo_stack_number`	Number of commands that can be undone. Default: `8` Valid values: 0–20

signalHistorySize

Controls the maximum number of items stored in the *Signal* field drop-down.

Syntax

viva.calculator signalHistorySize string "*signal_history_size***"**

Values

signal_history_size	Number of items stored in the *Signal* field drop-down in the function panel of the Calculator. Default: 8 Valid values: 0–20

xLocation

Controls the position where the Calculator window appears.

Syntax

`viva.calculator xLocation string "`*`x_position`*`"`

Values

`x_position`	Horizontal distance of the Calculator window from the left of the screen. Default: `600` Valid values: A positive integer.

yLocation

Controls the position of the Calculator window.

Syntax

`viva.calculator yLocation string "`*`y_position`*`"`

Values

`y_position`	Vertical distance of the Calculator window from the top of the screen. Default: 50 Valid values: A positive integer.

width

Controls the width of the calculator window.

Syntax

`viva.calculator width string "`*`width_pixels`*`"`

Values

`width_pixels`	Width of the Calculator window. Default: `640` Valid values: A positive integer greater than 640.

height

Controls the height of the Calculator window.

Syntax

`viva.calculator height string "`*`height_pixels`*`"`

Values

`height_pixels`	Height of the Calculator window. Default: `330` Valid values: A positive integer greater than 330.

showKeypad

Specifies whether the keypad is displayed.

Syntax

```
viva.calculator showKeypad string "true" | "false"
```

Values

`true`	Keypad is displayed. This is the default value.
`false`	Keypad is not displayed.

showStack

Specifies whether the stack (in RPN mode) or history (in Algebraic mode) is displayed.

Syntax

```
viva.calculator showStack string "true" | "false"
```

Values

`true`	Stack is displayed.
`false`	Stack is not displayed. This is the default value.

mathToolBar

Specifies whether the Math tool bar is displayed by default.

Syntax

`viva.calculator schematicToolBar string "true" | "false"`

Values

`true`	Math tool bar is displayed.
`false`	Math tool bar is not displayed. This is the default value

trigToolBar

Specifies whether the trigonometric tool bar is displayed by default.

Syntax

```
viva.calculator schematicToolBar string "true" | "false"
```

Values

`true`	Trigonometric tool bar is displayed.
`false`	Trigonometric tool bar is not displayed. This is the default value.

schematicToolBar

Specifies whether the schematic access buttons (in ADE L and ADE XL modes) are displayed.

Syntax

`viva.calculator schematicToolBar string "true" | "false"`

Values

`true`	Schematic access buttons are displayed. This is the default value
`false`	Schematic access buttons are not displayed.

schematicAnalyses

Controls the analyses for which the schematic access buttons (in ADE L and ADE XL modes) are displayed. This variable is active only when the schematicToolBar variable is set to True.

Syntax

```
viva.calculator displayContext string "tran" | "ac" | "dc" | "sweptDc" | "info"
     | "noise" | "rf"
```

Values

`tran`	Displays the *vt* and *i* buttons.
`ac`	Displays *vf* and *if* buttons.
`dc`	Displays the *vdc* and *idc* buttons.
`sweptDc`	Displays the *vs* and *is* buttons.
`info`	Displays the *op*, *var*, *opt*, and *mp* buttons.
`noise`	Displays the *vn* and *vn2* buttons.
`rf`	Displays the *sp*, *zp*, *vswr*, *yp*, *hp*, *gd*, *zm*, and *data* buttons.

userButton

Associates a function with the *user* button.

Syntax

viva.calculator userButton1 string "*abbreviation;function_name*" nil

viva.calculator userButton2 string "*abbreviation;function_name*" nil

viva.calculator userButton3 string "*abbreviation;function_name*" nil

viva.calculator userButton4 string "*abbreviation;function_name*" nil

viva.calculator userButton5 string "*abbreviation;function_name*" nil

viva.calculator userButton6 string "*abbreviation;function_name*" nil

viva.calculator userButton7 string "*abbreviation;function_name*" nil

viva.calculator userButton8 string "*abbreviation;function_name*" nil

viva.calculator userButton9 string "*abbreviation;function_name*" nil

viva.calculator userButton10 string "*abbreviation;function_name*" nil

viva.calculator userButton11 string "*abbreviation;function_name*" nil

viva.calculator userButton12 string "*abbreviation;function_name*" nil

Values

abbreviation	Name to be displayed on the user button. You can enter up to 6 characters for the abbreviation.
function_name	Function to be associated with the *user* button.

Example

viva.calculator userButton1 string "bw;bandwidth" nil

B

Calculator Functions

This chapter describes the functions in the function panel for both the SKILL and MDL modes.

Table B-1 Functions in the MDL Mode

abs	crosses	imag	rms
acos	d2r	int	round
acosh	db	integ	sample
angle	db10	ln	settlingTime
argmax	dBm	log10	sign
argmin	deltax	mag	sin
asin	deriv	max	sinh
asinh	dutycycle	min	slewRate
atan	dutycycles	mod	snr
atanh	exp	movingavg	sqrt
avg	falltime	overshoot	stathisto
bw	fft	period_jitter	tan
ceil	flip	ph	tanh
cfft	floor	phaseMargin	trim
clip	freq	pow	window
conj	freq_jitter	pp	xval
convolve	gainBwProd	psd	yval
cos	gainMargin	pzbode	
cosh	groupdelay	pzfilter	
cplx	histo	r2d	
cross	ifft	real	
	iinteg	riseTime	

angle

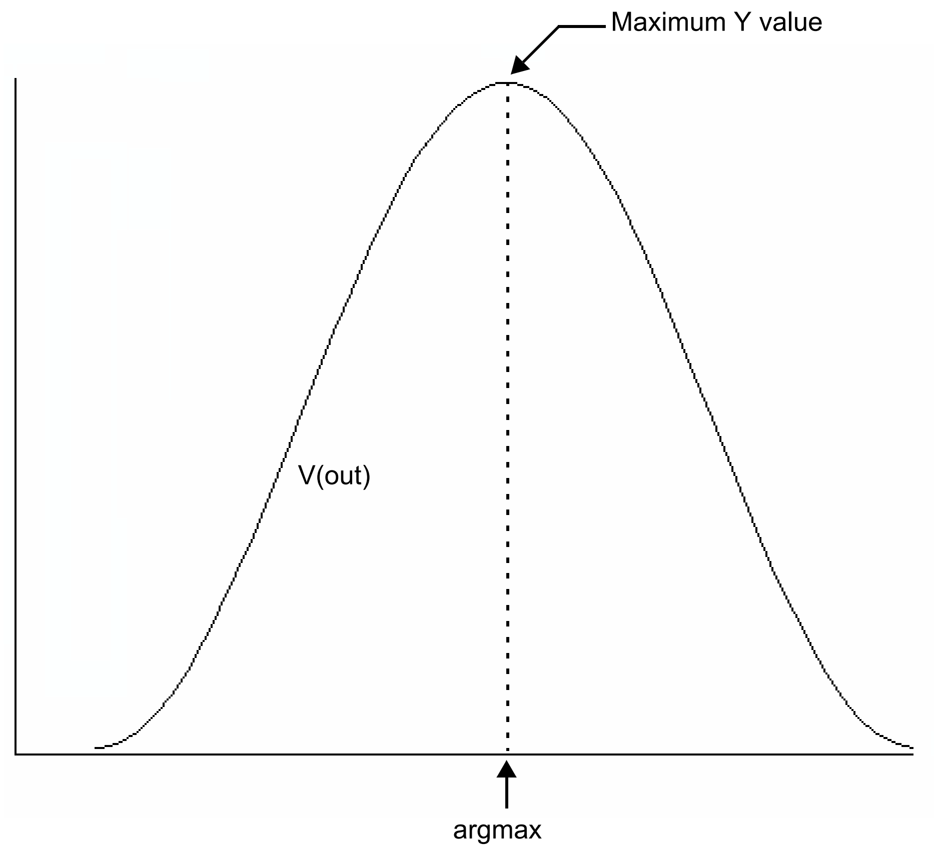
Returns the angle of a complex number in degrees. This function is available only in the MDL mode.

argmax

Returns the X value corresponding to the maximum Y value of a signal. This function is available only in the MDL mode.

If multiple X values are returned, the first one is used.

Example

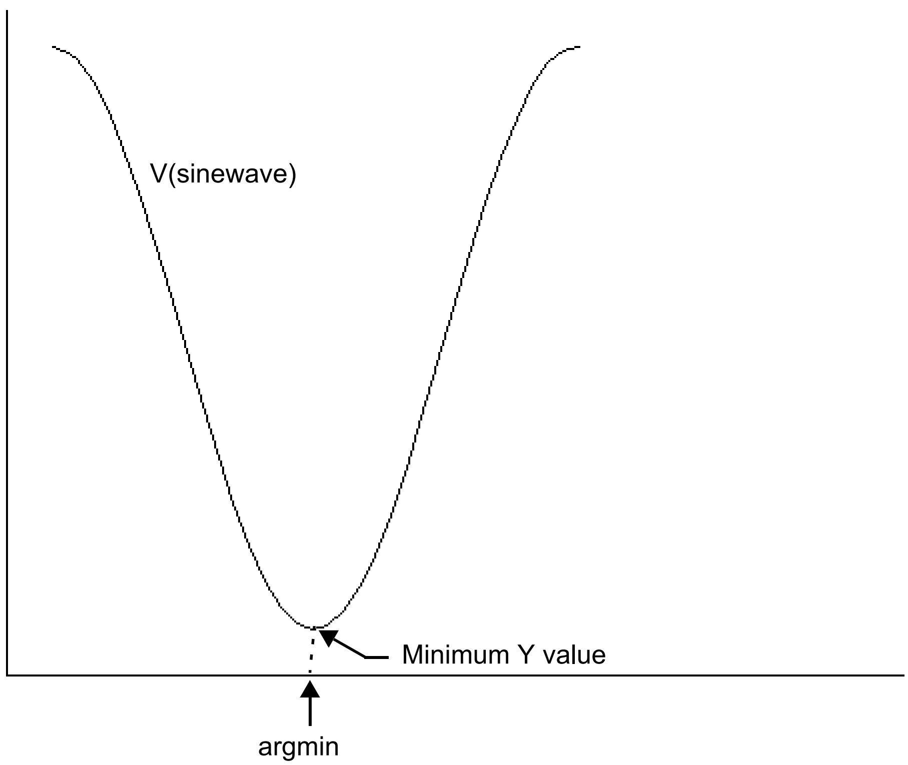


argmin

Returns the X value corresponding to the minimum Y value of a signal. This function is available only in the MDL mode.

If multiple X values are returned, the first one is used.

Example



b1f

Returns the stability factor b1f. This function is available only in the SKILL mode.

ceil

Rounds a real number up to the closest integer value. This function is available only in the MDL mode.

cfft

Performs a Fast Fourier Transform on a complex time domain waveform and returns its frequency spectrum. The `cfft` function takes two time signals that in combination form a complex input signal. Available only in MDL mode.

- *sig_re* is the real part of the signal.
- *sig_im* is the imaginary part of the signal.
- *from* is the starting X value.
- *to* is the ending Y value.
- *numPoints* is the number of data points to be used for calculating the cfft. If this number is not a power of 2, it is automatically raised to the next higher power of 2.
- *window* is the algorithm used for calculating the cfft. In this release, only one algorithm is supported.
- *smoothing* is not supported in this release.

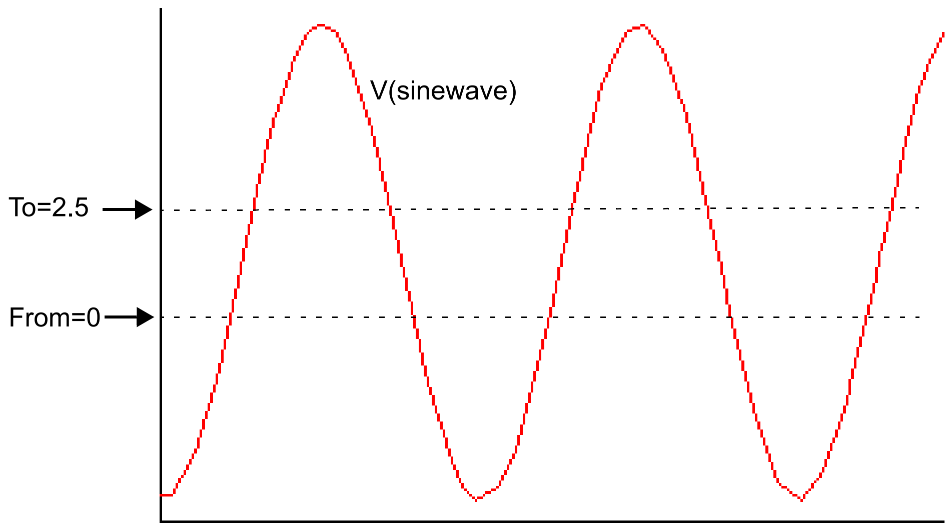
clip (MDL)

Returns the portion of a signal between two points along the Y-axis.

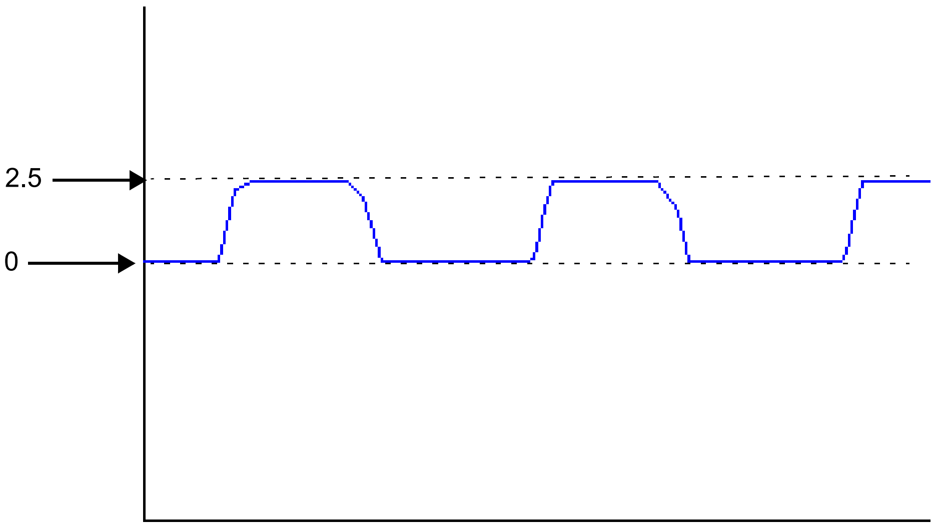
- *Signal* is the name of the signal.
- *From* is the starting point on the Y-axis.
- *To* is the ending point on the Y-axis.

Example

The following input signal



with the values *signal*=`V(sinewave)`, *From*=`0`, *To*=`2.5` is transformed into the following output signal.



cplx

Returns a complex number created from two real arguments. This function is available only in the MDL mode

- *R* is the value representing the real part.
- *I* is the value representing the imaginary part.

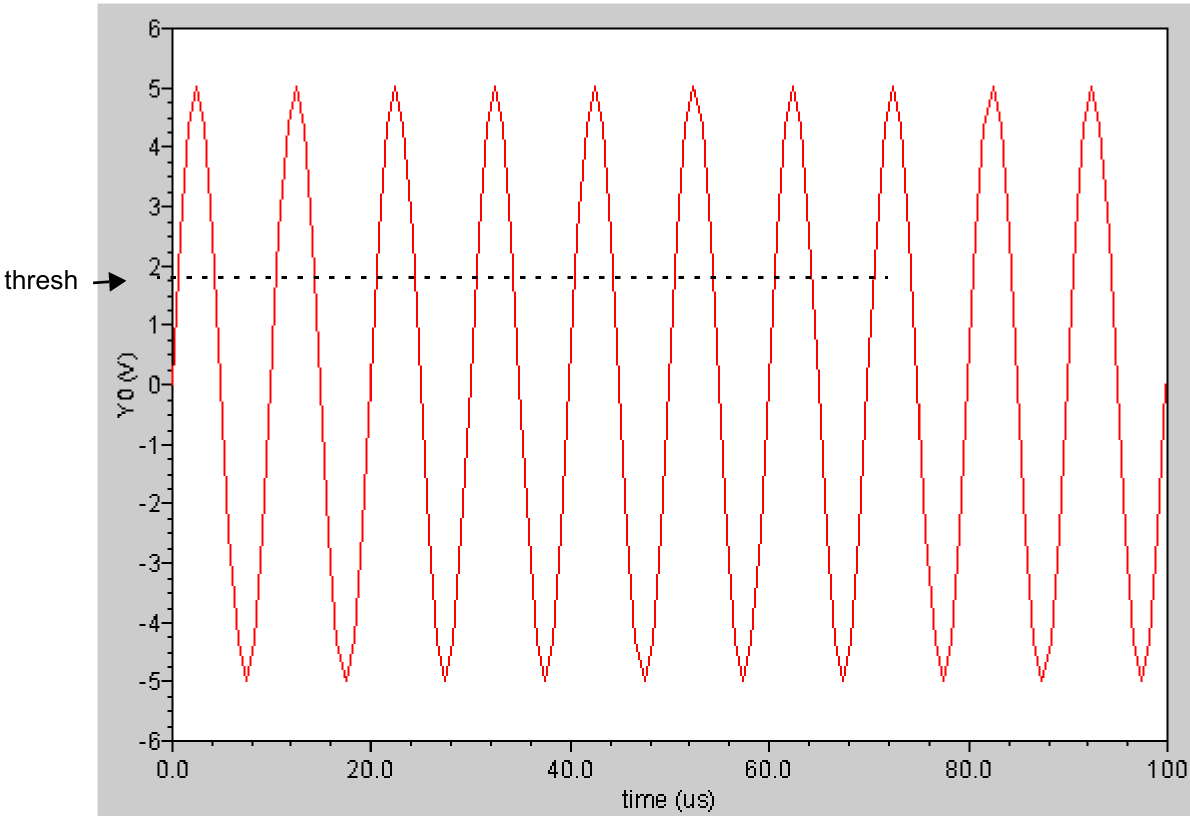
crosses

Returns the X values where a signal crosses the threshold Y value. This function is available only in the MDL mode.

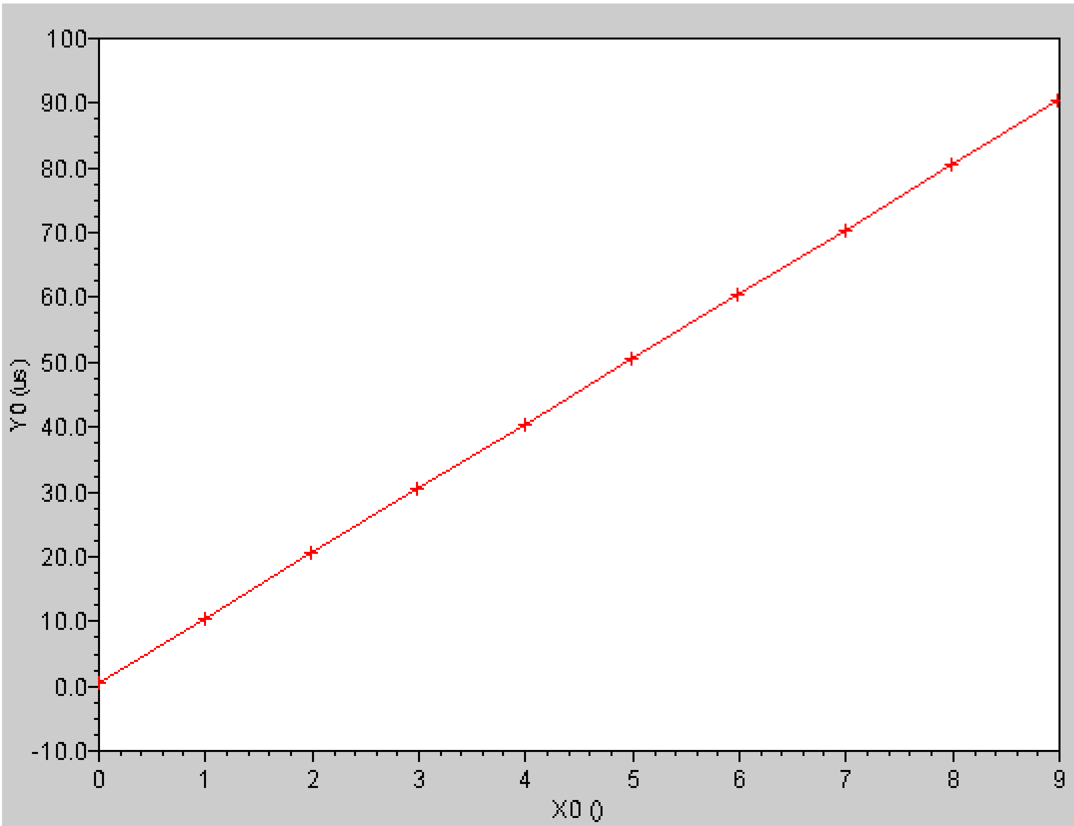
- *sig* is the name of the signal.
- *dir* is the direction of the crossing event. `'rise` directs the function to look for crossings where the Y value is increasing, `'fall` for crossings where the Y value is decreasing, and `'cross` for crossings in either direction.
- *n* is the occurrence of the crossing. If *n*=1, the function returns the first crossing and all subsequent crossings. If *n*=3, the function returns the third crossing and all subsequent crossings. The value of *n* can be negative numbers: if *n*=`-2`, only the last two crossings are returned.
- *thresh* is the threshold to be crossed.
- *start* is the time at which the function is enabled.
- *xtol* is the absolute tolerance in the X direction.
- *ytol* is the absolute tolerance in the Y direction.
- *accuracy* specifies whether the function should use interpolation, or use iteration controlled by the absolute tolerances to calculate the value. `'interp` directs the function to use interpolation, and `'exact` directs the function to consider the xtol and yval values.

Example

The following input signal with the values *sig*=`V(out)`, *dir*=`'rise`, and *thresh*=`1.0`



is transformed into the following output waveform.



d2r (degrees-to-radians)

Converts a waveform from degrees to radians. This function is available only in the MDL mode.

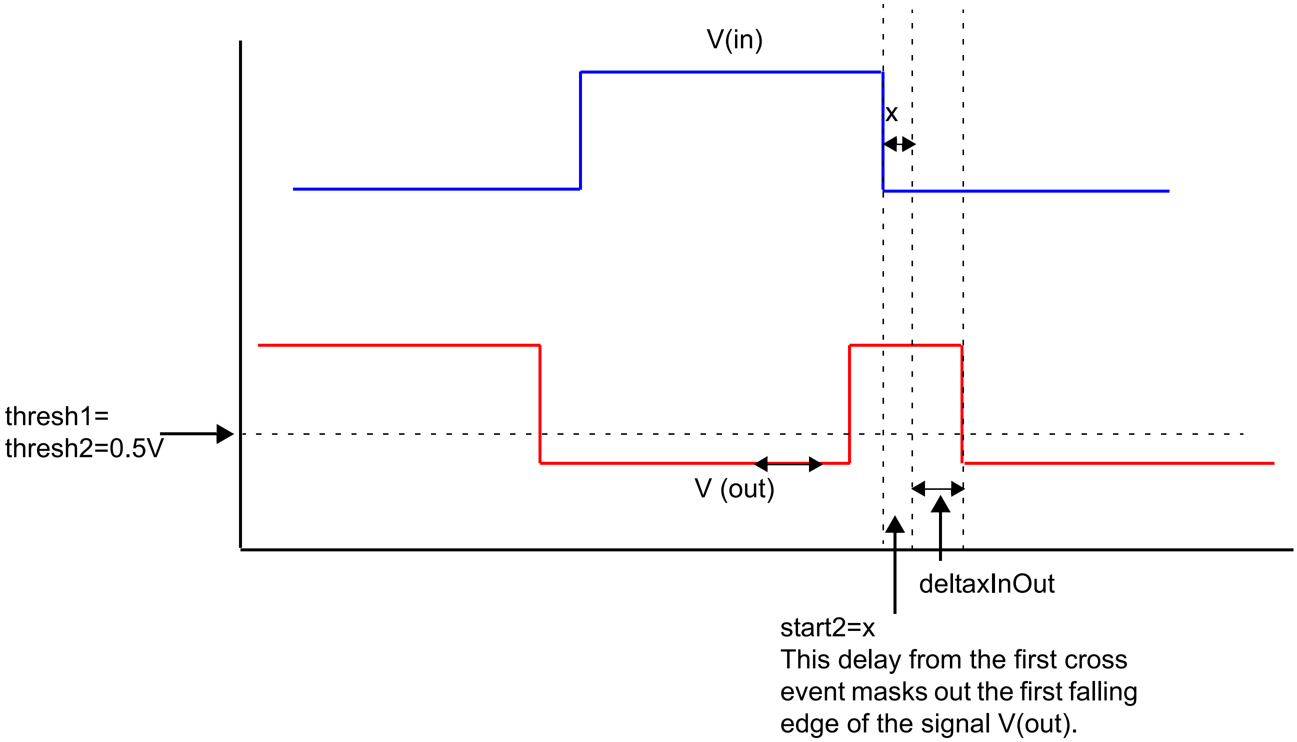
deltax

Returns the difference in the abscissas of two cross events. This function is available only in the MDL mode.

- *sig1* is the signal whose cross event begins the measurement interval.
- *sig2* is the signal whose cross event ends the measurement interval.
- *dir1* is the direction of the cross at the beginning of the measurement interval. `'rise` directs the function to look for crossings where the Y value is increasing, `'fall` for crossings where the Y value is decreasing, and `'cross` for crossings in either direction.
- *n1* is the occurrence of the crossing for the beginning of the measurement interval. The first crossing is n=1, the second crossing is n=2, and so on.
- *thresh1* is the Y value whose crossing begins the measurement interval.
- *start1* is the time at which the function is enabled.
- *dir2* is the direction of the cross at the end of the measurement interval. `'rise` directs the function to look for crossings where the Y value is increasing, `'fall` for crossings where the Y value is decreasing, and `'cross` for crossings in either direction.
- *n2* is the occurrence of the crossing for the end of the measurement interval. The first crossing is n=1, the second crossing is n=2, and so on.
- *thresh2* is the Y value whose crossing ends the measurement interval.
- *start2* is the offset from the time of the first cross event, where the function begins looking for the second cross event that ends the delay measurement.
- *xtol* is the absolute tolerance in the X direction.
- *ytol* is the absolute tolerance in the Y direction.
  Default: `1`
- *accuracy* specifies whether the function should use interpolation, or use iteration controlled by the absolute tolerances to calculate the value. `'interp` directs the function to use interpolation, and `'exact` directs the function to consider the xtol and yval values.
- *absstart2* specifies the time at which Virtuoso Visualization and Analysis XL should start looking for the second cross, instead of relative to the crossing of 1 (the *start2* parameter). You can use either the *start2* or *absstart2* parameter.

Example

The following diagram illustrates how the result is determined with the values *sig1*=`V(in)`, *sig2*=`V(out)`, *dir1*=`'fall`, *thresh1* = `0.5`, *dir2*=`'fall`, *thresh2*=`0.5`, *start2*=`x`



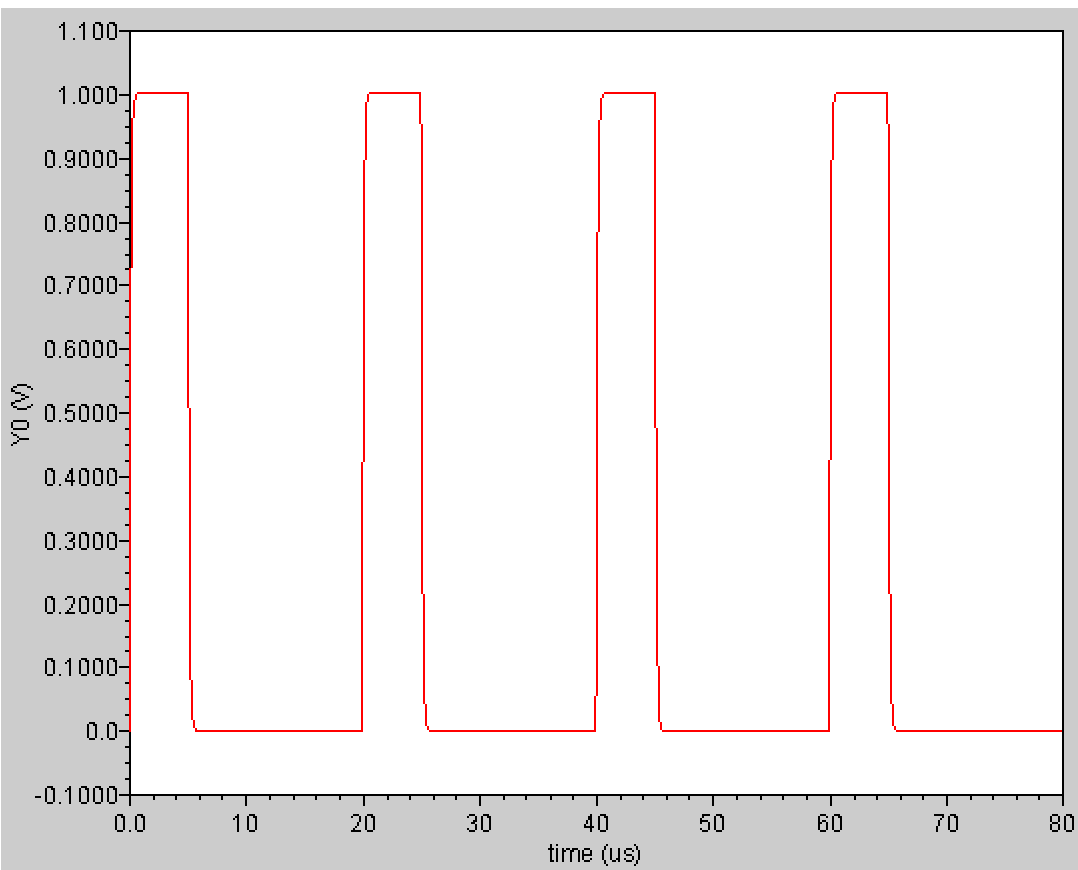
dutycycles

Returns the dutycycle of a nearly-periodic signal as a function of time. This function is available only in the MDL mode.

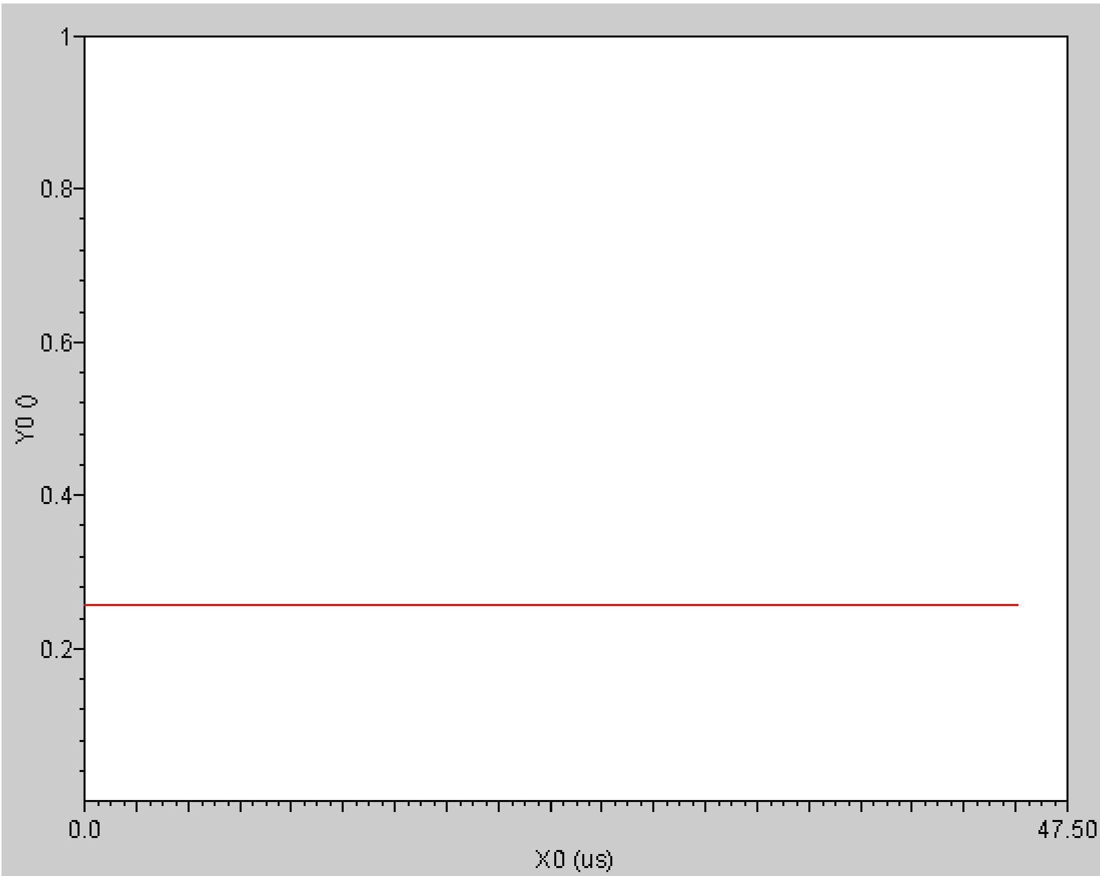
- *sig* is the name of the signal.
- *theta* is the percentage that defines the logic high of the signal. A threshold value is calculated as follows:
  yThresh=theta/100*(Ymax+Ymin)
  The portion of the signal above yThresh is taken as high.

Example

The following input signal with the values *sig*=`V(out)` and *theta*=`40`



transforms into the following output signal:



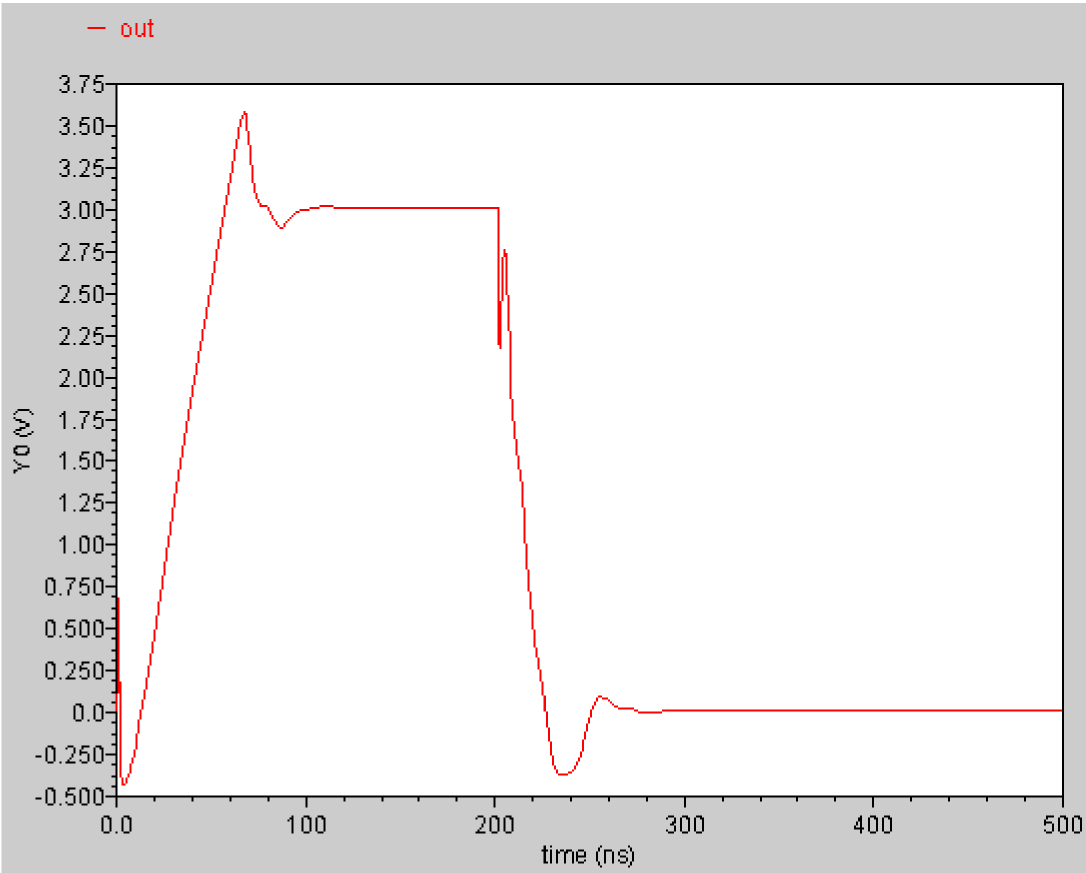
fft

Performs a Fast Fourier Transform on the signal and returns its frequency spectrum. This function is available only in the MDL mode.

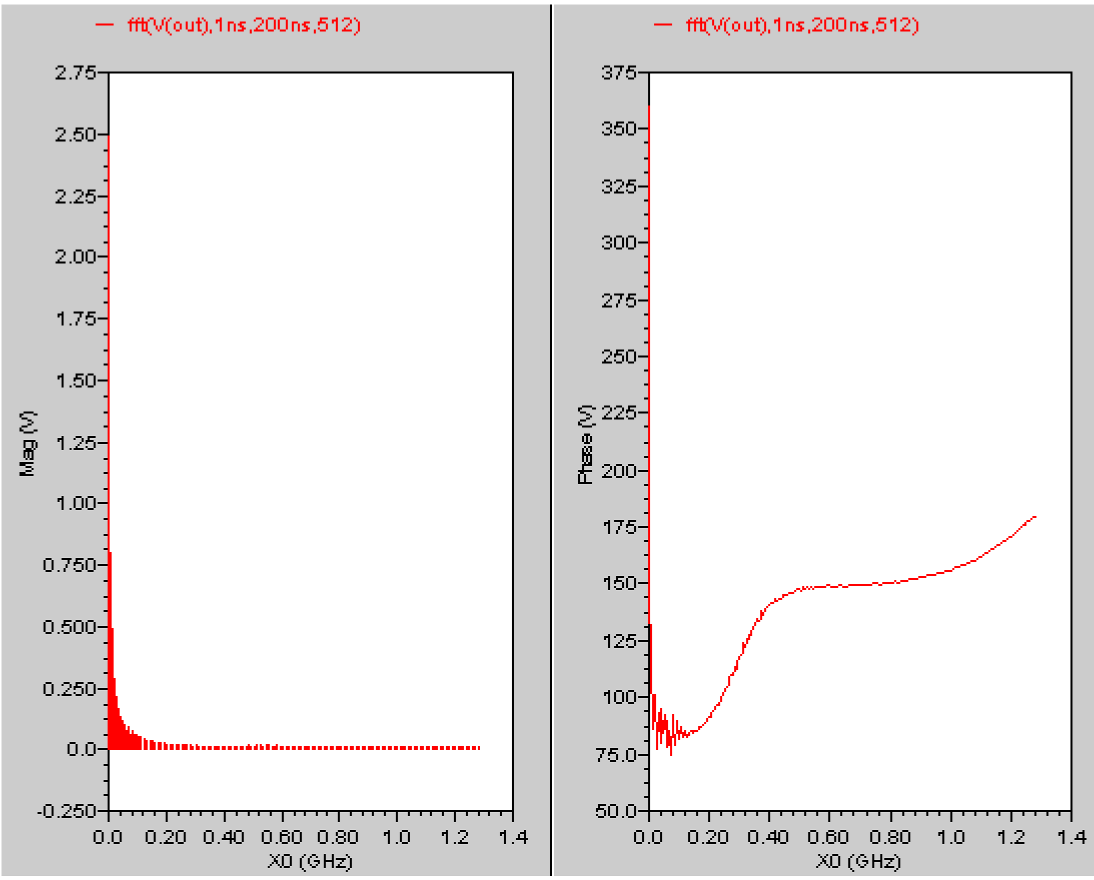
- *sig* is the name of the signal.
- *from* is the starting X value.
- `to` is the ending X value.
- *numPoints* is the number of data points to be used for calculating the fft. If this number is not a power of 2, it is automatically raised to the next higher power of 2.
- *window* is the algorithm used for calculating the fft. For more information, see `window`.

Example

The following input signal with the values *sig*=`V(out)`, f*rom*=`1ns`, *to*=`200ns`, *numpoints*=`512`, and *window*=`'bartlett`



transforms into the following output signal. The left subwindow shows the magnitude part of the spectrum and the right subwindow shows the phase part.



ga

Returns the available gain. This function is available only in the SKILL mode.

gac_freq

Returns the available power gain circles where the gain is fixed and frequency is swept. This function is available only in the SKILL mode.

- *Gain (dB)* is the specified gain.
- *Start* is the starting frequency.
- *Stop* is the ending frequency.
- *Step* is the frequency step size.

gac_gain

Returns the available power gain circles where the frequency is fixed and gain is swept. This function is available only in the SKILL mode.

- Frequency (Hz) is the specified frequency.
- *Start* is the starting gain.
- *Stop* is the ending gain.
- *Step* is the gain step size.

.

gmax

Returns the maximum available gain for a two port. This function is available only in the SKILL mode.

gmin

Returns the optimum noise reflection coefficient for NFmin. This function is available only in the SKILL mode.

gmsg

Returns the maximum stable power gain for a two port. This function is available only in the SKILL mode.

gp

Returns the power gain. This function is available only in the SKILL mode.

gpc_freq

Returns the operating power gain circles where the gain is fixed and frequency is swept. This function is available only in the SKILL mode.

- *Gain (dB)* is the specified gain.
- *Start* is the starting frequency.
- *Stop* is the ending frequency.
- *Step* is the frequency step size.

gpc_gain

Returns the operating power gain circles where the frequency is fixed and gain is swept. This function is available only in the SKILL mode.

- Frequency (Hz) is the specified frequency.
- *Start* is the starting gain.
- *Stop* is the ending gain.
- *Step* is the gain step size.

gt

Returns the transducer gain. This function is available only in the SKILL mode.

gumx

Returns the maximum unilateral power gain for a two port. This function is available only in the SKILL mode.

ifft

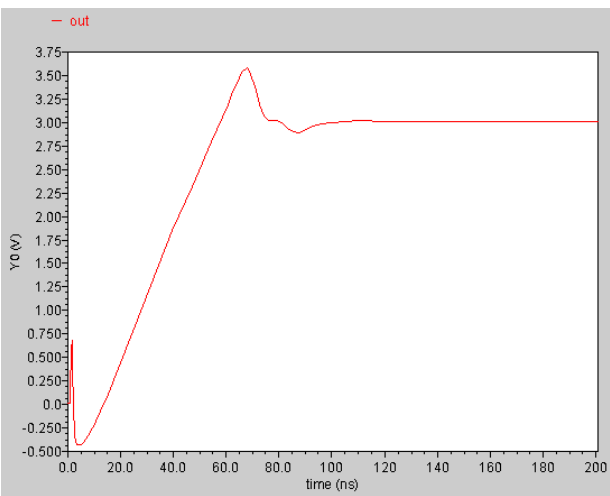
Performs an inverse Fast Fourier Transform on a frequency spectrum and returns the time domain representation of the spectrum. This function is available only in the MDL mode.

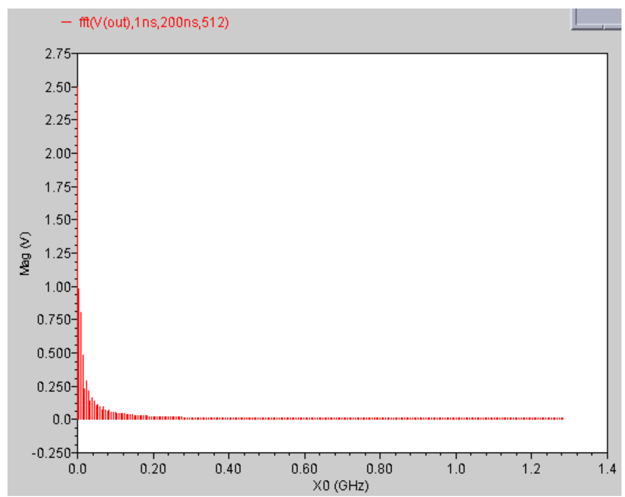
The frequency spectrum.

Example

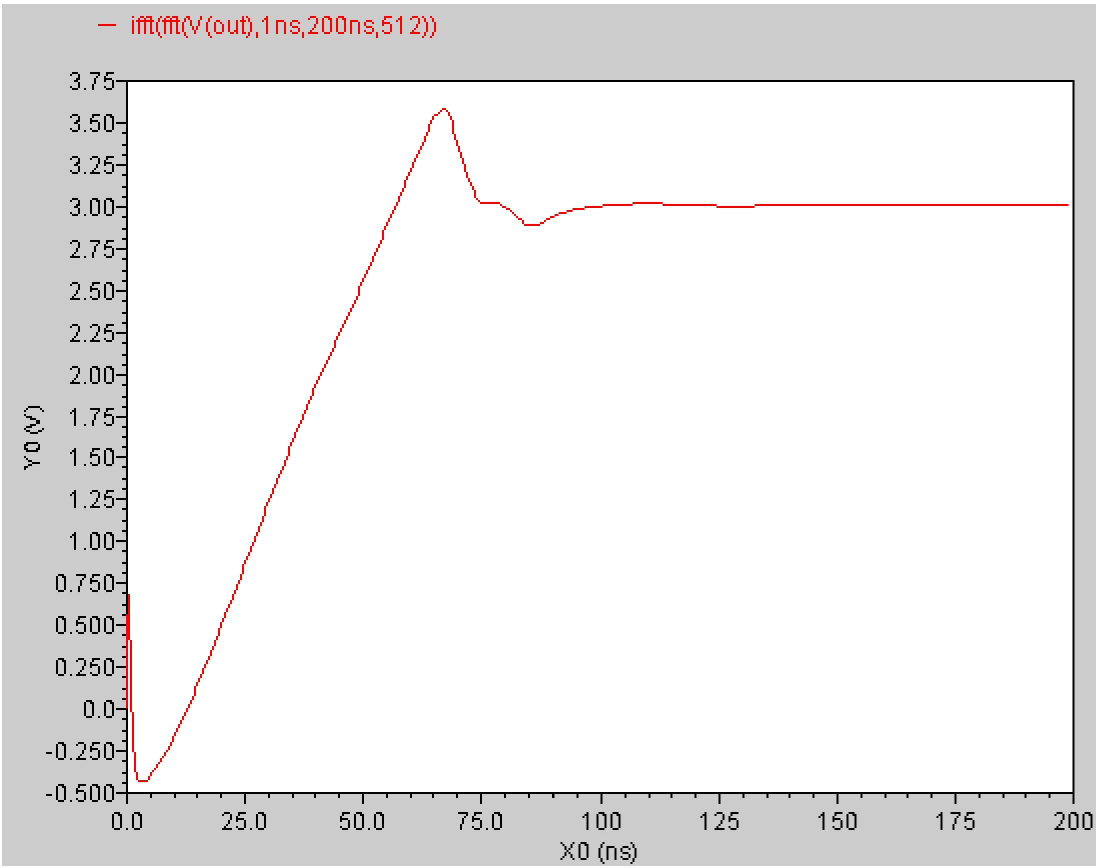
The input signal on the left side with the values *sig*=`V(out)`, *from*=`1ns`, *to*=`200ns`, *npoints*=`512` results in the graph on the right side.

The signal `out` | Fast fourier transform of the signal `out`





Now if you perform an `ifft` with the values *sig*=`V(out)`, *from*=`1`ns, *to*=`200`ns, *npoints*=`512`, the result is the same as the original signal (`out`) – from 1ns to 200ns.



im

Returns the imaginary part of a complex number. This function is available only in the MDL mode.

kf

Returns the stability factor K. This function is available only in the SKILL mode.

loadpull

Plots load pull contour for the given waveform of PSS analysis. This function works only for two-dimensional sweep PSS results. The inner sweep should be phase and the outer sweep should be magnitude.

- *Signal* is the name of the input waveform.
- *Max Value* is the largest value of the contour to be drawn. Default value is `nil`, which specifies that the largest value is to be taken from the results.
- *Min Value* is the smallest value of the contour to be drawn. Default value is `nil`, which specifies that the smallest value is to be taken from the results.
- *Number of Contours* is the number of points on the contour. Default value is `9`.
- *Close Contour* is *a* Boolean flag that specifies if a closed or open contour is to be drawn. If this field is set to `yes`, it specifies that a closed contour is to be drawn. If set to `no`, it specifies that an open contour is to be drawn.

Example: `awvRfLoadPull(i("V2:p" ?maxValue nil ?minValue nil ?numCont 9 ?closeCont nil)`

lsb (Load Stability Circles)

Returns the load stability circles. This function is available only in the SKILL mode.

- *Start (Hz)* is the start of the frequency range.
- *Stop (Hz)* is the end of the frequency range.
- *Step* is the increment for the frequency range.

max

Returns the absolute value of a signal, or the maximum value of two real values. This function is available only in the MDL mode.

min

Returns the minimum value of a signal. This function is available only in the MDL mode.

mod

Returns the floating point remainder of the dividend divided by the divisor. The divisor cannot be zero. This function is available only in the MDL mode.

movingavg

Calculates the moving average for the specified signal. This function is available only in the MDL mode.

nc_freq (Noise Circles - Sweep Frequency)

Returns noise circles with fixed gain. This function is available only in the SKILL mode.

- Level (dB)
- *Start* is the starting frequency.
- *Stop* is the ending frequency.
- *Step* is the frequency step size.

nc_gain (Noise Circles - Sweep Level)

Returns noise circles with fixed frequency. This function is available only in the SKILL mode.

- Level (dB)
- *Start* is the starting gain.
- *Stop* is the ending gain.
- *Step* is the gain step size.

nf

Retrieves F from the PSF file. This function is available only in the SKILL mode.

nf=dB10(F)

where nf is the noise figure and F is the noise factor.

nfmin

Retrieves Fmin from the PSF file. This function is available only in the SKILL mode.

nfmin=dB10(Fmin)

where nfmin is the minimum noise figure and Fmin is the minimum noise factor.

phaseDeg

Calculates the wrapped phase in degrees of a waveform and returns a waveform.

Example: `phaseDeg(v("net9" ?result "tran"))`

phaseDegUnwrapped

Calculates the unwrapped phase in degrees of a waveform and returns a waveform.

Example: `phaseDegUnwrapped(v("net9" ?result "tran"))`

phaseRadUnwrapped

Calculates the unwrapped (continuous) phase in radians of a waveform and returns a waveform.

Example: `phaseRadUnwrapped(v("net9" ?result "tran"))`

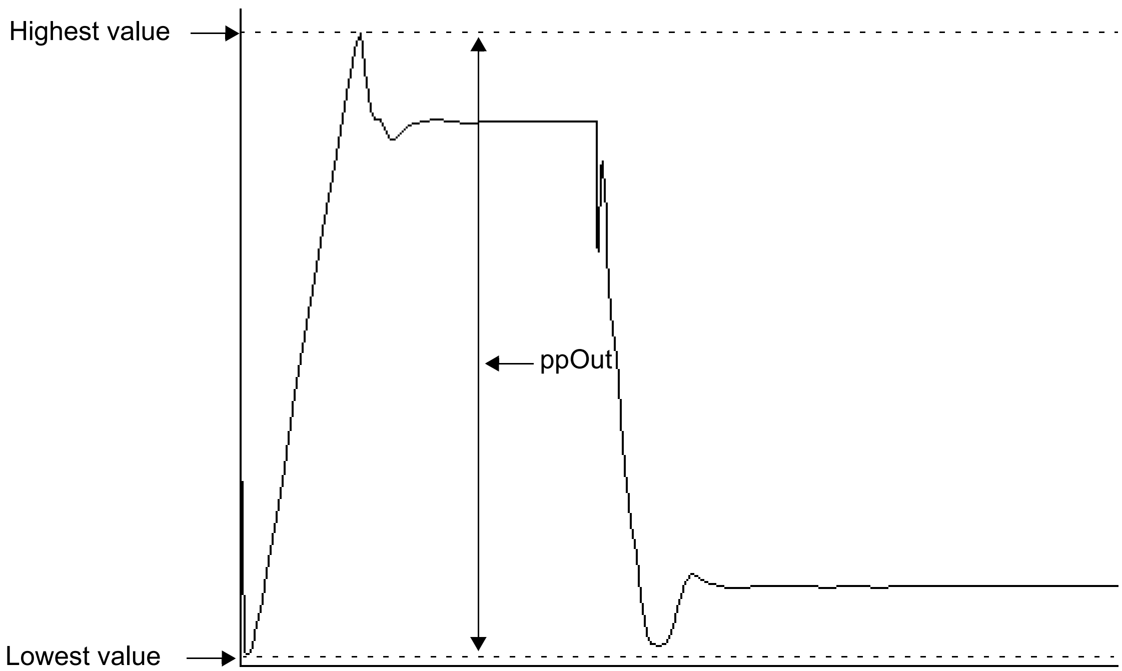
Takes the input waveform, representing the voltage of `net9`, and returns the waveform object representing the unwrapped phase in degrees.

pp (peak-to-peak)

Returns the difference between the highest and lowest values of a signal. This function is available only in the MDL mode.

Example 1

The following diagram illustrates how the pp value is determined.



r2d (radians-to-degrees)

Converts a scalar or waveform expressed in radians to degrees. This function is available only in the MDL mode.

re

Returns the real portion of a complex number. This function is available only in the MDL mode.

rmsVoltage

Computes the root-mean-square voltage between two nets for fast and regular envelop analysis. You must specify the name of at least one net for reference. If the name of the second net is not specified, by default, *gnd* is considered as the other net for reference.

rn

Returns the normalized equivalent noise resistance. This function is available only in the SKILL mode.

round

Rounds a number to the closest integer value. This function is available only in the MDL mode.

s11

Returns 2-port S-parameters. This function is available only in the SKILL mode.

s12

Returns 2-port S-parameters. This function is available only in the SKILL mode.

s21

Returns 2-port S-parameters. This function is available only in the SKILL mode.

s22

Returns 2-port S-parameters. This function is available only in the SKILL mode.

sign

Returns a value that corresponds to the sign of a number. This function is available only in the MDL mode.

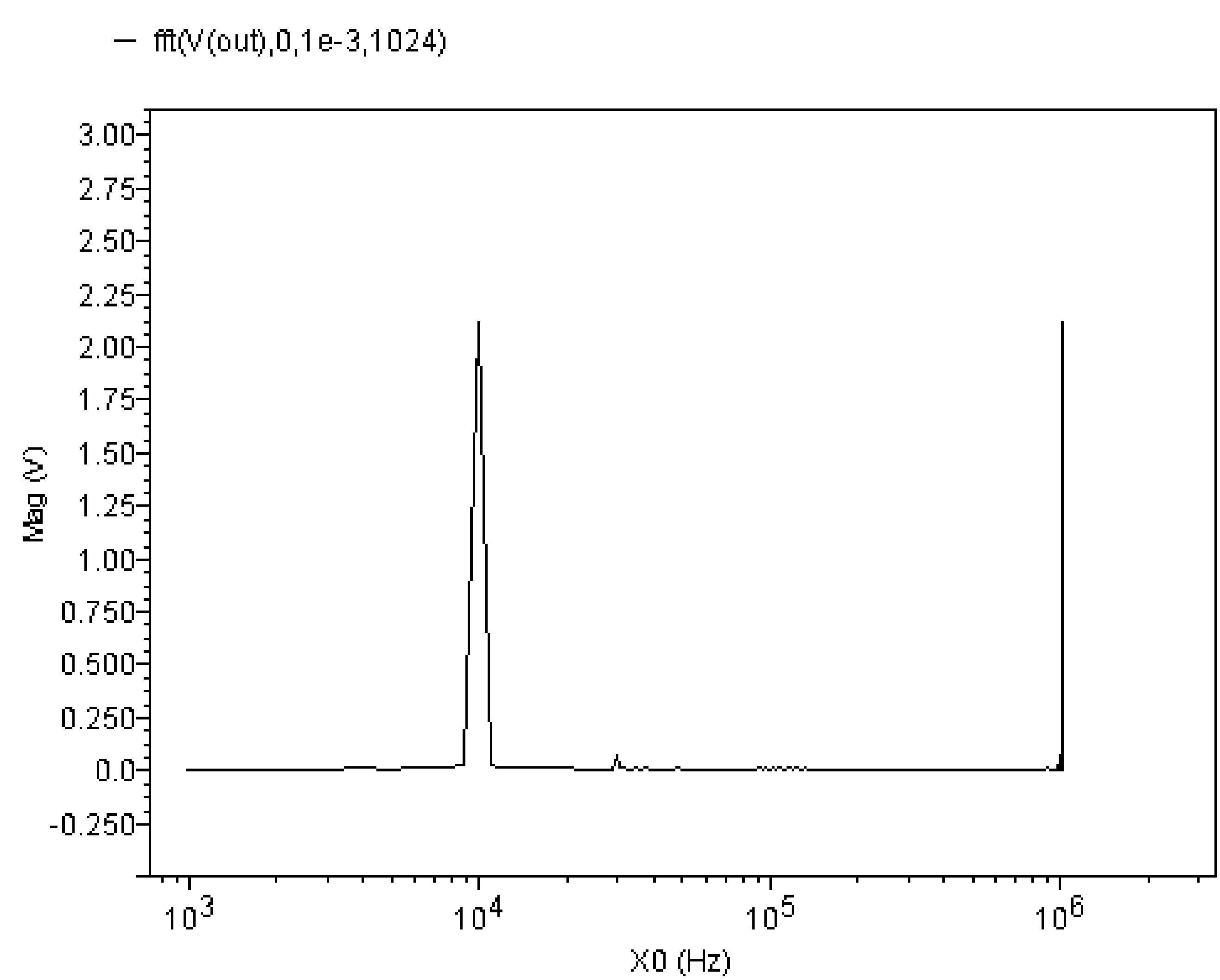
snr

Calculates the signal to noise ratio from a complex frequency based signal. This function is available only in the MDL mode.

- *sig* is the name of the signal.
- *sig_from* is the left window border of the signal. The *sig_from* value must be greater than or equal to *noise_from*.
- *sig_to* is the right window border of the signal. The *sig_to* value must be less than or equal to *noise_to*.
- *noise_from* is the left window border of the noise.
- *noise_to* is the right window border of the noise.

Example

You have the following frequency plot.



To determine the signal-to-noise ratio, you use the values

```
export real snr(fft(V(out),0,1e-3,1024),9e3,11e3,1,500e3)
```

which, in this case, returns

```
29.268026738835342dB
```

stathisto

Creates a histogram from a signal. This function is available only in the MDL mode.

The stathisto function is available from the calculator. It is not supported within a Spectre MDL control file since it returns a scalar and not a waveform.

- *sig* is the waveform.
- *nbins* is the number of bins to be created.
- *min* is the value that specifies the smaller end point of the range of values included in the histogram.
- *max* is the value that specifies the larger end point of the range of values included in the histogram.
- *innerswpval* is the inner-most sweep parameter in the dataset. You use this parameter to slice through parametric waveforms to extract the data for the histogram.
  Default: The first available value of time in the dataset.

Example

Assume that you have the results of running a Monte Carlo analysis on top of a transient analysis, so that the inner-most swept variable is time. Now, for the particular value of time specified by the `innerswpval` argument specification, the `stathisto` function creates a histogram by analyzing all the Monte Carlo iterations and extracting from each one the value of the signal at the specified time.

For example, to create a histogram for the time 100ns, you might use the following statement.

```
stathisto(I(V10\:p),innerswpval=100e-9)
```

To create a histogram for the time 650ps, you might use the following statement.

```
stathisto(I(V10\:p),innerswpval=.65e-9)
```

trim

Returns the portion of a signal between two points along the X-axis. This function is available only in the MDL mode.

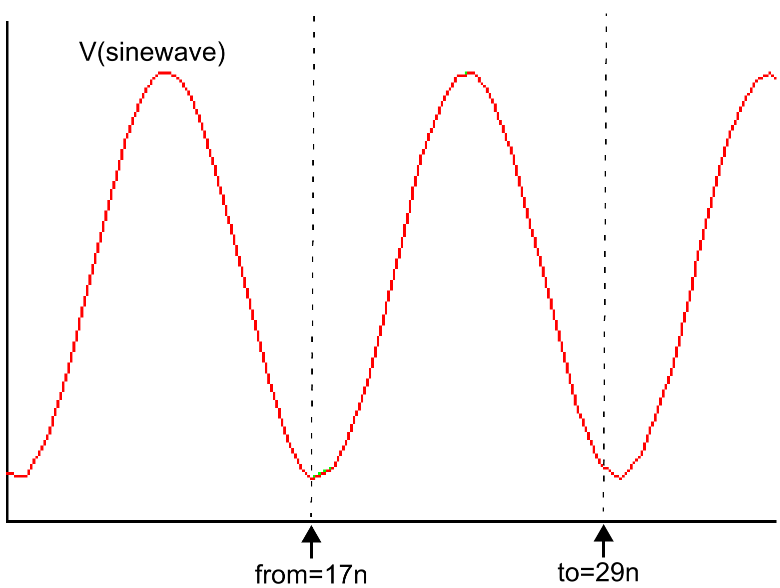
- *sig* is the name of the signal.
- *from* is the starting point on the X-axis.
- *to* is the ending point on the X-axis.

Example 1

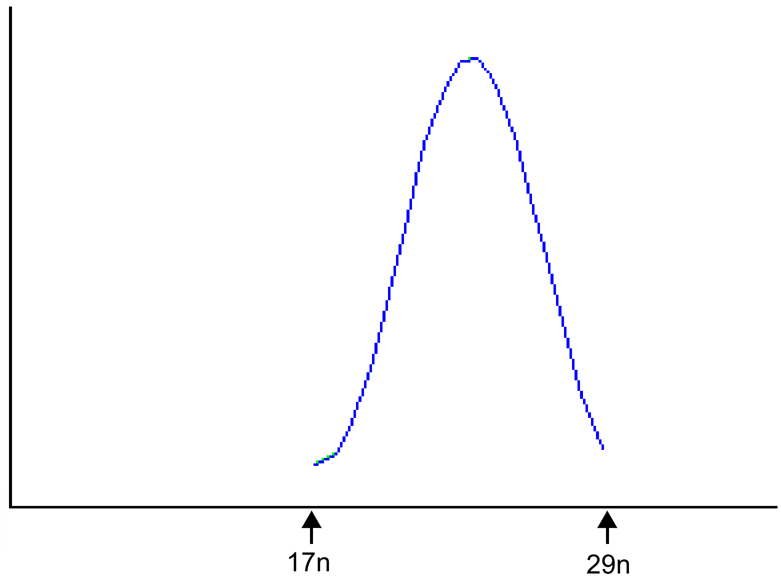
In Virtuoso Visualization and Analysis XL,

```
trim ( sig=V(sinewave), from=17n, to=29n )
```

transforms the following input signal



into the following output signal



window

Applies the specified window to a signal. This function is available only in the MDL mode.

- *arg* is the name of the signal.
- *window* is the window to be applied.

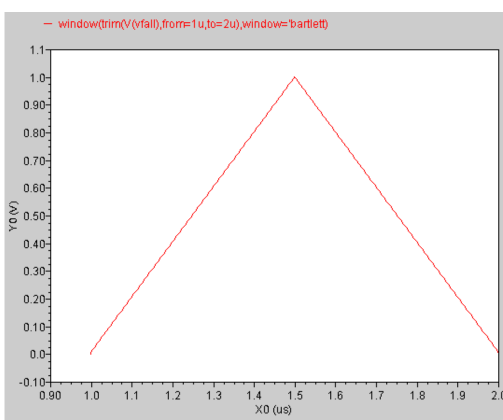
Equations and Examples

This section describes the equations used by each type of window and then shows an example. In the equations:

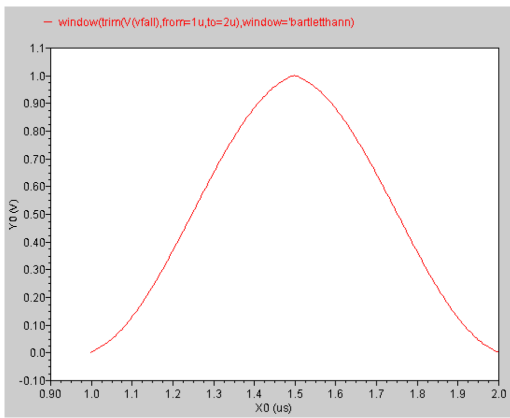
N = total number of waveform points

n = current waveform point

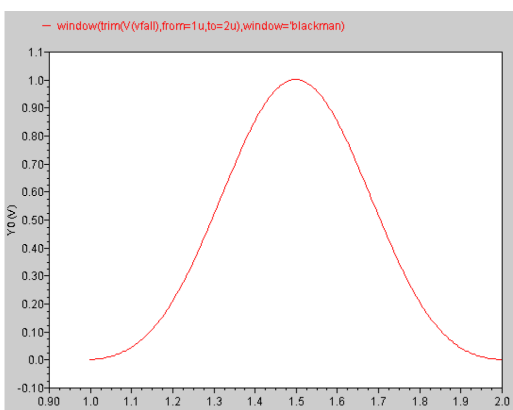
Window	Equation and Example	Where
`'rectangular`	$w(n) = 1$	
`'bartlett`	$w(n) = 1 - abs\left(2 \times \frac{n}{N} - 1\right)$	$0 \le n \le N$
	$w(n) = 0$	otherwise



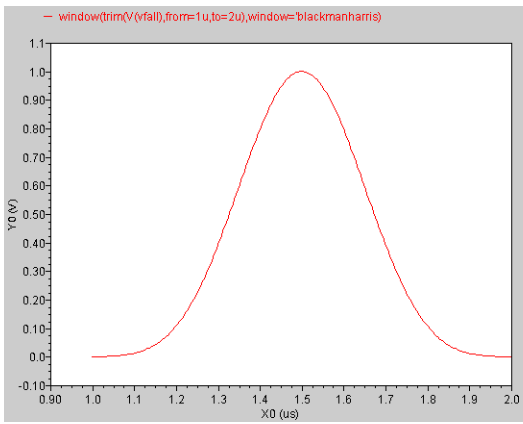
Window	Equation and Example	Where
`'bartletthann`	$w(n) = 0.62 - 0.48 \times abs\left(\frac{n}{N} - 0.5\right) + 0.38 \times \cos\left(2 \times {}'pi \times \left(\frac{n}{N} - 0.5\right)\right)$	$0 \le n \le N$
	$w(n) = 0$	otherwise



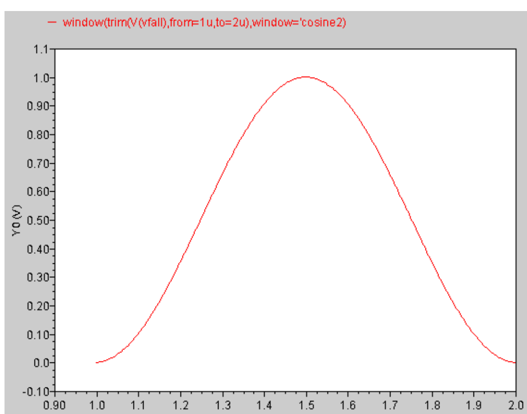
Window	Equation and Example	Where
`'blackman`	$'(n) = 0.42 - 0.50 \times \cos\left(2 \times {}'pi \times \frac{n}{N}\right) + 0.08 \times \cos(4 \times {}'pi \times \frac{n}{N}$	$0 \le n \le N$
	$w(n) = 0$	otherwise



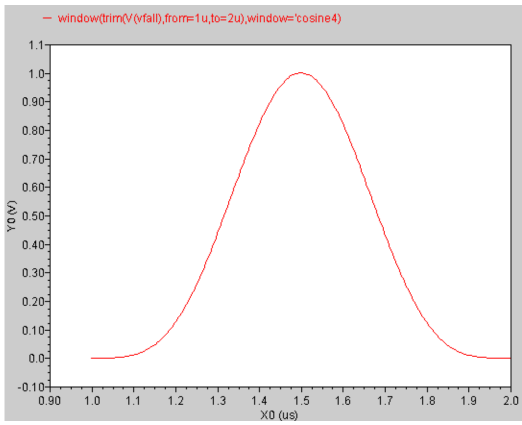
Window	Equation and Example	Where
`'blackmanharris`	$w(n) = 0.35875 - 0.48829 \times \cos\left(2 \times ' pi \times \frac{n}{N}\right) + 0.14128 \times \cos\left(4 \times ' pi \times \frac{n}{N}\right) + -0.01168 \times \cos\left(6 \times ' pi \times \frac{n}{N}\right)$	$0 \le n \le N$
	$w(n) = 0$	otherwise



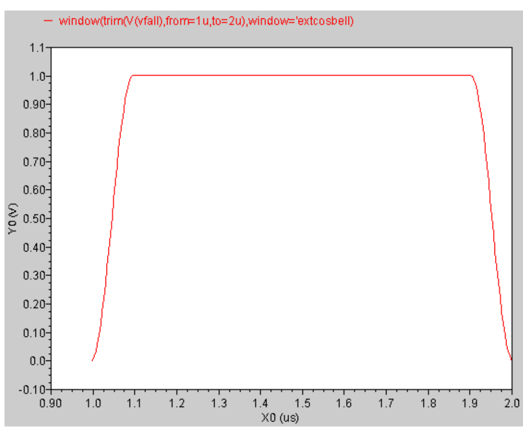
Window	Equation and Example	Where
`'cosine2`	$w(n) = 0.5 - 0.5 \times \cos\left(2 \times ' pi \times \frac{n}{N}\right)$	$0 \le n \le N$
	$w(n) = 0$	otherwise



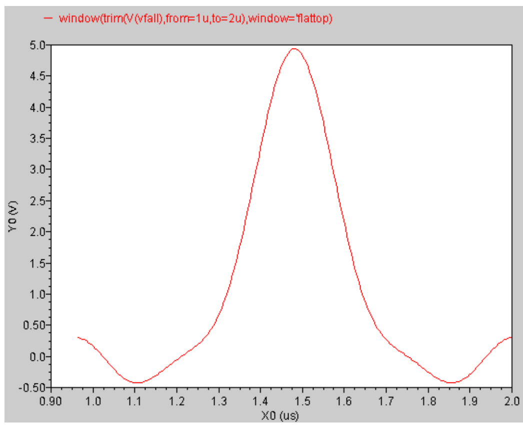
Window	Equation and Example	Where
`'cosine4`	$v(n) = \left(0.5 - 0.5 \times \cos\left(2 \times {'}pi \times \frac{n}{N}\right)\right)^2$	0≤n≤N
	$w(n) = 0$	otherwise



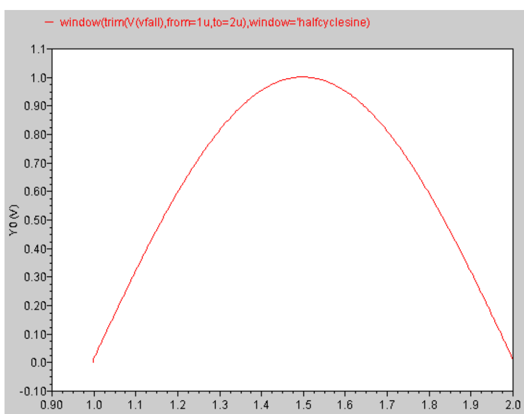
Window	Equation and Example	Where
`'extcosbell`	$w(n) = 0.5 - 0.5 \times \cos\left(10 \times {'}pi \times \frac{n}{N}\right)$	abs(n/N - 0.5) > 0.4
	$w(n) = 1$	otherwise



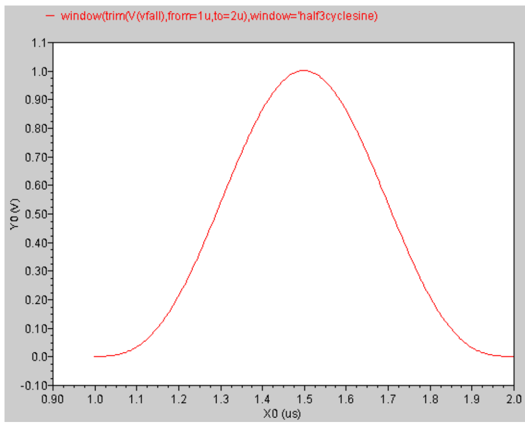
Window	Equation and Example	Where
`'flattop`	$w(n) = 1 - 1.93 \times \cos\left(2 \times 'pi \times \frac{n}{N}\right) + 1.29 \times \cos\left(4 \times 'pi \times \frac{n}{N}\right) - 0.388$ $\times \cos\left(6 \times 'pi \times \frac{n}{N}\right) + 0.322 \times \cos\left(8 \times 'pi \times \frac{n}{N}\right)$	0≤n≤N
	$w(n) = 0$	otherwise



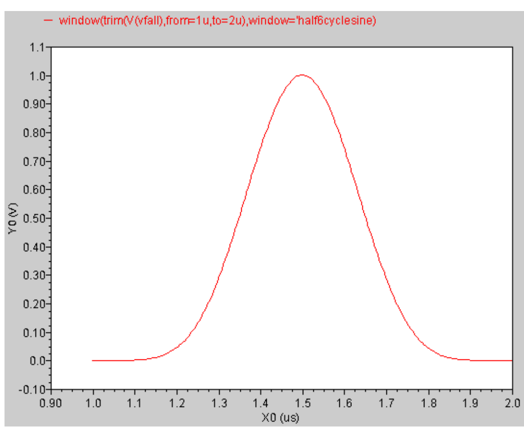
Window	Equation and Example	Where
`'halfcyclesine`	$w(n) = \sin\left('pi \times \frac{n}{N}\right)$	0≤n≤N
	$w(n) = 0$	otherwise



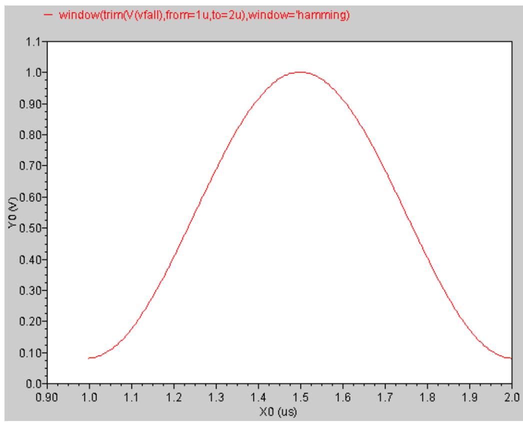
Window	Equation and Example	Where
`'half3cyclesine` and `'halfcyclesine3`	$w(n) = \left(\sin\left('pi \times \frac{n}{N}\right)\right)^3$	$0 \le n \le N$
	$w(n) = 0$	otherwise



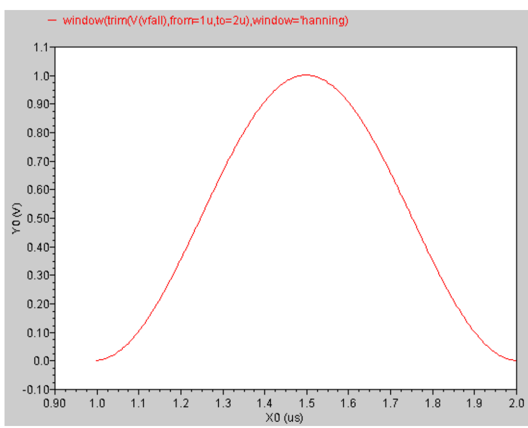
Window	Equation and Example	Where
`'half6cyclesine` and `'halfcyclesine6`	$w(n) = \left(\sin\left('pi \times \frac{n}{N}\right)\right)^6$	$0 \le n \le N$
	$w(n) = 0$	otherwise

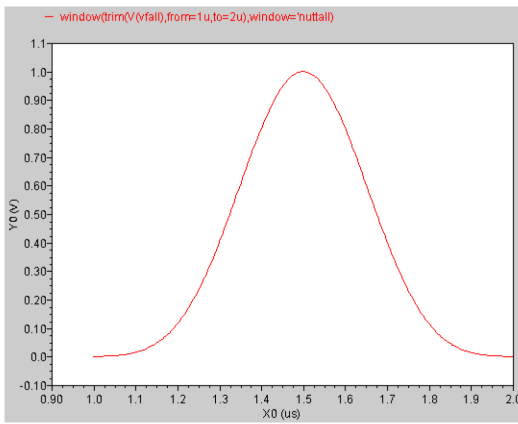
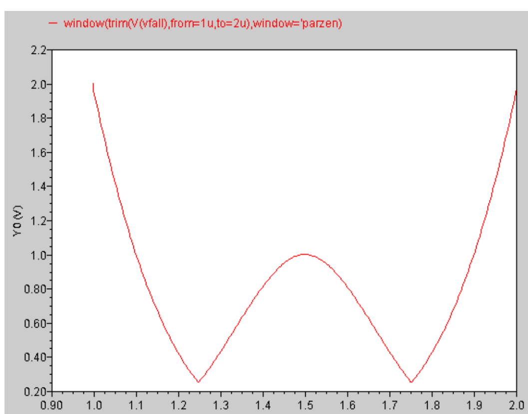


Window	Equation and Example	Where
`'hamming`	$w(n) = 0.54 - 0.46 \times \cos\left(2 \times 'pi \times \frac{n}{N}\right)$	$0 \le n \le N$
	$w(n) = 0$	otherwise



Window	Equation and Example	Where
`'hanning`	$w(n) = 0.5 - 0.5 \times \cos\left(2 \times 'pi \times \frac{n}{N}\right)$	$0 \le n \le N$
	$w(n) = 0$	otherwise



Window	Equation and Example	Where
`'nuttall`	$w(n) = 0.3635819 - 0.4891775 \times \cos\left(2 \times {'}pi \times \frac{n}{N}\right) + 0.1365995 \times \cos\left(4 \times {'}pi \times \frac{n}{N}\right) - 0.0106411 \times \cos\left(6 \times {'}pi \times \frac{n}{N}\right)$	0≤n≤N
	$w(n) = 0$	otherwise
		
`'parzen`	$w(n) = 1 - 6 \times abs\left(2 \times \frac{n}{N} - 1\right) + 6 \times abs\left(2 \times \frac{n}{N} - 1\right)$	abs(2*n/N-1)≤0.5
	$w(n) = 2 \times abs\left(2 \times \frac{n}{N} - 1\right)$	otherwise
		
`'triangular`	Same as `'bartlett`.	

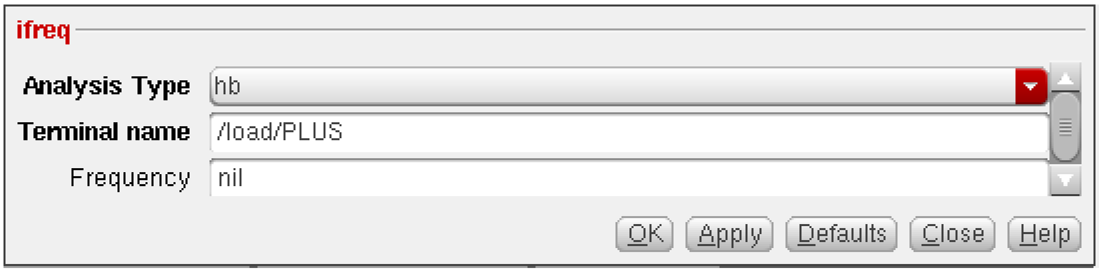
Spectre RF Functions

This section describes the following Spectre RF functions in the Calculator:

- ifreq
- ih
- itime
- pir
- pmNoise
- pn
- pvi
- pvr
- spm
- totalNoise
- vfreq
- vh
- vtime
- ypm
- ypm

ifreq

Returns the current of the terminal at a specified frequency or at all frequencies in the frequency domain.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `hb`, `pss`, `qpss`, `pac`, `hbac`, `qpac`, and `ac`.
  Default value: `hb`
- *Terminal name*—Specify a terminal name on the schematic or signal name from the Results Browser.

  When you open Calculator from ADE, you can select the terminal name directly from the Schematic window. The selected terminal name appears in this field.

  If you are working in the standalone mode and building expressions by using the Results Browser, you can get the terminal name from the Results Browser:

  - Select the *Wave* radio button on the Selection toolbar.
  - Click a current signal in the Results Browser.

    The selected signal appears in the *Terminal name* field.
- *Frequency*—Specify the frequency for which you want to plot the results. It is an optional field.
  Valid values: Any integer or floating point number.
  Default value: `nil`.

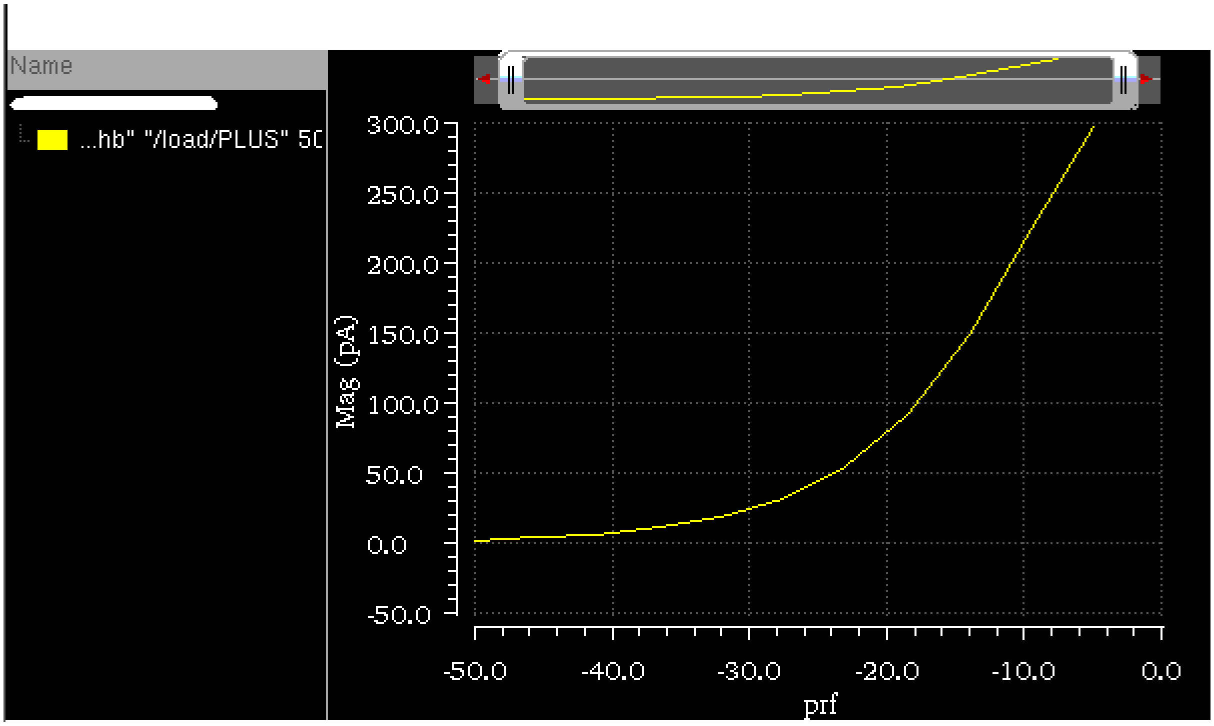
Example

This example shows the output plot generated if you build the following expression in the Buffer:

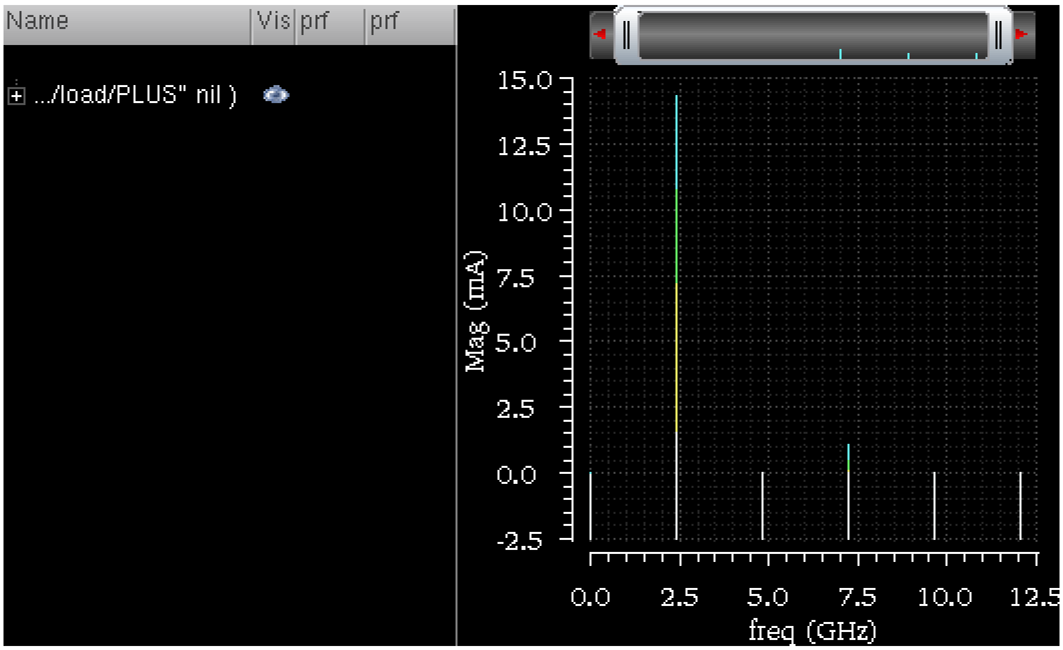
```
ifreq("hb" "/load/PLUS" 50 )
```

Here,

- *Analysis Type* is `hb`
- *Terminal name* is `/load/PLUS`
- *Frequency* is `50`



If you build the above expression with *Frequency*=`nil`, the current on all the frequency points are returned (as shown in the figure below).



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the current at a specified frequency or at all frequency points:

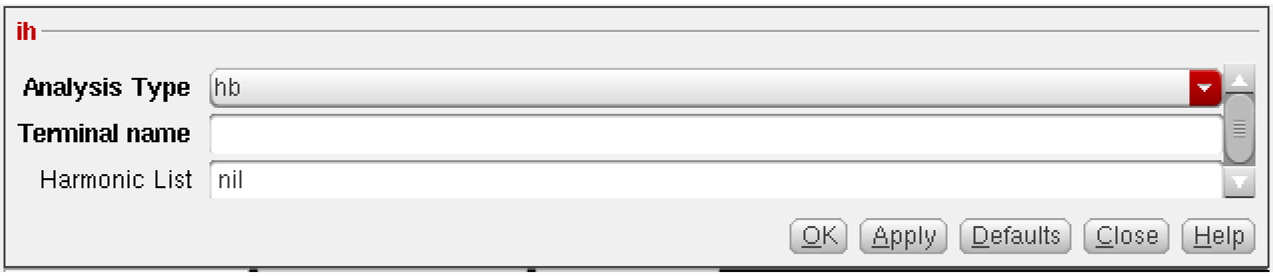
```
i("/load/PLUS" ?result "hb_fd")
```

However, the similar calculation is now performed with the *ifreq* function and the following expression is created in the Buffer:

```
ifreq("hb" "/load/PLUS" nil )
```

ih

Returns the current of the terminal for a specified harmonic or for the harmonic list in the frequency domain.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `hb`, `pss`, `qpss`, `pac`, `hbac`, and `qpac`.
  Default value: `hb`

- *Terminal name*— Specify the terminal name on the schematic or signal name from the Results Browser.

  When you open Calculator from ADE, you can select the terminal name directly from the Schematic window. The selected terminal name appears in this field.

  If you are working in the standalone mode and building expressions by using the Results Browser, you can get the terminal name from the Results Browser:

  - Select the *Wave* radio button on the Selection toolbar.
  - Click a current signal in the Results Browser.

    The selected signal appears in the *Terminal name* field.

- *Harmonic List*—Specify the harmonics for which you want to plot the results. It is an optional field. For analyses, such as `hb`, `pss`, `pac`, and `hbac`, you can add either single harmonic value or available list of harmonic values in this field.
  Valid values: Any integer or a list from the available list of harmonics. You can find the available harmonics by using the `harmonicList` function.
  `harmonicList( [?resultsDir t_resultsDir] [?result S_resultName]`
  Default value: `nil`.

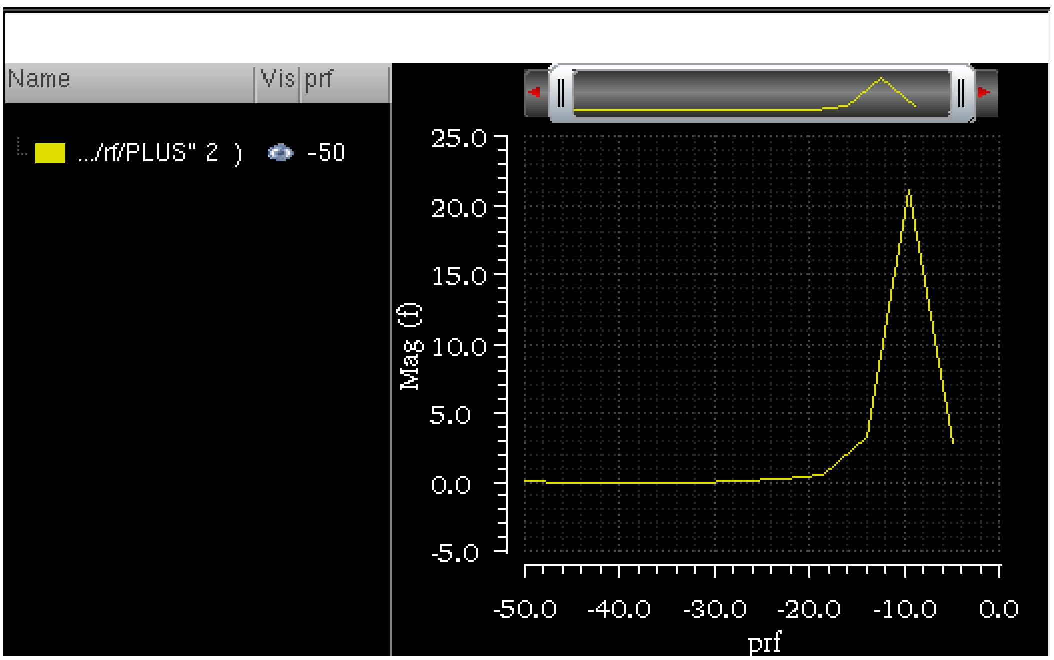
Example

This example shows the output plot generated if you build the following expression in the Buffer:

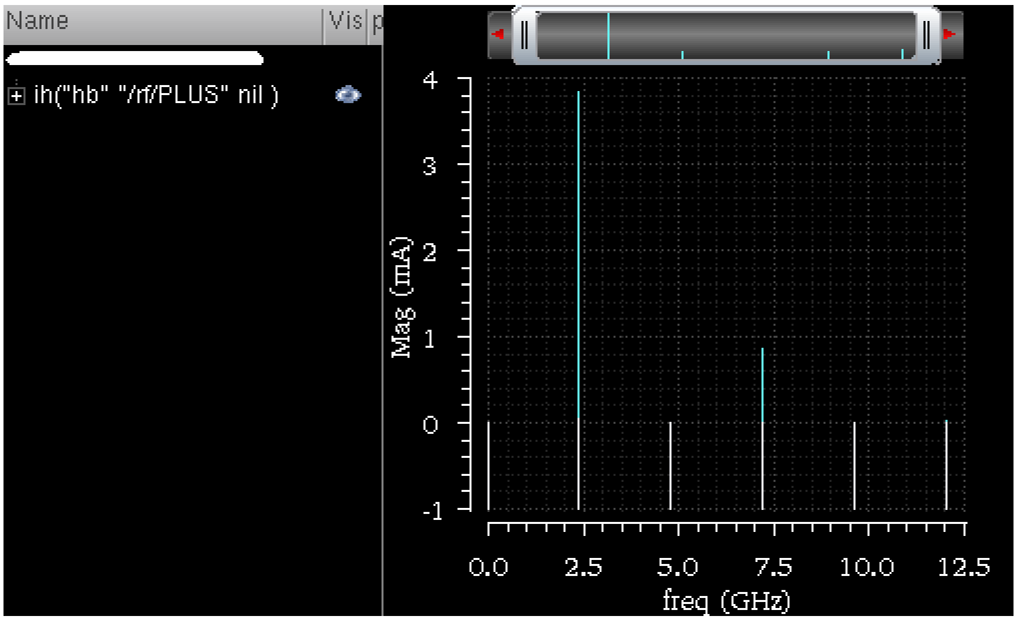
```
ih("hb" "/rf/PLUS" 2 )
```

Here,

- *Analysis Type* is `hb`
- *Terminal name* is `/rf/PLUS`
- *Harmonic List* is `2`



If you build the above expression with *Harmonic List*=`nil`, the current on all the harmonic points are returned (as shown in the figure below).



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the current at a specified harmonic or at all harmonic points:

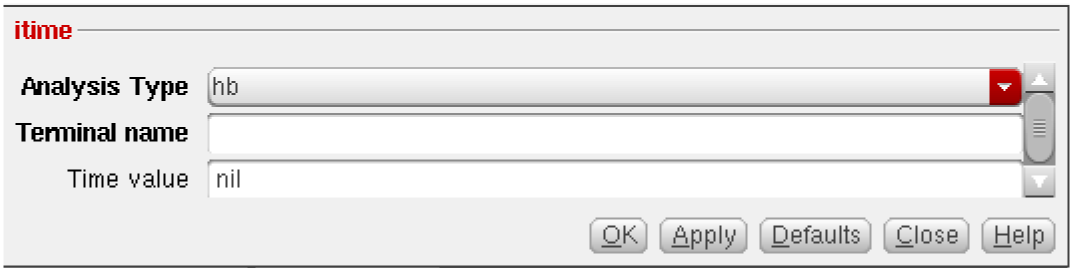
```
i("/load/PLUS" ?result "hb_fd")
```

However, the similar calculation is now performed with the *ih* function and the following expression is created in the Buffer:

```
ih("hb" "/load/PLUS" nil )
```

itime

Returns the current of the terminal at a specified time point or at all time points in the time domain.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `hb`, `pss`, and `tran`.
  Default value: `hb`
- *Terminal name*— Specify the terminal name on the schematic or signal name from the Results Browser.

  When you open Calculator from ADE, you can select the terminal name directly from the Schematic window. The selected terminal name appears in this field.

  If you are working in the standalone mode and building expressions by using the Results Browser, you can get the terminal name from the Results Browser:

  - Select the *Wave* radio button on the Selection toolbar.
  - Click a current signal in the Results Browser.

    The selected signal appears in the *Terminal name* field.
- *Time value*—Specify the time points for which you want to plot the results. If you specify a time point in this field, the result of the specified time is returned. Otherwise, result on all the time points are returned. It is an optional field.
  Valid values: Any integer or floating point number.
  Default value: `nil`.

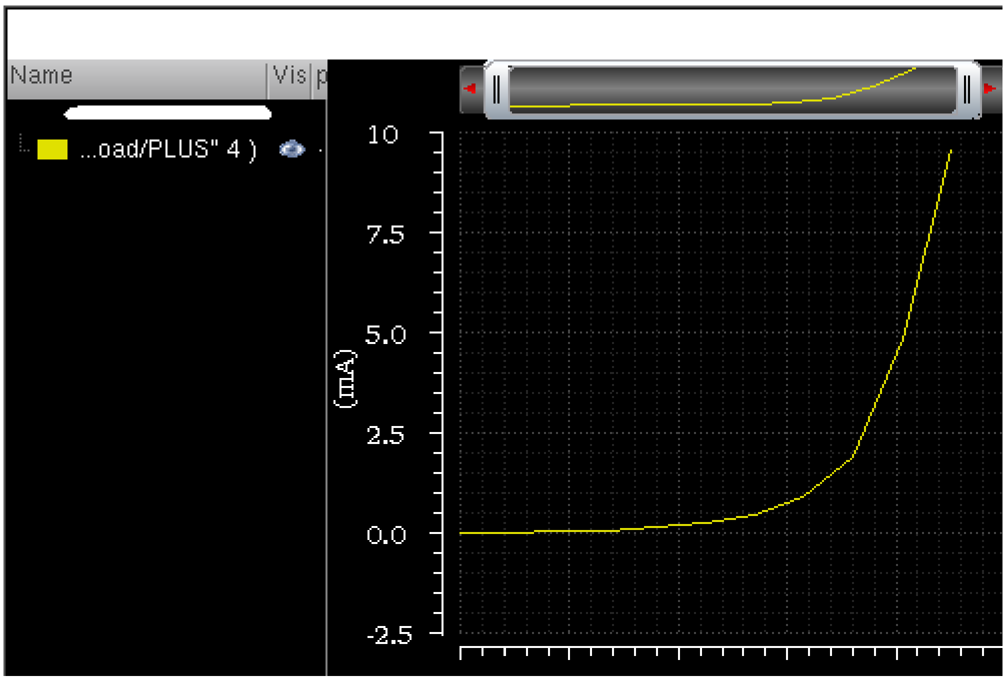
Example

This example shows the output plot generated if you build the following expression in the Buffer:

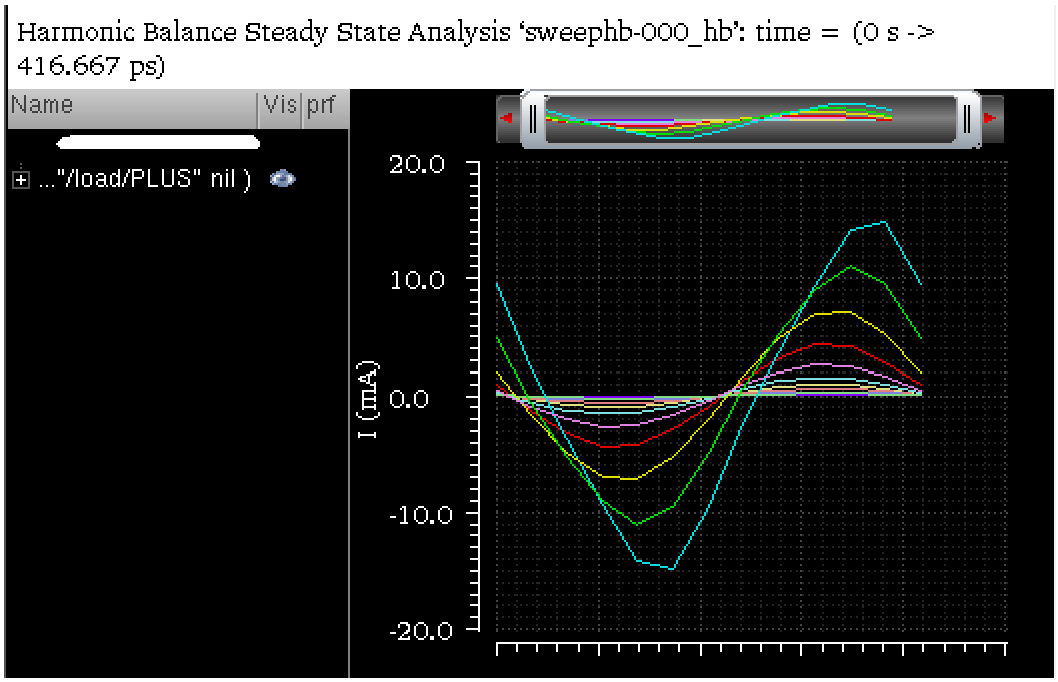
```
itime("hb" "/load/PLUS" 4 )
```

Here,

- *Analysis Type* is `hb`
- *Terminal name* is `/load/PLUS`
- *Time value* is `4`



If you build the above expression with *Time value*=`nil`, the current on all the time points are returned (as shown in the figure below).



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the current at a time point or at all time points:

```
i("/load/PLUS" ?result "hb_td")
```

However, the similar calculation is now performed with the *itime* function and the following expression is created in the Buffer:

```
itime("hb" "/load/PLUS" nil )
```

pir

Returns the spectral power from the current at two terminals and resistance for a specified harmonic list.

This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `hb`, `pss`, `qpss`, `pac`, `hbac` and `qpac`.
  Default value: `hb`

- *Branch name 1*— Specify the first branch name in the schematic or signal name from the Results Browser.

- *Branch name 2*—Specify the second branch name in the schematic or signal name from the Results Browser.

  When you open Calculator from ADE, you can select the branch name directly from the Schematic window. The selected branch name appears in the branch name fields.

  If you are working in the standalone mode and building expressions by using the Results Browser, you can get the branch name from the Results Browser:

  - Select the *Wave* radio button on the Selection toolbar.

  - Click a current signal in the Results Browser.

  The selected signal appears in the *Branch name* field.

- *Resistance*—Specify the resistance value. This field can contain any integer or floating point value. By default, it is set to blank.

- *Harmonic List*—Specify the harmonics for which you want to plot the results. It is an optional field. For analyses, such as `hb`, `pss`, `pac`, and `hbac`, you can add either single harmonic value or available list of harmonic values in this field.

Valid values: Any integer or a list from the available list of harmonics. You can find the available harmonics by using the `harmonicList` function.
`harmonicList( [?resultsDir t_resultsDir] [?result S_resultName]`
Default value: `nil`.

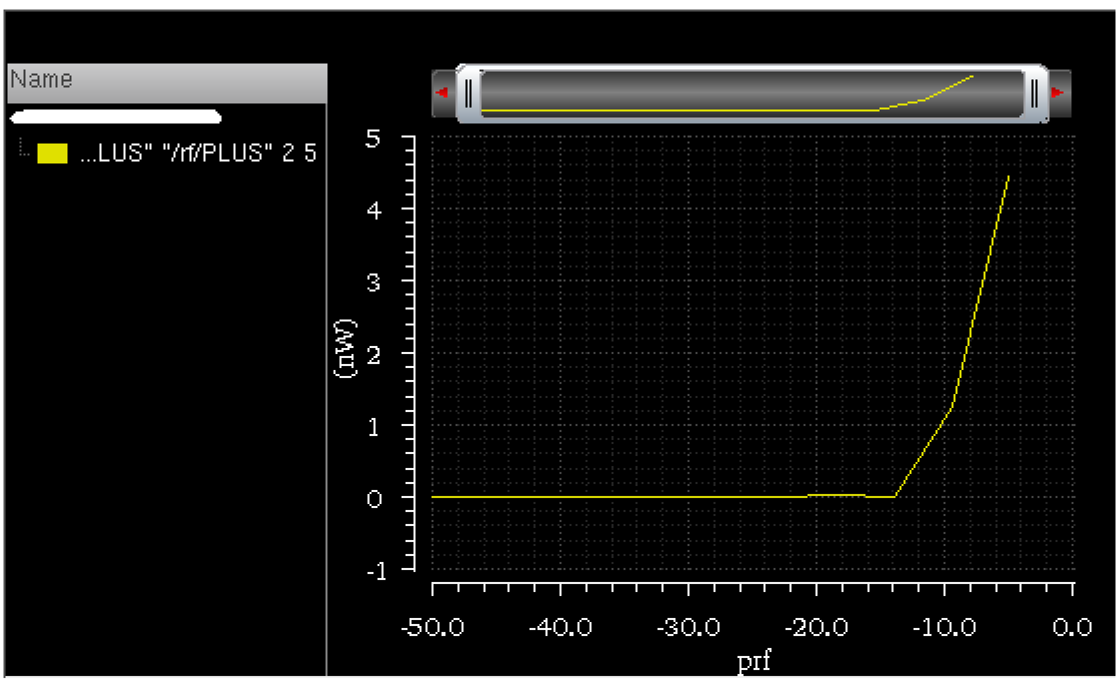
Example

This example shows the output plot generated if you build the following expression in the Buffer:

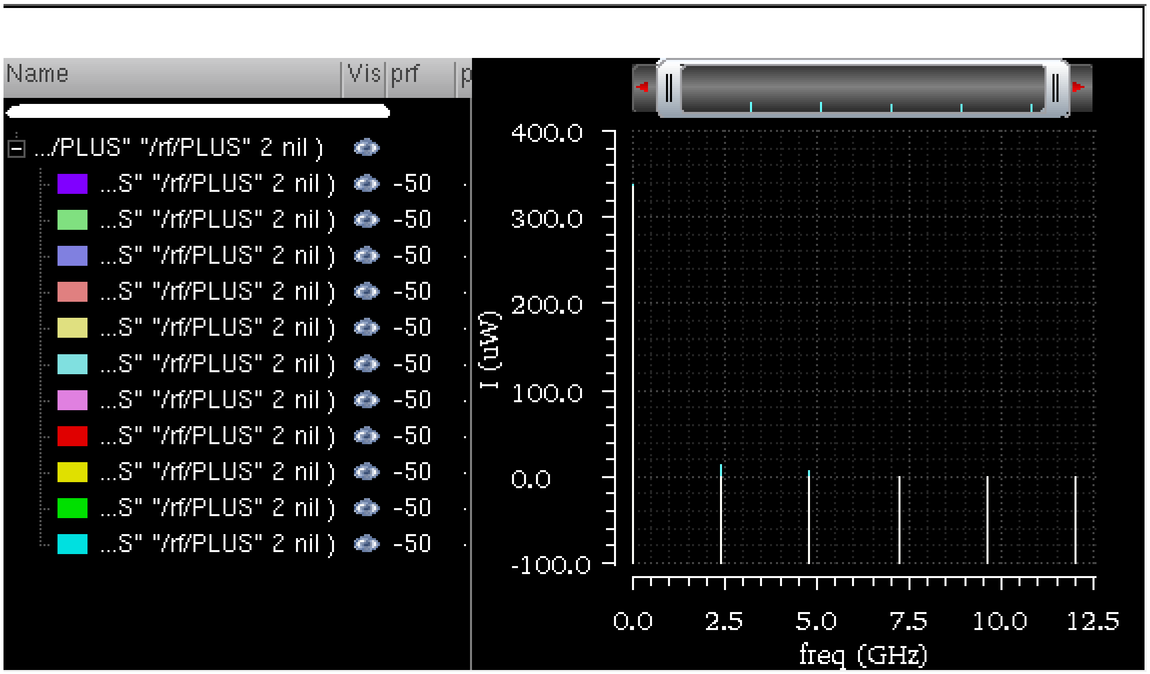
```
pir("hb" "/V1/PLUS" "/rf/PLUS" 2 5 )
```

Here,

- *Analysis Type* is `hb`
- *Branch name 1* is `/V1/PLUS`
- *Branch name 2* is `/rf/PLUS`
- *Resistance* is `2`
- *Harmonic List* is `5`



If you build the above expression with *Harmonic List*=`nil`, the spectral power on all the harmonic points are returned (as shown in the figure below).



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the spectral power from current and resistance for a specified harmonic list:

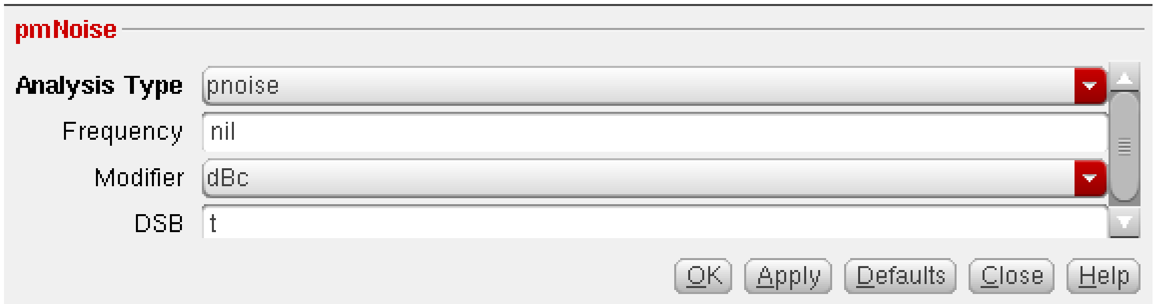
```
spectralPower(((i("/rf/PLUS" ?result "hb_fd") - i("/load/PLUS"
?result "hb_fd")) * 50) (i("/rf/PLUS" ?result "hb_fd") - i("/load/
PLUS" ?result "hb_fd")))
```

However, the similar calculation is now performed with the *pir* function and the following expression is created in the Buffer:

```
pir("hb" "/rf/PLUS" "/load/PLUS" 50 nil )
```

pmNoise

Returns the modulated phase noise for a specified frequency or for the entire spectrum.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `pnoise` and `hbnoise`.
  Default value: `pnoise`
- *Frequency*—Specify the frequency for which you want to calculate the phase noise. By default, this field is set to `nil`, which means the modulated phase noise at all frequency points are calculated.
- *Modifier*—Select the value of the modifier to be used.
  Valid values: `dBc`, `normalized`, `Power`, `Magnitude`, and `dBV`
  Default value: `dBc`
- *DSB*—Specify whether you want to include the double side band.
  Valid values: `t` and `nil`
  Default value: `t`

Example

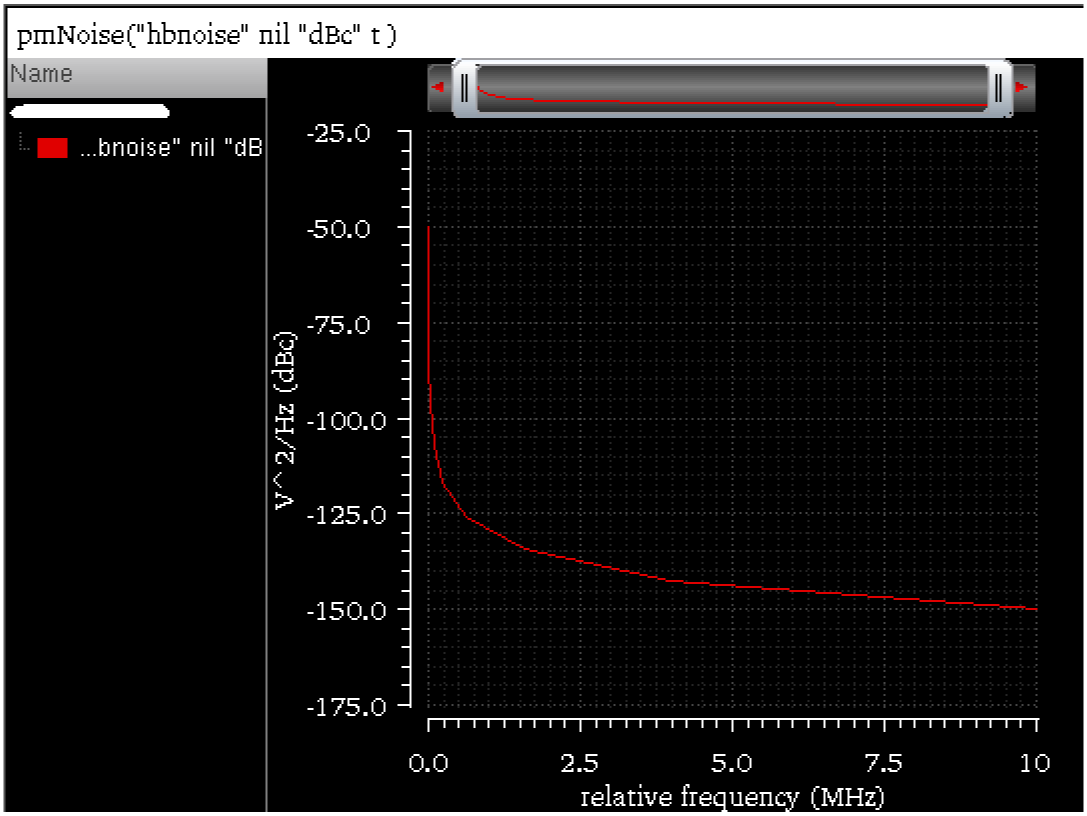
This example shows the output generated if you build the following expression in the Buffer:

```
pmNoise("hbnoise" 50 "dBc" t )
```

- *Analysis Type* is `hbnoise`
- *Frequency* is `50`
- *Modifier* is `dBc`
- *DSB* is `t`

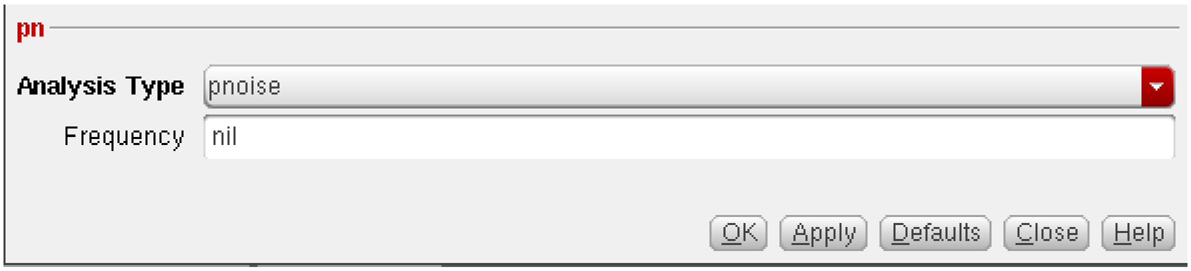
The output generated in the Buffer is `-50.0`, which is a scalar value.

Now, if you specify *Frequency*=`nil`, phase noise on all frequency points are returned as shown in the figure below:



pn

Returns the phase noise at a specified frequency or at all frequency points.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list.
  Valid values: `pnoise`, `hbnoise`, and `qpnoise`.
  Default value: `pnoise`
- *Frequency*—Specify the frequency for which you want to calculate the phase noise. By default, this field is set to `nil`, which means the frequency at all points are calculated.

Example

This example shows the output generated if you build the following expression in the Buffer:

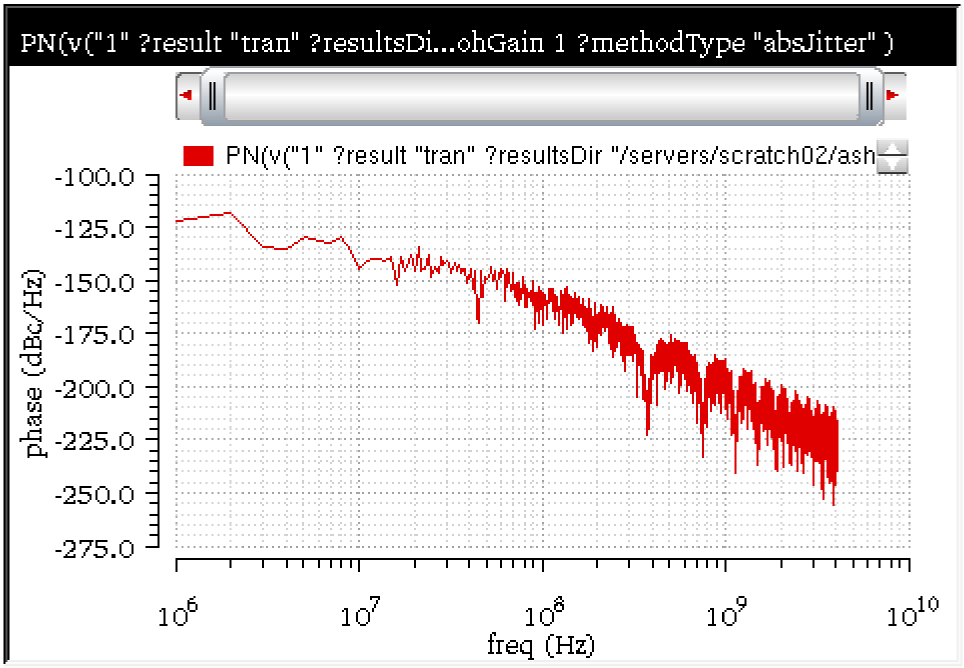
```
pn("hbnoise" 50 )
```

- *Analysis Type* is `hbnoise`
- *Frequency* is `50`

The output generated in the Buffer is `-53.01`, which is a scalar value.

Now, if you specify *Frequency*=`nil`, the phase noise on all frequency points are returned as shown in the figure below:

```
pn("hbnoise" nil )
```



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the phase noise at a specified frequency or at all frequency points:

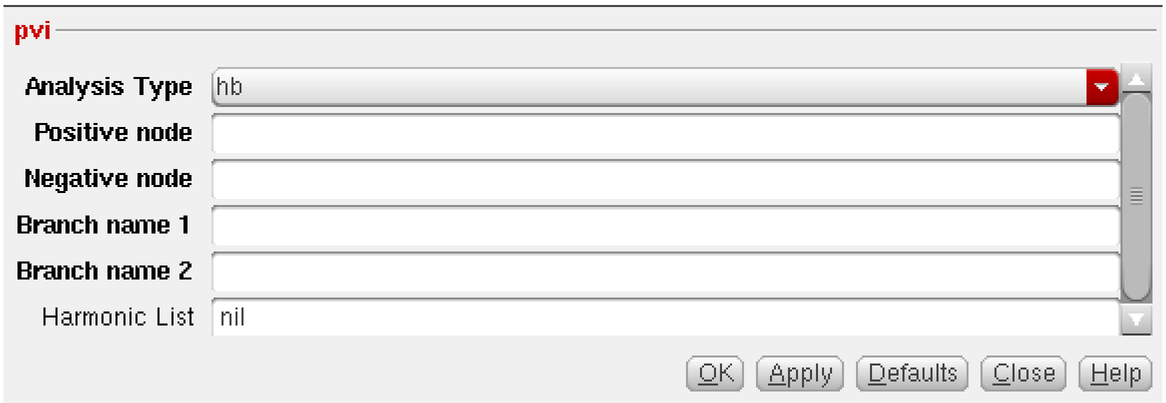
```
phaseNoise(3 "hb_fd" ?result "hbnoise")
```

However, the similar calculation is now performed with the *pn* function and the following expression is created in the Buffer:

```
pn('hbnoise)
```

pvi

Returns the spectral power voltage on the positive and negative nodes and the current at two terminals for a specified harmonic list or for all harmonics.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `hb`, `pss`, `qpss`, `pac`, `hbac` and `qpac`.
  Default value: `hb`
- *Positive node*—Specify the positive node or net. This field can also contain an explicit voltage value.
- *Negative node*—Specify the negative node or net. This field can also contain an explicit voltage value.
- *Branch name 1*— Specify the first branch name in the schematic or signal name from the Results Browser.
- *Branch name 2*—Specify the second branch name in the schematic or signal name from the Results Browser.

  When you open Calculator from ADE, you can select the node and branch name directly from the Schematic window. The selected node or branch name appears in the node and branch name fields.

  If you are working in the standalone mode and building expressions by using the Results Browser, you can get the node and branch name from the Results Browser:

  - Select the *Wave* radio button on the Selection toolbar.
  - Click a current signal in the Results Browser.

  The selected signal appears in the node and branch name fields.

- *Harmonic List*—Specify the harmonics for which you want to plot the results. It is an optional field. For analyses, such as `hb`, `pss`, `pac`, and `hbac`, you can add either single harmonic value or available list of harmonic values in this field.
  Valid values: Any integer or a list from the available list of harmonics. You can find the available harmonics by using the `harmonicList` function.
  `harmonicList( [?resultsDir t_resultsDir] [?result S_resultName]`
  Default value: `nil`

Example 1

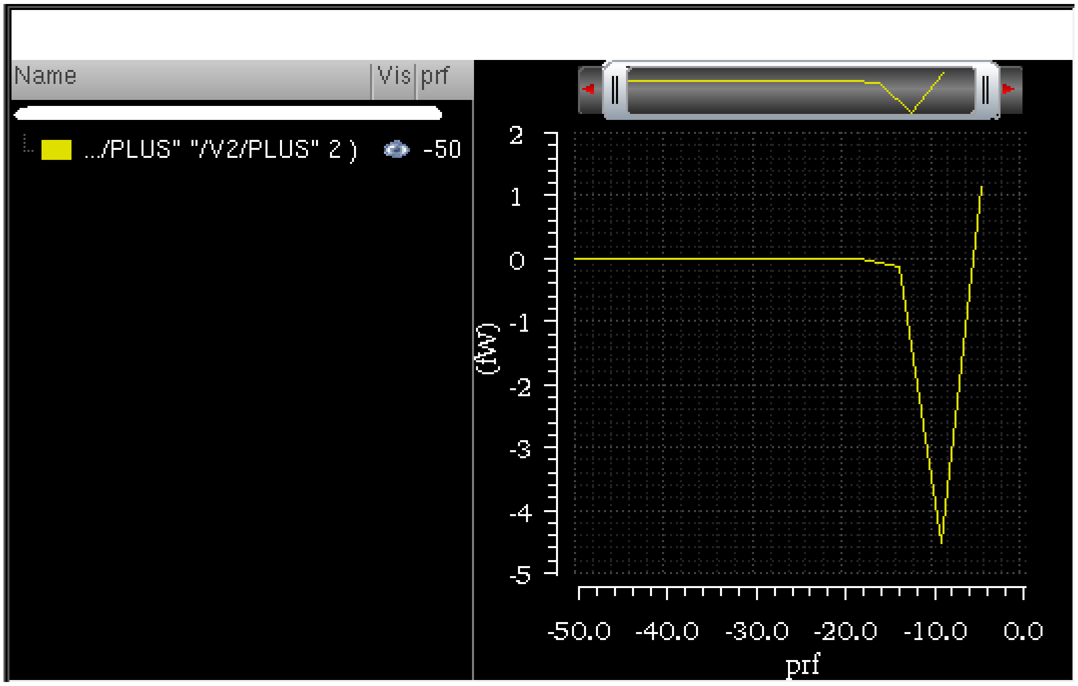
The following example shows the output plot generated if you build the following expression in the Buffer:

```
pvi("hb" "/RFin" "/RFout" "/V1/PLUS" "/V2/PLUS" 2)
```

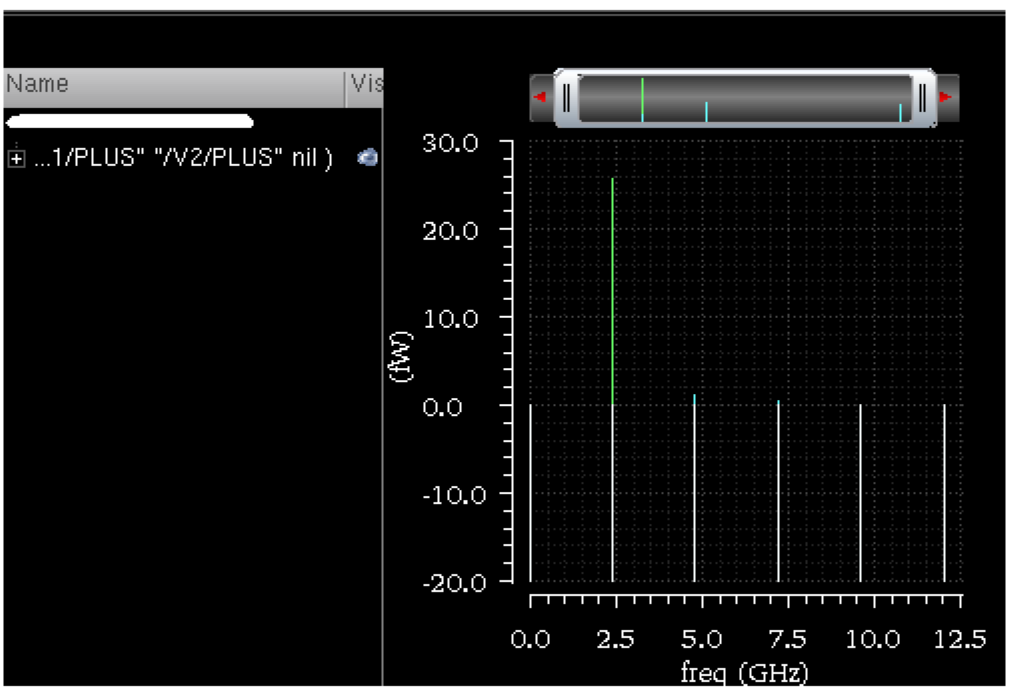
Here,

- *Analysis Type* is `hb`
- *Positive node* is `/RFin`
- *Negative node* is `/RFout`
- *Branch name* 1 `/V1/PLUS`
- *Branch name 2* `/V2/PLUS`

- *Harmonic List* is `2`



If you build the above expression with *Harmonic List*=`nil`, the spectral power on all the harmonic points are returned (as shown in the figure below).



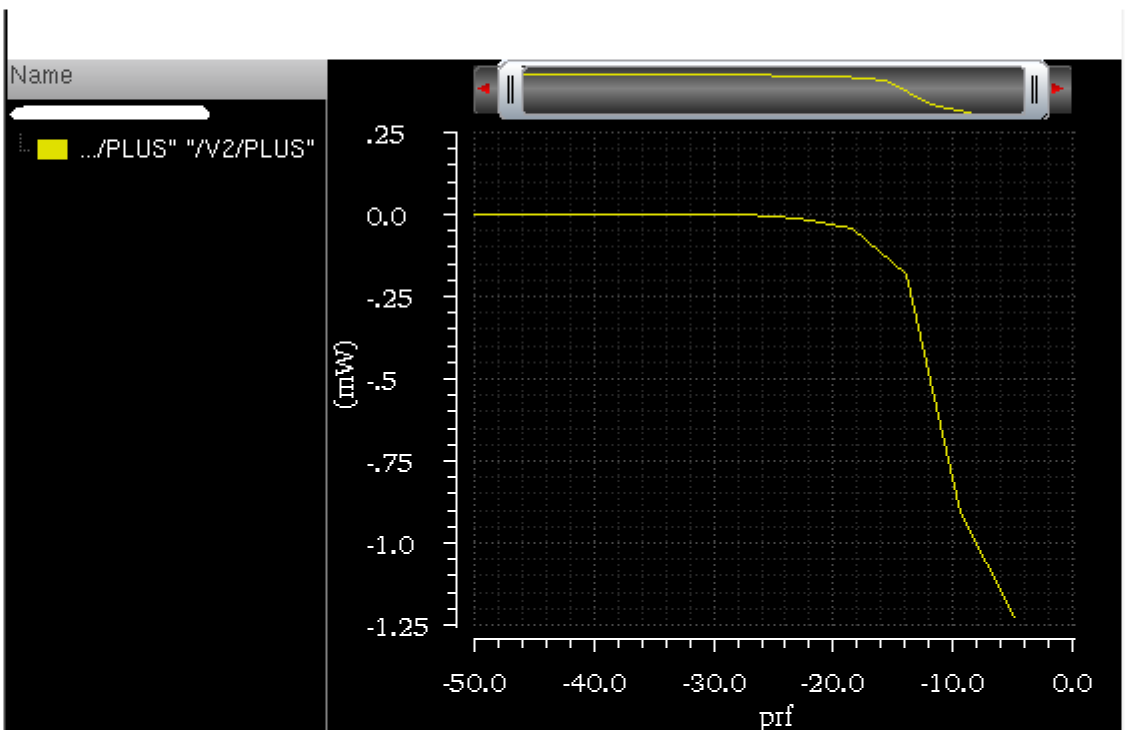
Example 2

The following example shows the output plot generated if you build the following expression in the Buffer:

```
pvi("hb" 2 3 "/V1/PLUS" "/V2/PLUS" 2 )
```

Here,

- *Analysis Type* is `hb`
- *Positive node* is `2`
- *Negative node* is `3`
- *Branch name* 1 `/V1/PLUS`
- *Branch name 2* `/V2/PLUS`
- *Harmonic List* is `2`



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the spectral power from voltage and current at a specified harmonic list or at all harmonics:

```
spectralPower(i("/load/PLUS" ?result "hb_fd") v("/RFout" ?result
"hb_fd"))
```

However, the similar calculation is now performed with the *pvi* function and the following expression is created in the Buffer:

```
pvi("hb" "/RFout" 0 "/load/PLUS" 0 nil )
```

pvr

Returns the spectral power for a specified harmonic list or for all harmonics with resistor and voltage on the positive and negative nodes.

This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `hb`, `pss`, `qpss`, `pac`, `hbac` and `qpac`.
  Default field: `hb`
- *Positive node*—Specify the positive node or net. This field can also contain an explicit voltage value.
- *Negative node*—Specify the negative node or net. This field can contain an explicit voltage value.

  When you open Calculator from ADE, you can select the node name directly from the Schematic window. The selected node name appears in the node and branch name fields.

  If you are working in the standalone mode and building expressions by using the Results Browser, you can get the node name from the Results Browser:

  - Select the *Wave* radio button on the Selection toolbar.
  - Click a current signal in the Results Browser.

  The selected signal appears in the node name fields.
- *Resistance*—Specify the resistance value. This field can contain any integer or floating point value. By default, it is set to blank.

- *Harmonic List*—Specify the harmonics for which you want to plot the results. It is an optional field. For analyses, such as `hb`, `pss`, `pac`, and `hbac`, you can add either single harmonic value or available list of harmonic values in this field.
  Valid values: Any integer or a list from the available list of harmonics. You can find the available harmonics by using the `harmonicList` function.
  `harmonicList( [?resultsDir t_resultsDir] [?result S_resultName]`
  Default value: `nil`

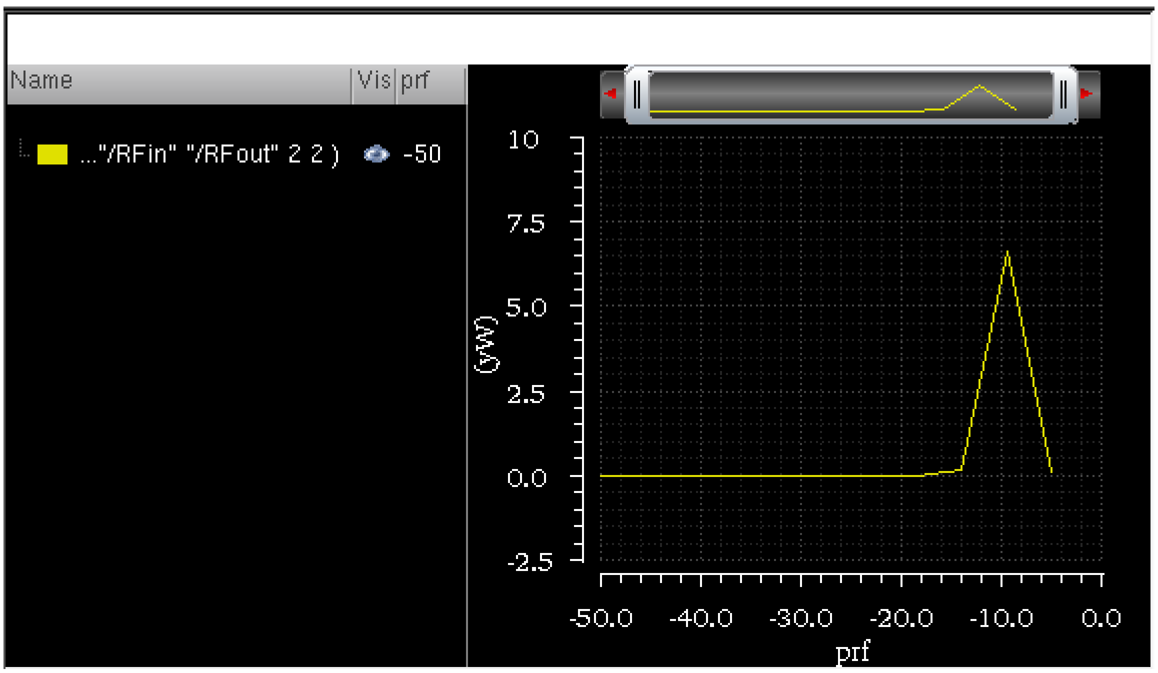
Example

The following example shows the output plot generated if you build the following expression in the Buffer:

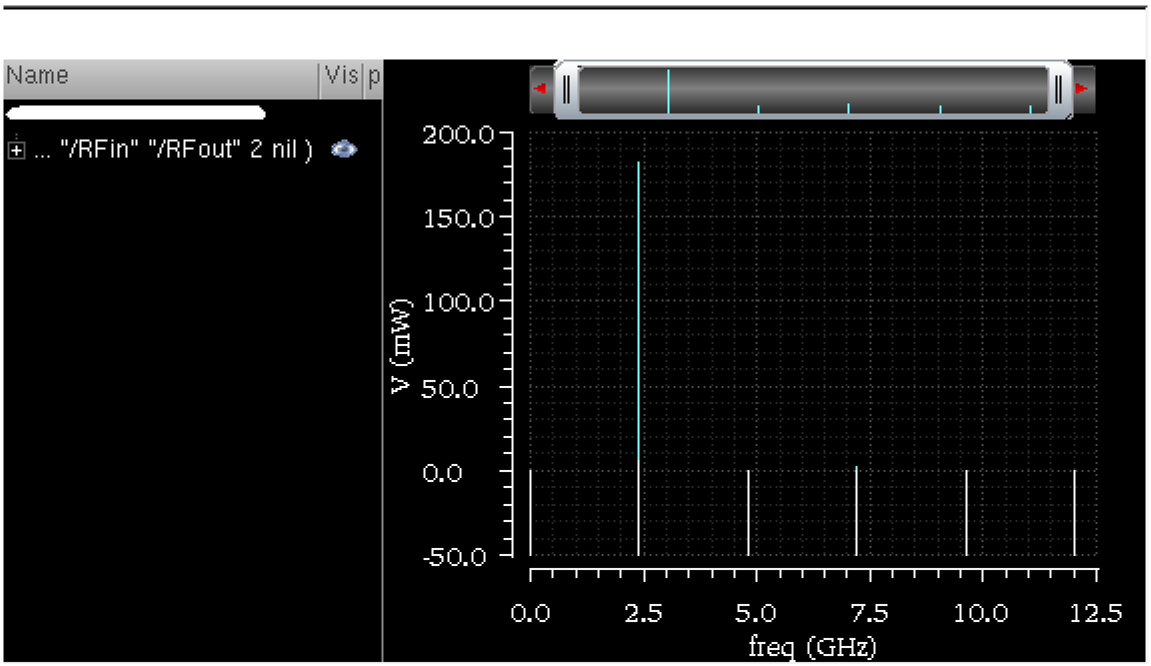
```
pvr("hb" "/RFin" "/RFout" 2 2 )
```

Here,

- *Analysis Type* is `hb`
- *Positive node* is `/RFin`
- *Negative node* is `/RFout`
- *Resistance* is `2`
- *Harmonic List* is `2`



If you build the above expression with *Harmonic List*=`nil`, the spectral power on all the harmonic points are returned (as shown in the figure below).



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the spectral power at a specified harmonic or at all harmonics with resistor and voltage on the positive and negative nodes.

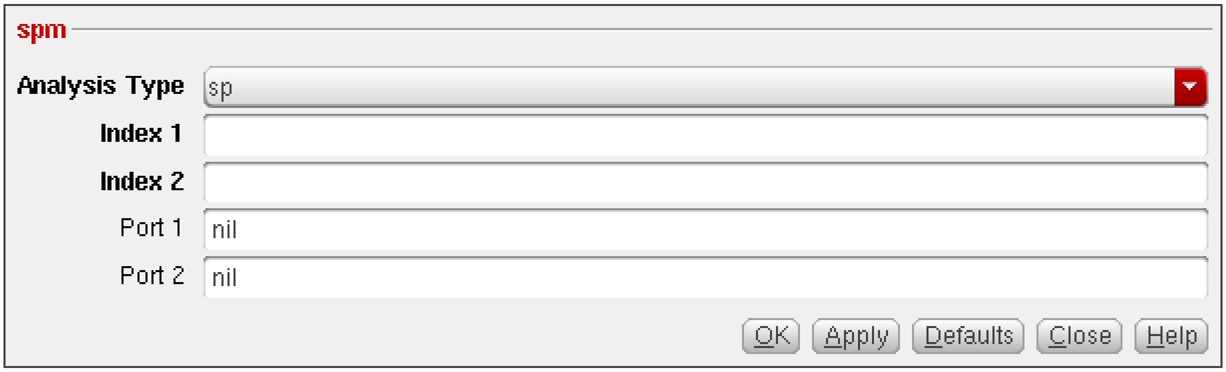
```
spectralPower(((v("/RFin" ?result "hb_fd") - v("/RFout" ?result
"hb_fd")) / 50) (v("/RFin" ?result "hb_fd") - v("/RFout" ?result
"hb_fd")))
```

However, the similar calculation is now performed with the *pvr* function and the following expression is created in the Buffer:

```
pvr("hb" "/RFin" "/RFout" 50 nil )
```

spm

Returns the waveform for s-parameters.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list.
  Valid values: `sp`, `psp`, `qpsp`, and `hbsp`
  Default value: `sp`
- *Index 1*—Specify the port index for sp simulation. By default, this field is set to blank.
  Valid values: Available port index, such as `1`, `2`.
- *Index 2*—Specify the port index for two-port sp simulation. By default, this field is set to blank.
  Valid values: Available port index, such as `1`, `2`.
- *Port 1*—Specify the port instance. The port instance can be specified only for the differential s-parameter analysis and not applicable for `psp`, `qpsp` and `hbsp` analyses.
  Valid values: Predefined values "`c`" and "`d`" for Spectre simulator.
- *Port 2*—Specify the port instance. The port instance can be specified only for the differential s-parameter analysis and not applicable for `psp`, `qpsp` and `hbsp` analyses.
  Valid values: Predefined values "`c`" and "`d`" for Spectre simulator.

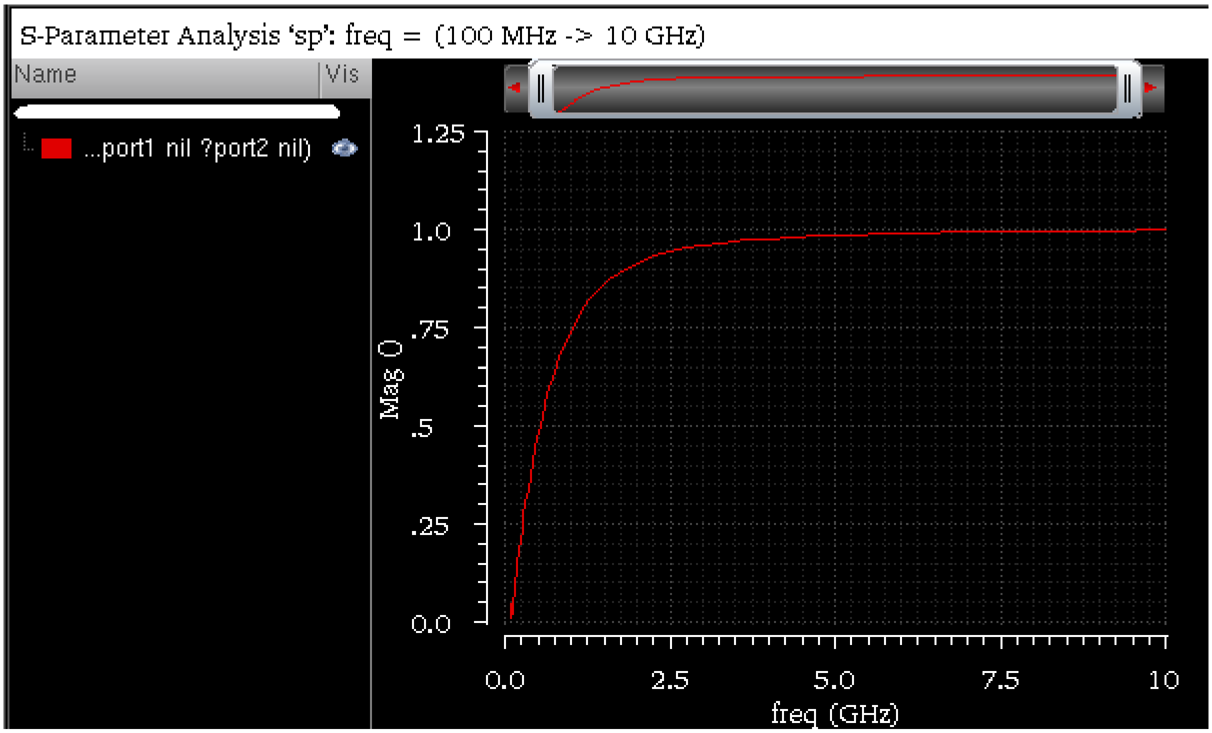
Example 1:

This example shows the output plot generated when you build the following expression in the Buffer:

```
spm("sp" 1 1 ?port1 nil ?port2 nil)
```

Here,

- *Analysis Name* is `sp`
- *Index 1* is `1`
- *Index 2* is `1`
- *Port 1* is `nil`
- *Port 2* is `nil`



Example 2:

This example shows the output plot generated when you build the following expression in the Buffer for differential s-parameter analysis:

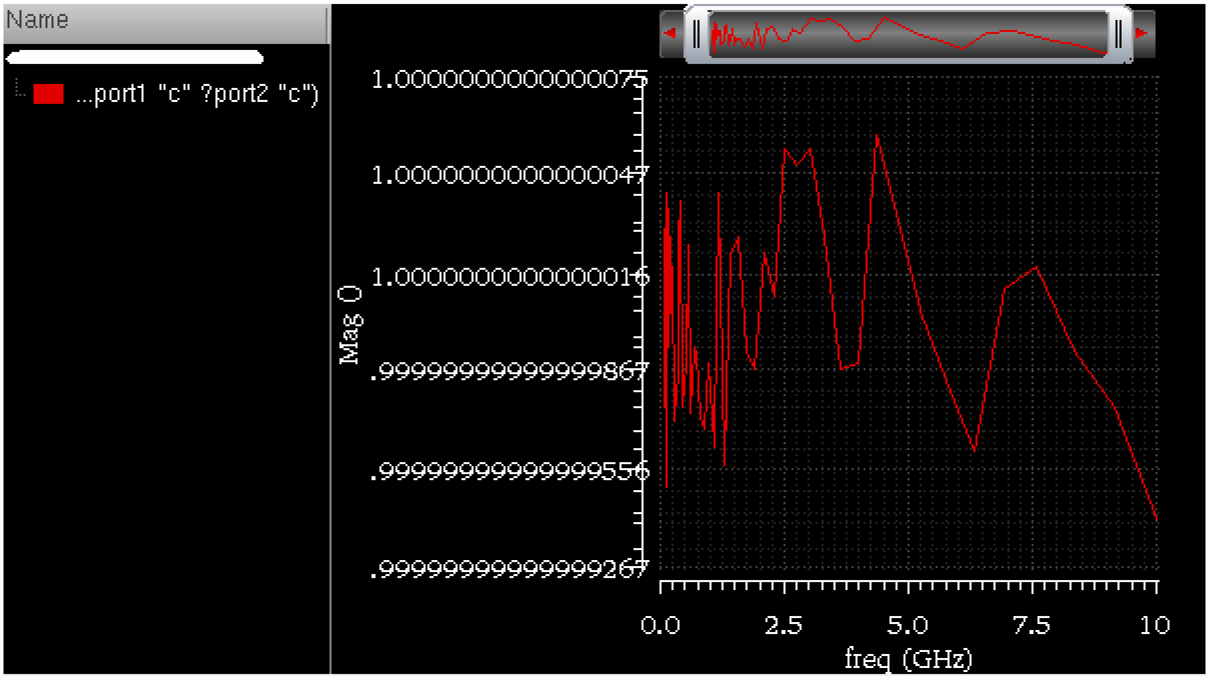
```
spm("sp" 1 1 ?port1 "c" ?port2 "c")
```

Here,

- *Analysis Name* is `sp`
- *Index 1* is `1`
- *Index 2* is `1`

- *Port 1* is “c”
- *Port 2* is “c”

S-Parameter Analysis ‘sp’: freq = (100 MHz -> 10 GHz)



Additional Information

The following expression was created in the Buffer in the previous releases when you plotted the waveform for s-parameters:

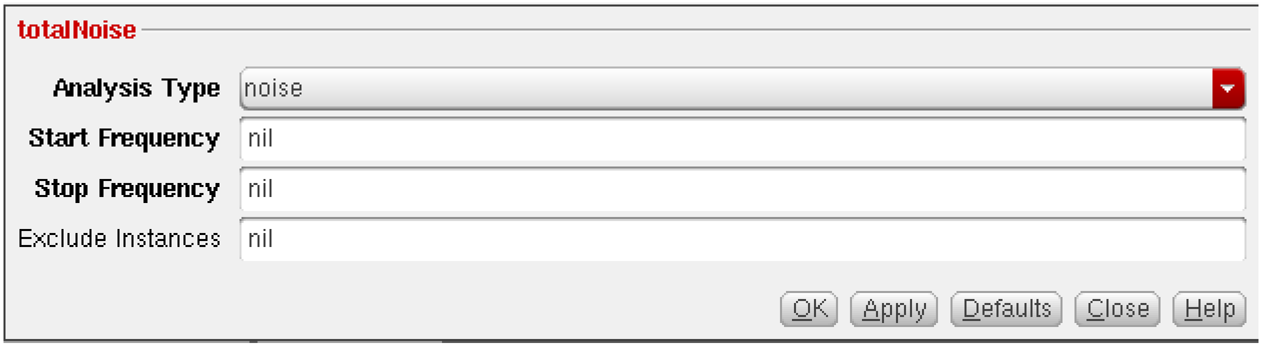
```
sp( 1 1 ?result \"sp\" ?port1 nil ?port2 nil)
```

However, the similar plot is now generated with the *spm* function and the following expression is created in the Buffer:

```
spm( \"sp\" 1 1 ?port1 nil ?port2 nil)
```

totalNoise

Returns the total noise in a specified frequency limit.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `noise`, `pnoise`, `qpnoise`, and `hbnoise`.
  Default value: `noise`
- *Start Frequency*—Specify the start frequency.
- *End Frequency*—Specify the end frequency.
- *Exclude Instances*—Specify a list of instances or instance names. The noise contributed by the instances specified in this field is ignored while calculating the total noise. This is an optional field.

Example

This example shows the output generated when you build the following expression in the Buffer:

```
totalNoise("hbnoise" 1k 100k nil )
```

Here,

- *Analysis Name* is `hbnoise`
- *Start Frequency* is `1KHz`
- *End Frequency* `100KHz`
- *Exclude Instances* is `nil`

The output generated in the Buffer is `479.7E-9`, which is a scalar value.

Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the total noise in a specified frequency limit:

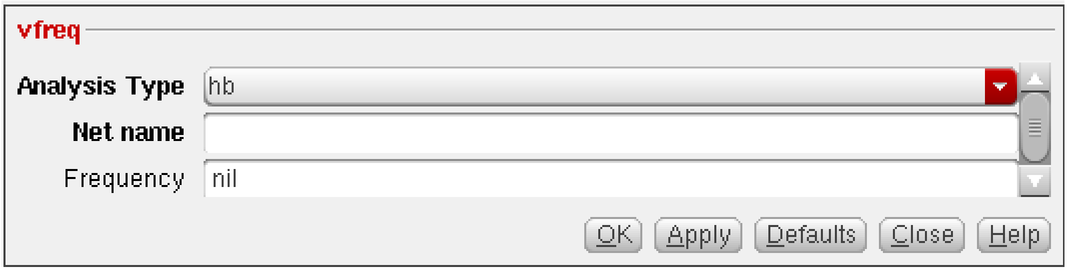
```
integ((getData("out" ?result "hbnoise")**2-pv("out" "total" ?result
"hbnoise")) 1k 100k)
```

However, the similar calculation is now performed with the *totalNoise* function and the following expression is created in the Buffer:

```
totalNoise('hbnoise,1k,100k "out")
```

vfreq

Returns the voltage of net at a specified frequency or at all frequencies in the frequency domain.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `hb`, `pss`, `qpss`, `pac`, `hbac`, `qpac`, and `ac`.
  Default value: `hb`
- *Net name*— Specify the net name from the schematic or signal name from the Results Browser.

  When you open Calculator from ADE, you can select the net name directly from the Schematic window. The selected net name appears in this field.

  If you are working in the standalone mode and building expressions by using the Results Browser, you can get the net name from the Results Browser:

  - Select the *Wave* radio button on the Selection toolbar.
  - Click a current signal in the Results Browser.

    The selected signal appears in the *Net name* field.
- *Frequency*—Specify the frequency for which you want to plot the results. It is an optional field.
  Valid values: Any integer or floating point number.
  Default Value: `nil`

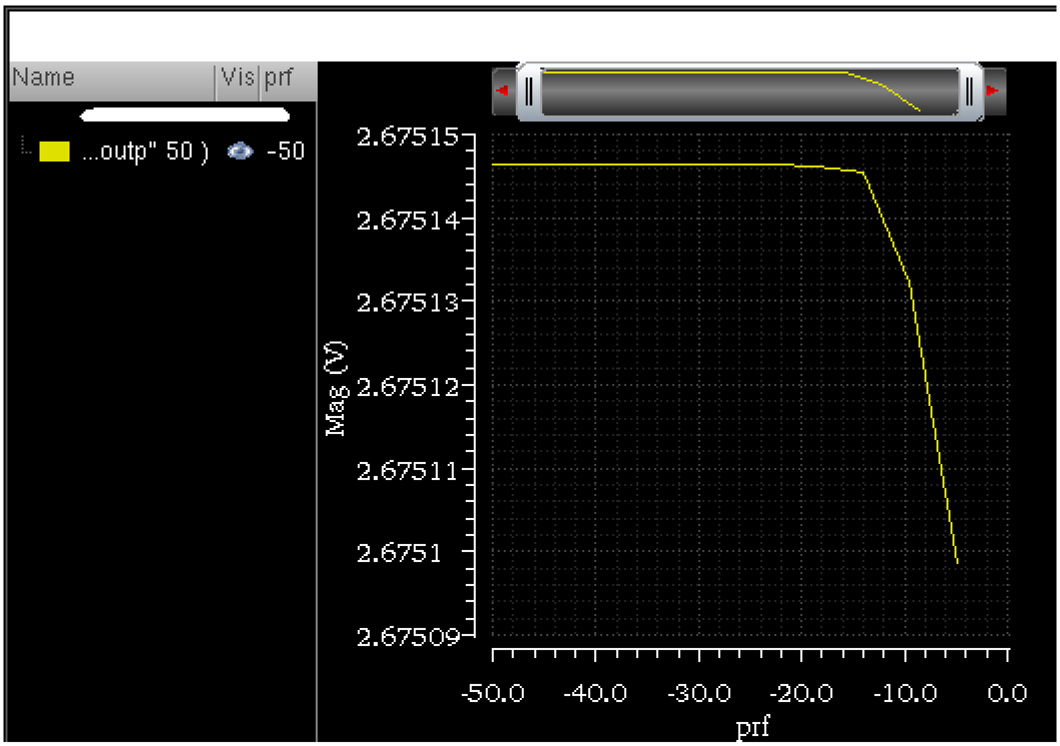
Example

This example shows the output plot generated if you build the following expression in the Buffer:

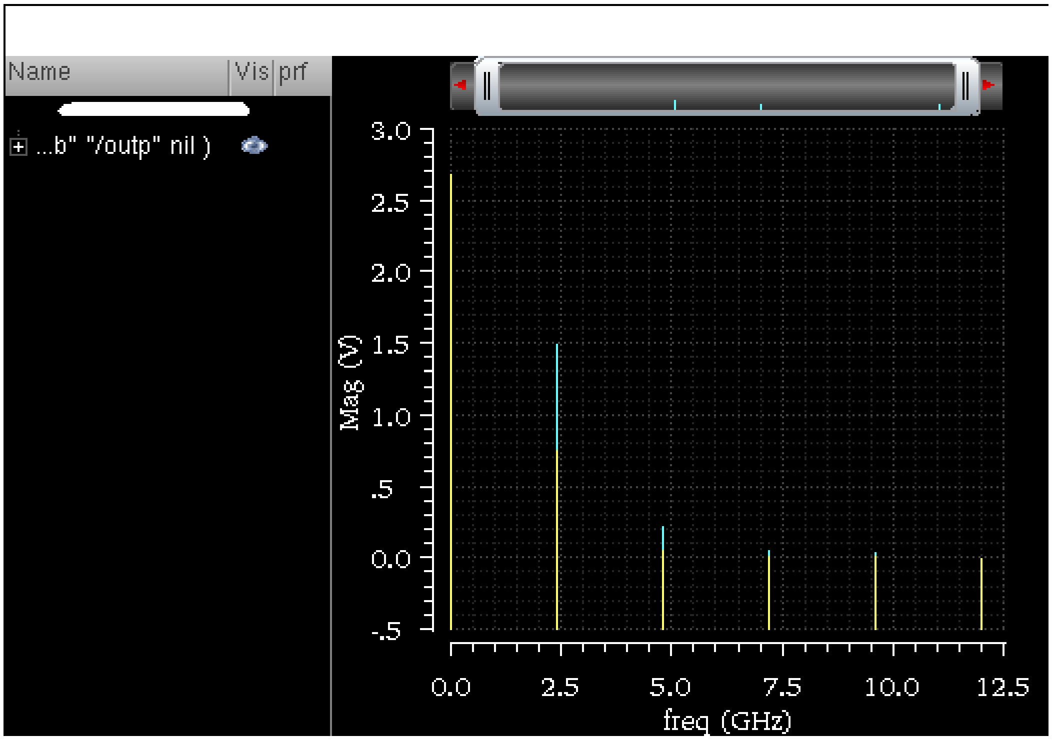
```
vfreq("hb" "/outp" 50 )
```

Here,

- *Analysis Name* is `hb`
- *Net name* is `/outp`
- *Frequency* is `50`



If you build the above expression with *Frequency*=`nil`, the voltage on all the frequency points are returned (as shown in the figure below).



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated voltage of net at a specified frequency or at all frequencies in the frequency domain:

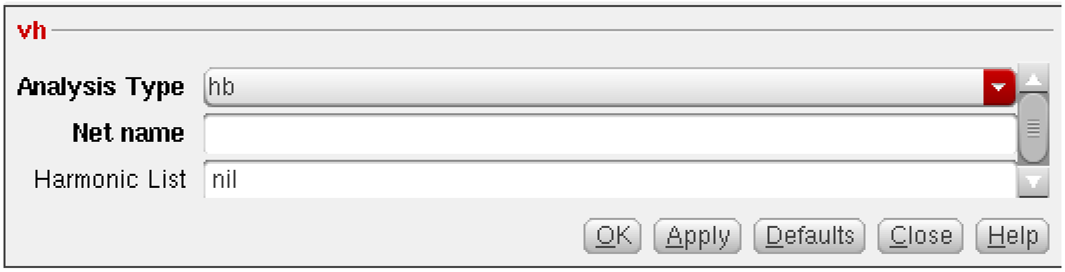
```
v("/RFout" ?result "hbac")
```

However, the similar calculation is now performed with the *vfreq* function and the following expression is created in the Buffer:

```
vfreq('hbac "/RFout")
```

vh

Returns the voltage on net at a specified harmonic or at all harmonics in the frequency domain.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `hb`, `pss`, `qpss`, `pac`, `hbac`, and `qpac`.
  Default value: `hb`
- *Net name*— Specify the net name from the schematic or signal name from the Results Browser.

  When you open Calculator from ADE, you can select the net name directly from the Schematic window. The selected net name appears in this field.

  If you are working in the standalone mode and building expressions by using the Results Browser, you can get the net name from the Results Browser:

  - Select the *Wave* radio button on the Selection toolbar.
  - Click a current signal in the Results Browser.

    The selected signal appears in the *Net name* field.
- *Harmonic List*—Specify the harmonics for which you want to plot the results. It is an optional field. For analyses, such as `hb`, `pss`, `pac`, and `hbac`, you can add either single harmonic value or available list of harmonic values in this field.
  Valid values: Any integer or a list from the available list of harmonics. You can find the available harmonics by using the `harmonicList` function.
  `harmonicList( [?resultsDir t_resultsDir] [?result S_resultName]`
  Default value: `nil`

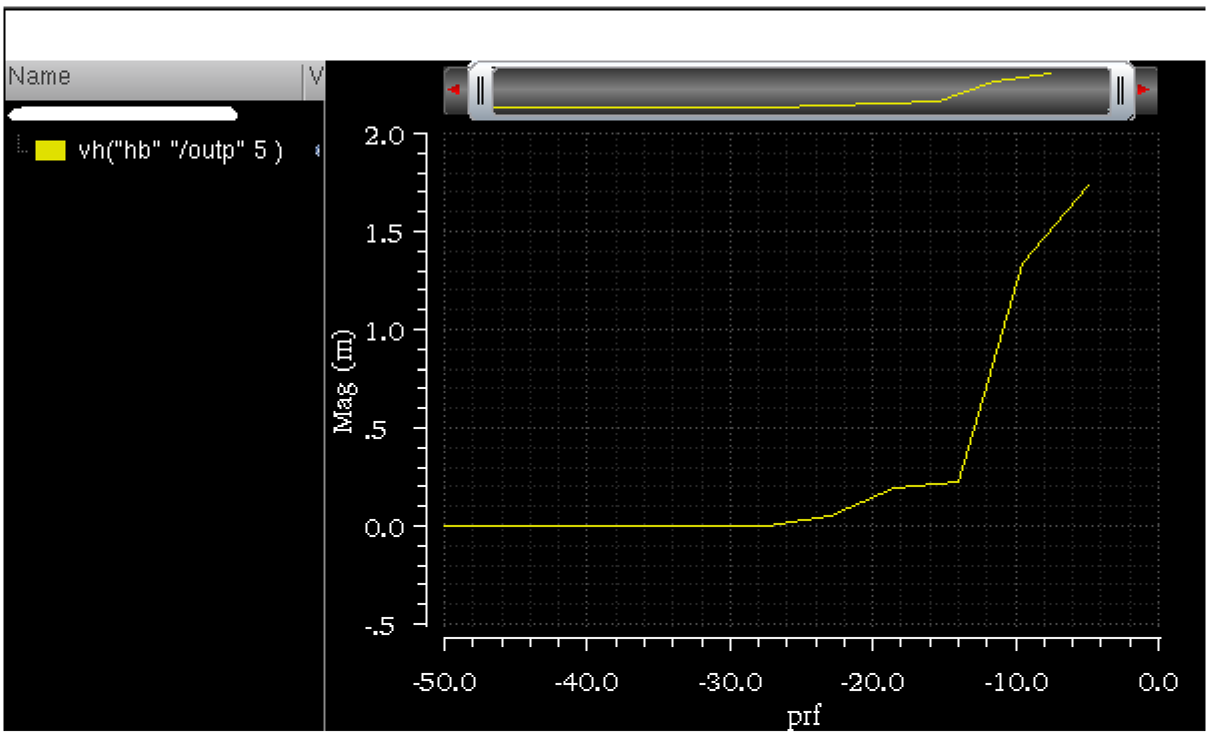
Example

This example shows the output plot generated if you build the following expression in the Buffer:

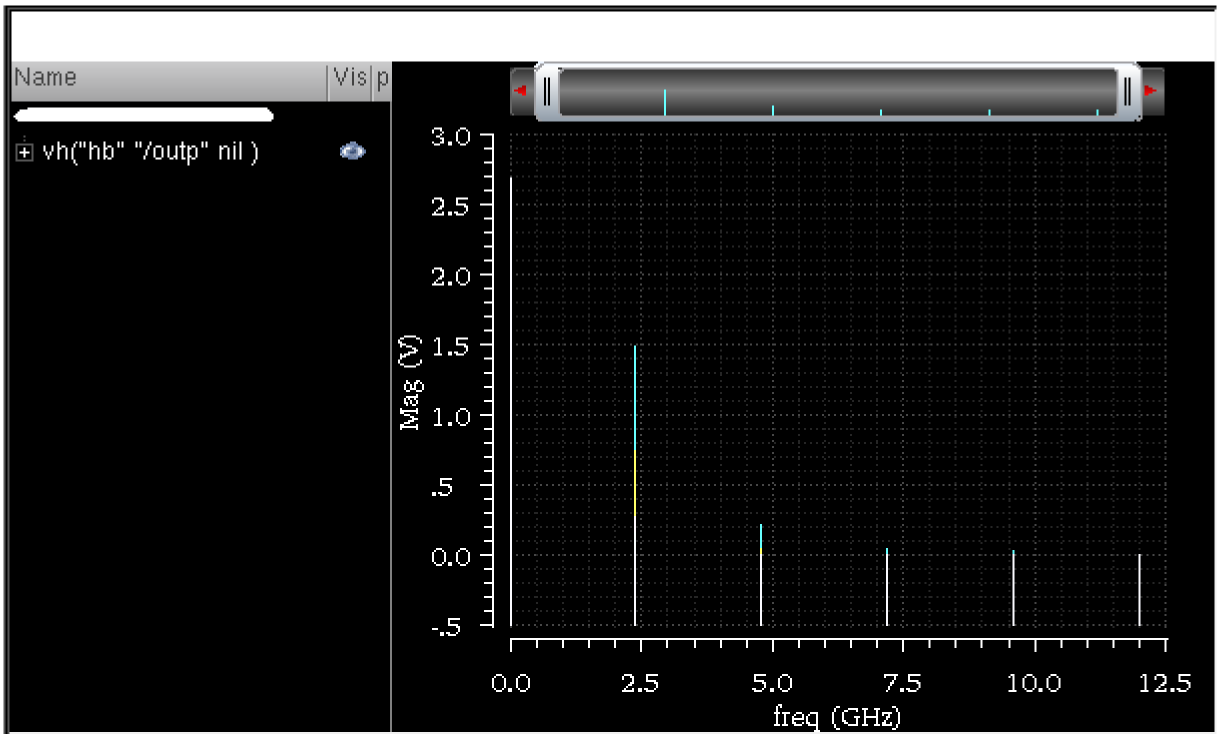
```
vh("hb" "/outp" 5 )
```

Here,

- *Analysis Name* is `hb`
- *Net name* is `/outp`
- *Harmonic List* is `5`



If you build the above expression with *Harmonic List*=`nil`, the voltage on all the harmonic points are returned (as shown in the figure below).



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the voltage on net at a specified harmonic or at all harmonics in the frequency domain:

```
v("/RFout" ?result "hb_fd")
```

However, the similar calculation is now performed with the *vh* function and the following expression is created in the Buffer:

```
vh('hb "/RFout")
```

vtime

Returns the voltage of net at a specified time point or at all time points in the time domain.

This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list. The available analyses are `hb`, `pss`, and `tran`.
  Default value: `hb`
- *Net name*— Specify the net name from the schematic or signal name from the Results Browser.

  When you open Calculator from ADE, you can select the terminal name directly from the Schematic window. The selected terminal name appears in this field.

  If you are working in the standalone mode and building expressions by using the Results Browser, you can get the terminal name from the Results Browser:

  - Select the *Wave* radio button on the Selection toolbar.
  - Click a current signal in the Results Browser.

    The selected signal appears in the *Terminal name* field.
- *Time value*—Specify the time points for which you want to plot the results. If you specify a time point in this field, the result of the specified time is returned. Otherwise, It is an optional field.
  Valid values: Any integer or floating point number.
  Default value: `nil`.

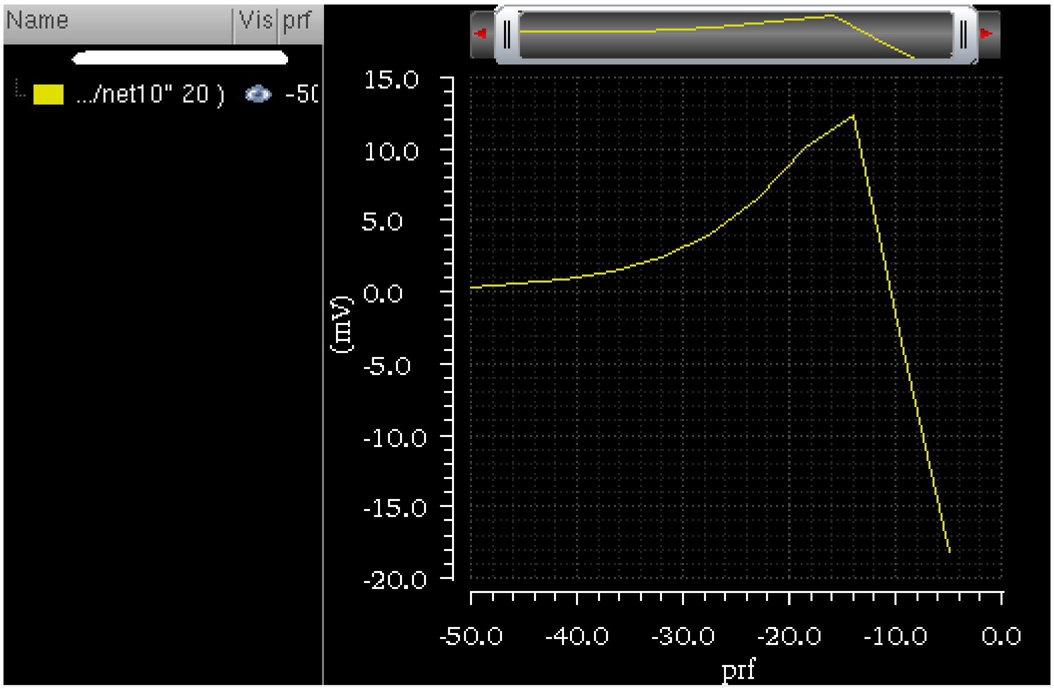
Example

This example shows the output plot generated if you build the following expression in the Buffer:

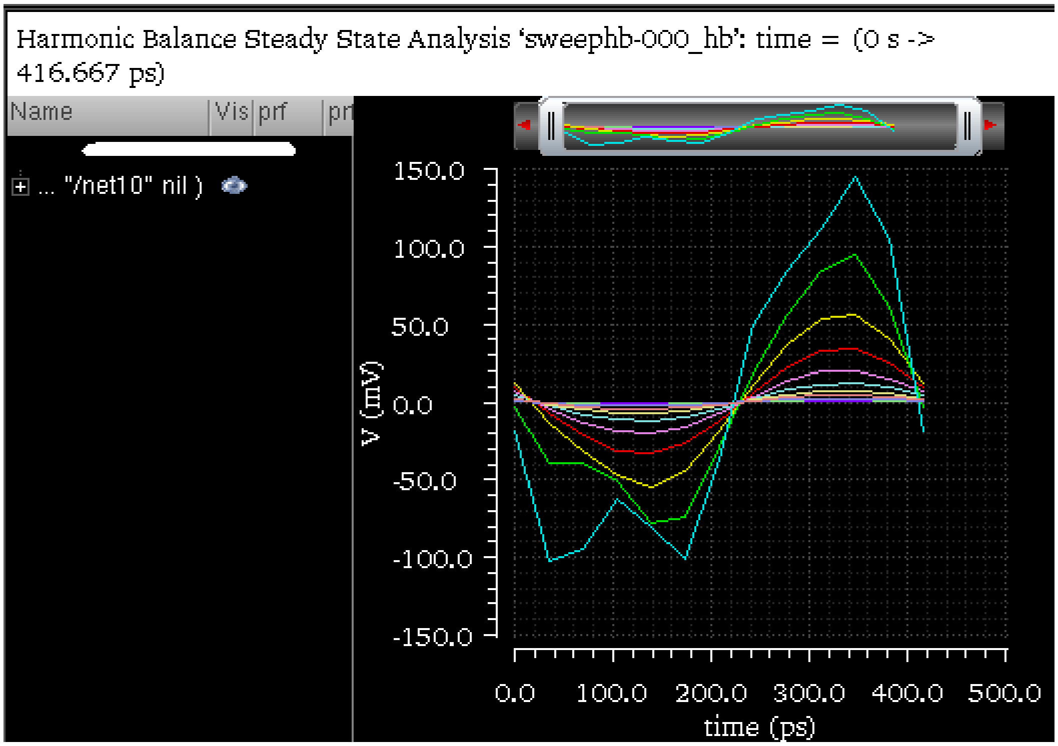
```
vtime("hb" "/net10" 20)
```

Here,

- *Analysis Type* is `hb`
- *Net name* is `/net10`
- *Time value* is `20`



If you build the above expression with *Time value*=`nil`, the voltage on all the time points are returned (as shown in the figure below).



Additional Information

The following expression was created in the Buffer in the previous releases when you calculated the voltage of net at a specified time point or at all time points in the time domain:

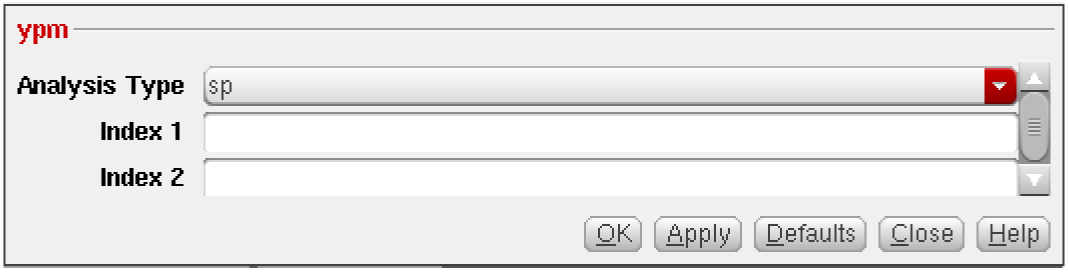
```
v("/RFin" ?result "hb_td")
```

However, the similar calculation is now performed with the *vtime* function and the following expression is created in the Buffer:

```
vtime("hb" "/RFin" nil )
```

ypm

Returns the waveform for the y-parameter.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list.
  Valid values: `sp`, `psp`, `qpsp`, and `hbsp`
  Default value: `sp`
- *Index 1*—Specify the port index for sp simulation. By default, this field is set to blank.
  Valid values: Available port index, such as `1`, `2`
- *Index 2*—Specify the port index for sp simulation. By default, this field is set to blank.
  Valid values: Available port index, such as `1`, `2`

Example

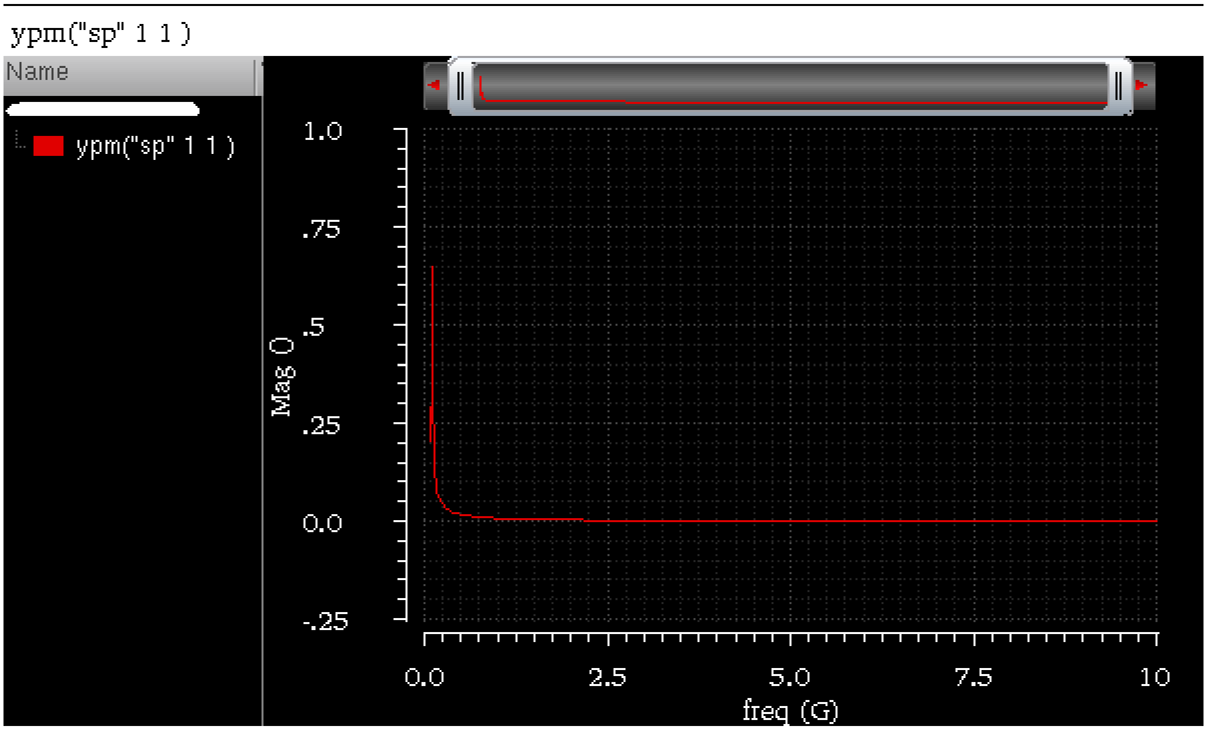
This example shows the output plot generated when you build the following expression in the Buffer:

```
ypm("sp" 1 1)
```

Here,

- *Analysis Name* is `sp`
- *Index 1* is `1`

- *Index 2* is `1`



Additional Information

The following expression was created in the Buffer in the previous releases when you plotted the waveform for y-parameters:

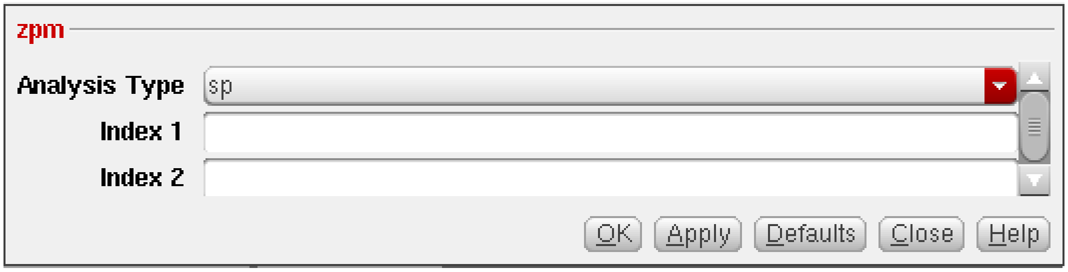
```
yp(1 1 ?result \"sp\")
```

However, the similar plot is now generated with the *ypm* function and the following expression is created in the Buffer:

```
ypm('sp 1 1)
```

zpm

Returns the waveform for the z-parameter.



This function includes the following fields:

- *Analysis Type*—Select the analysis type from the drop-down list.
  Valid values: `sp`, `psp`, `qpsp`, and `hbsp`
  Default value: `sp`
- *Index 1*—Specify the port index for sp simulation. By default, this field is set to blank.
  Valid values: Available port index, such as `1`, `2`
- *Index 2*—Specify the port index for sp simulation. By default, this field is set to blank.
  Valid values: Available port index, such as `1`, `2`

Example

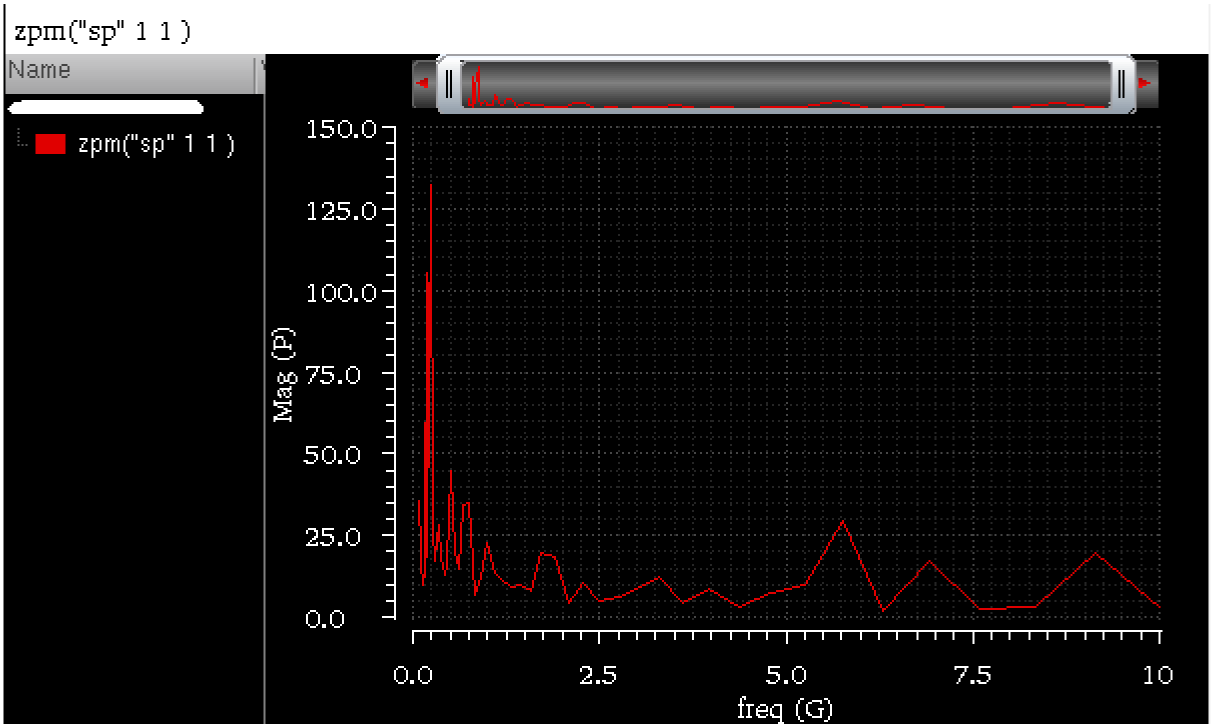
This example shows the output plot generated when you build the following expression in the Buffer:

```
zpm("sp" 1 1)
```

Here,

- *Analysis Name* is `sp`
- *Index 1* is `1`

- *Index 2* is `1`



Additional Information

The following expression was created in the Buffer in the previous releases when you plotted the waveform for z-parameters:

```
zp(1 1 ?result \"sp\")
```

However, the similar plot is now generated with the *zpm* function and the following expression is created in the Buffer:

```
zpm('sp 1 1)
```

Special Functions

This section describes the following special functions available in the Virtuoso Visualization and Analysis XL Calculator:

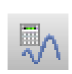
- a2d
- abs_jitter
- average
- bandwidth
- clip
- compare
- compression
- compressionVRI
- convolve
- cross
- d2a
- dBm
- delay
- deriv
- dft
- dftbb
- dnl
- dutycycle
- evmQAM
- evmQpsk
- eyeDiagram
- fallTime
- flip
- fourEval

- freq
- freq_jitter
- frequency
- gainBwProd
- gainMargin
- getAsciiWave
- groupDelay
- harmonic
- harmonicFreq
- histogram2D
- iinteg
- inl
- integ
- intersect
- ipn
- ipnVRI
- lshift
- normalQQ
- overshoot
- pavg
- peak
- peakToPeak
- period_jitter
- phaseMargin
- phaseNoise
- pow
- prms

- psd
- psdbb
- pstddev
- pzbode
- pzfilter
- risetime
- rms
- rmsNoise
- root
- rshift
- sample
- settlingTime
- slewrate
- spectralPower
- spectrumMeas
- stddev
- tangent
- thd
- unityGainFreq
- value
- xmax
- xmin
- xval
- ymax
- ymin

Basic Steps For Running Calculator Functions

You need to perform the following steps in Virtuoso Visualization and Analysis XL to run the Calculator functions:

1. Open the results database in the Results Browser window. For information about how to open and access result databases in Results Browser, see Working with the Results Directory on page 38.
2. Right-click the signal you want to use as an input to the function and choose *Calculator*. Alternatively, you can plot the signal in the graph window and then send the plotted signal from graph to Calculator.
3. The expression for the selected signal is displayed in the Buffer. This signal expression can then be used as an input signal while running the functions. For some functions, you need to input more than one signals. In this case, you can use Stack to store the signals. For information about how to use Stack, see Stack on page 322.
4. Now, open the Function Panel and select the signal you want to run. For more information about how to use the Function Panel, see Function Panel on page 324.
5. If the function includes argument fields, a form appears in Function Panel where you can provide the argument values. By default, the default values are populated in the argument fields. If signal does not require arguments, it is directly applied to the signal expression in the Buffer.
6. Click *OK*. The expression for the function is displayed in the Buffer.
7. Set the output plotting mode from the *Plotting Mode* drop-down on the *Selection* toolbar.
8. Then, click the Evaluate Buffer icon .

   If the result is a scalar value, it will be displayed in the Buffer. If the result is a waveform, it is plotted in the graph window in the specified plotted mode.

a2d

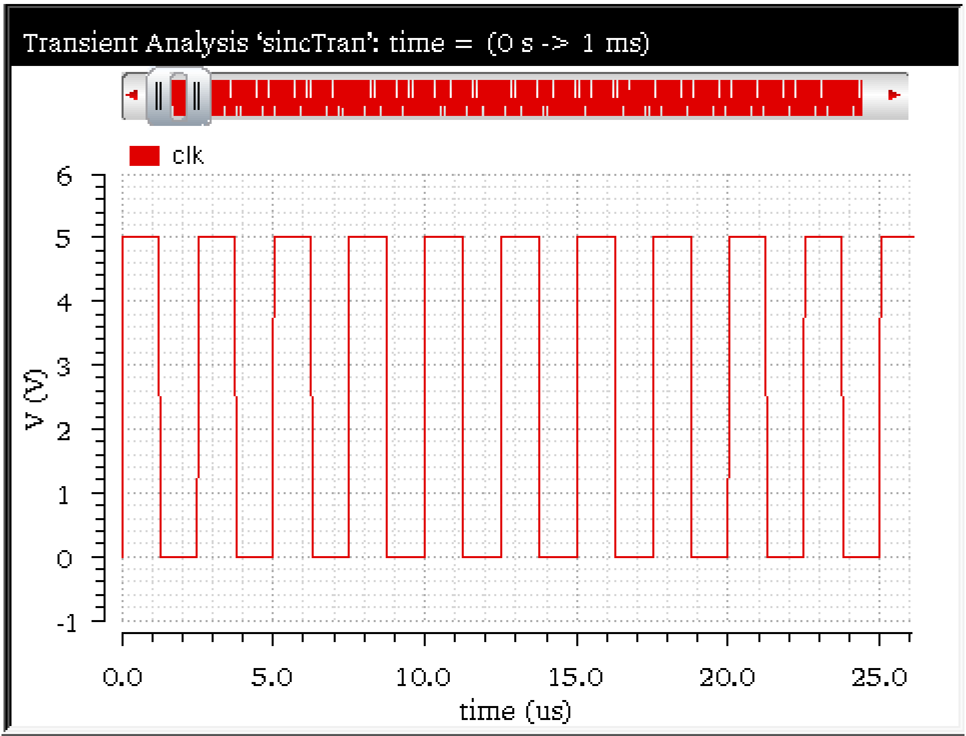
Returns the digital form of an analog input waveform, which may be a scalar, list or family of waveforms or a string representation of expressions. The corresponding SKILL command name for this function is `awvAnalog2Digital`.

This function includes the following arguments:

- *Signal*—Name of the analog signal.
- *Logic Threshold*—Select the logic voltage threshold as `High/Low` or `Centre`. The default value is `High/Low`.
- *Voltage High/Low*—Provide the high and low voltage values, if you select the logic threshold as `High/Low`. Any value higher than the Voltage High is the high state and any value lower than the Voltage Low is the low state. The default
- *Centre Voltage*—Provide the centre voltage value, if you select the logic threshold as `Centre`. Any value higher than the centre voltage is the high state and any value lower than the centre voltage is the low state.
- *Time to X*—The value that determines logic X.

Example

Consider the following input signal:



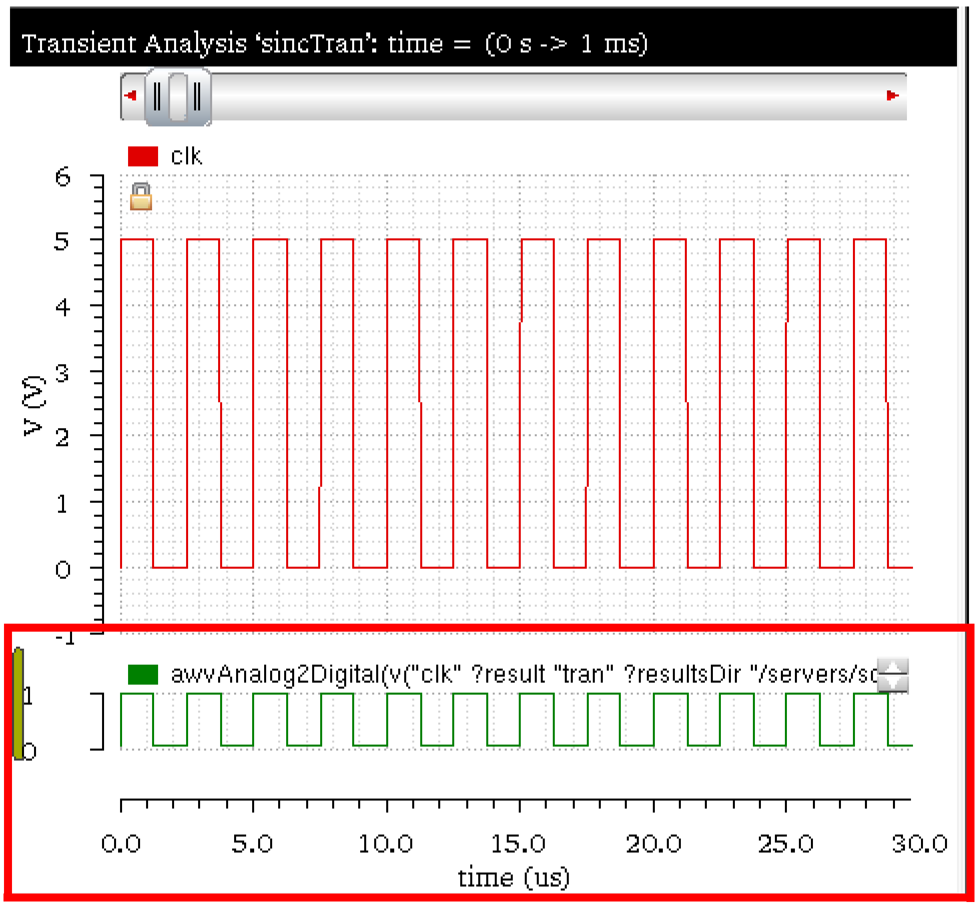
The `a2d` function is applied on this input signal with the following arguments:

- *Signal*—`v("clk" ?result "tran" ?resultsDir "./mixed/test/adc_8bit.raw")`
- *Logic Threshold*—High/Low
- *Voltage High/Low*—`1.78` and `0.2`
- *Centre Voltage*—`nil`
- *Time to X*—`1n`

The corresponding expression created in the Buffer is as follows:

```
awvAnalog2Digital(v("clk" ?result "tran" ?resultsDir "./mixed/test/
adc_8bit.raw") 1.78 0.2 nil 1n "hilo" )
```

When you evaluate this expression and plot the results in the append mode, the given analog signal is converted into a digital signal as shown in the figure below.



Related OCEAN Function

The equivalent OCEAN command for `a2d` is:

```
awvAnalog2Digital( o_waveform n_vhi n_vlo n_vc n_timex t_thresholdType )
     => o_digWave | n_digval/nil
```

For more information, see `a2d` in *OCEAN Reference*.

abs_jitter

Returns a waveform that contains the absolute jitter values in the input waveform for the given threshold. The output waveform can be expressed in degrees, radians, or unit intervals (UI). The absolute jitter can be plotted as a function of cycle number, crossing time, or reference clock time.

The function is available only in the SKILL mode and includes the following fields:

- *Waveform*—Name of the waveform, expression, or family of waveforms. The expression displayed in the Buffer can be added to this field by selecting *buffer*.
- *Cross Type*—Specifies whether the jitter value can be calculated on rising (`rising`) or falling (`falling`) curves. Points at which the curves of the waveform intersect with the threshold.
- *Threshold*—Value at which the input waveform intersects to calculate the absolute jitter.
- *X-Unit*—Unit defined for X-axis of the output waveform. Specify whether you want to output the absolute jitter against *time* or *cycle*. Cycle numbers refer to the nth occurrence where the waveform crosses the given threshold.
- *Y-Unit*—Unit defined for Y-axis of the output waveform. Specify whether you want to calculate the phase in degrees (*s*), radians (*rad)*, or unit intervals (*UI*).
- *Tnom*—Nominal time period of the input waveform. The waveform is expected to be a periodic waveform that contains noise. If you do not enter the *Tnom* value, the `abs_jitter` function autocalculates the approximate time period of the input waveform by using the following equation:

  ```
  (lastCrossing-firstCrossing)/(numCrossings-1)
  ```

  where,

- ❑ Crossing times are determined by the time at which the specified threshold is crossed.
- ❑ `numCrossings` determines the number of crossings.

Additional Information

The absolute jitter can be defined as follows:

For a given waveform $v(t), t_{start} \le t \le t_{end}$, with the following properties:

- Oscillating with expected nominal period, `T`,
- Between minimum and maximum values, $v_{min} \le v(t) \le v_{max}$,
- Rising and falling through a given threshold, v_{th}
- In time intervals, $t_k; 0 \le k \le N$

The absolute jitter for the waveform can be defined as:

$$J_a(k) = t_k - k \cdot T$$

The period jitter of the waveform can be defined as:

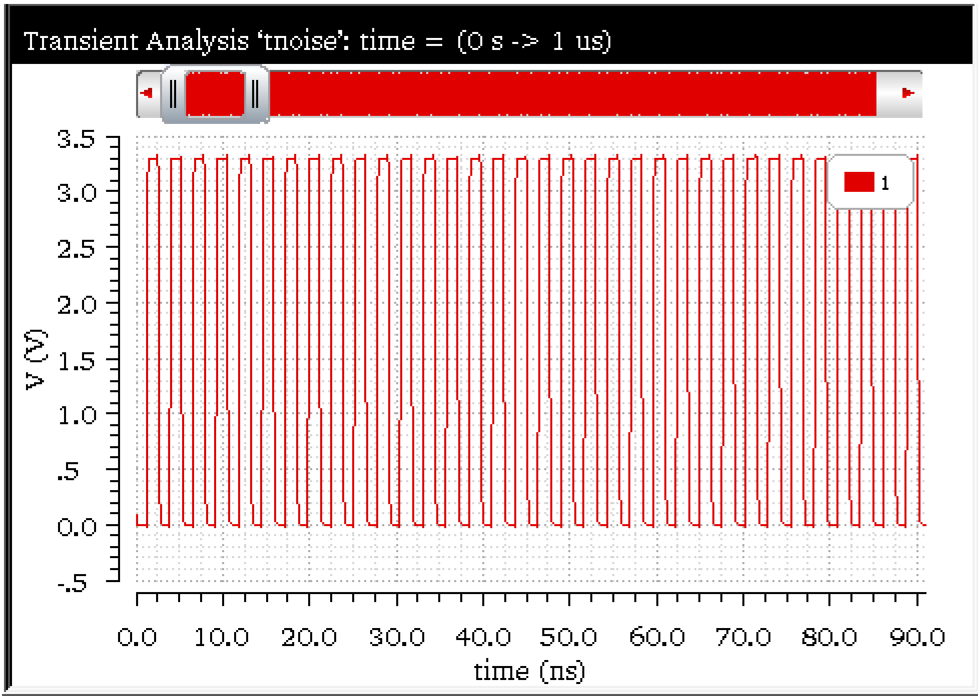
$$J_p(k) = t_k - t_{k-1} - T = J_a(k) - J_a(k-1) = \frac{d}{dk} J_a(k)$$

The jitter can be expressed in units of time (seconds) or in units of phase (radians or unit intervals). These values can be converted by using the following formula:

$$J_a[s] = \frac{J}{T}[UI] = \frac{J}{T} \cdot 2\pi[rad]$$

Example

Consider the following input waveform:



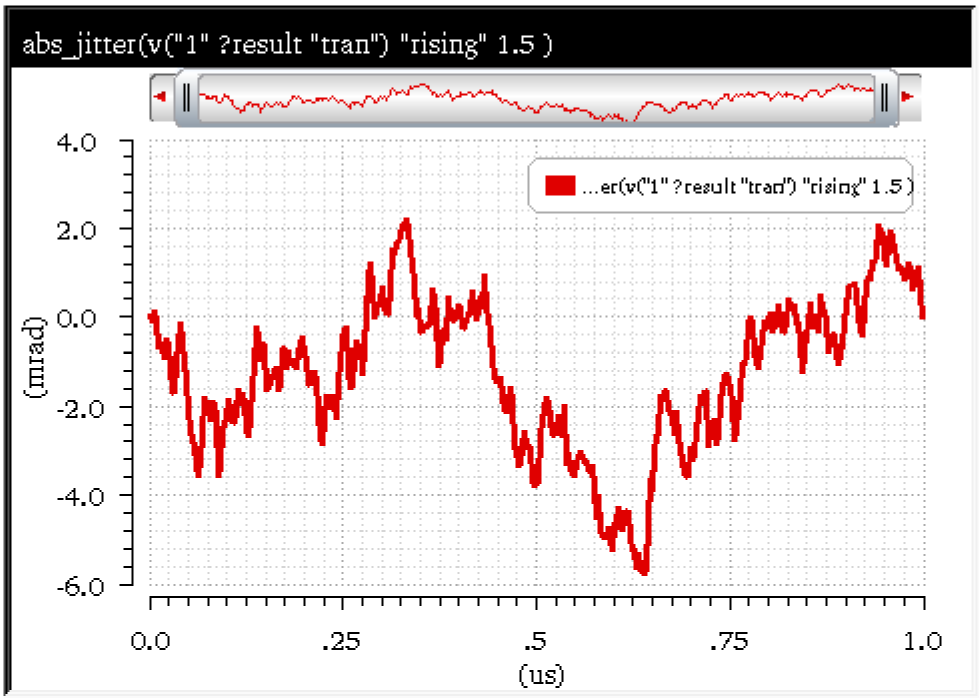
The `abs_jitter` function is applied to this waveform with the following arguments:

- *Waveform*—`v("1" ?result "tran")`
- *Cross Type*—`rising`
- *Threshold*—`1.5`
- *X-unit*—`s`
- *Y-unit*—`rad`
- *Tnom*—`nil`

The expression created in the Buffer is as follows:

```
abs_jitter(v("1" ?result "tran" ?resultsDir "nand2_ring_tnoise.raw")
"rising" 1.5)
```

When you evaluate this expression, the following output waveform showing the absolute jitter values is displayed in the new graph window:



Related OCEAN Function

The equivalent OCEAN command for `abs_jitter` is:

```
abs_jitter(o_waveform t_crossType n_threshold
     ?xUnit t_xUnit ?yUnit t_yUnit ?Tnom n_Tnom)
     => o_waveform/nil
```

For more information, see abs_jitter in *OCEAN Reference*.

average

Computes the average of a waveform over its entire range. Average is defined as the integral of the expression `f(x)` over the range of `x`, divided by the range of `x`.

If `y=f(X)`

$$Average(y) = \frac{\int_{from}^{to} f(x)dx}{to - from}$$

where, `from` is the initial value for x and `to` is the final value of x.

Additional Information

When you use the `average` function to create expressions that are to be measured across corners in ADE XL, the function includes an additional argument, `overall`. The expression created is as follows:

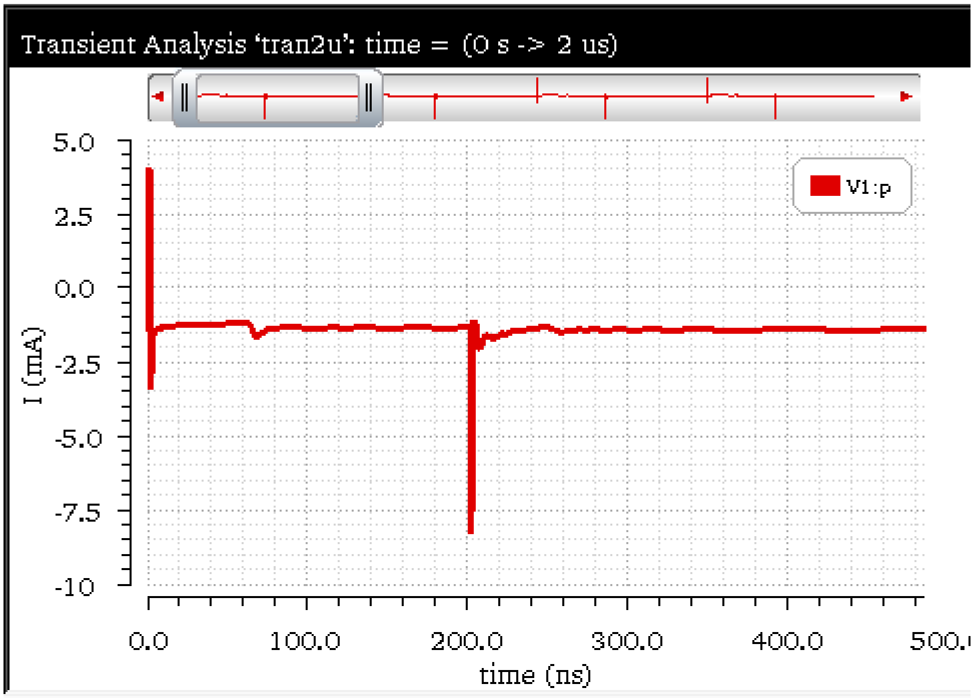
```
average(v("out" ?results "tran") ?overall t)
```

When the `overall` argument is set to `t`, it performs the calculation on the results of corner simulations for each design point that are treated as discrete values for evaluation. When set to `nil`, it creates a waveform from the data points and then calculates the average value of the waveform.

For more information, see Creating Expressions to be Measured Across Corners in *Virtuoso Analog and Design Environment XL User Guide*.

Example

This example shows the average value calculated when you apply the `average` function on the following input signal:



When you apply the `average` function on this signal, the following expression is created in the Buffer:

```
average(i("V1:p" ?result "tran" ?resultsDir "./ampsim.raw"))
```

Now, when you evaluate this expression, the following output is displayed in the Buffer:

-1.402E-3

Related OCEAN Function

The equivalent OCEAN function for `average` is:

```
average( o_waveform )
     => n_average/o_waveformAverage/nil
```

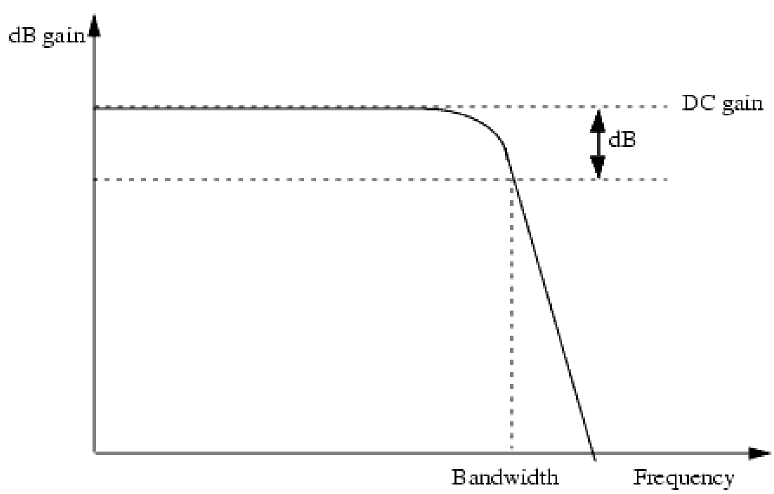
For more information, see average in *OCEAN Reference*.

bandwidth

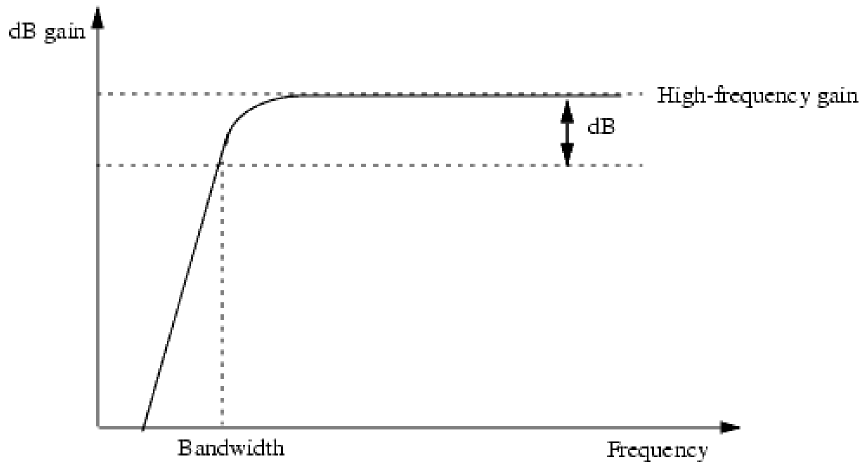
Calculates the bandwidth of a waveform. The output in MDL includes the unit, in SKILL it does not.

This function includes the following fields:

- *Signal* —Name of the signal. In SKILL mode, Virtuoso Visualization and Analysis XL wraps the signal with the `mag`, `dB`, or `log` function. In MDL mode, you need to wrap the signal name with the `mag`, `dB`, or `log` function; otherwise, the tool returns an error.
- *Db*—Decibels down from the peak, which means how far below the peak value you want to see data. In SKILL mode, *Db* is a number equal to or greater than zero. In MDL mode, *Db* is a number less than zero.
- *Type*—Response type.
  - When `low`, computes the low-pass bandwidth by determining the smallest frequency at which the magnitude of the input waveform drops *`Db`* decibels below the DC gain. DC gain is obtained by zero-order extrapolation from the lowest or highest computed frequency, if necessary. An error occurs if the magnitude of the input waveform does not drop *`Db`* decibels below the DC gain.



- When `high`, computes the high-pass bandwidth by determining the largest frequency at which the magnitude of the input waveform drops *db* decibels below the gain at the highest frequency in the response waveform. An error occurs if the magnitude of the input waveform does not drop n decibels below the gain at high frequency.



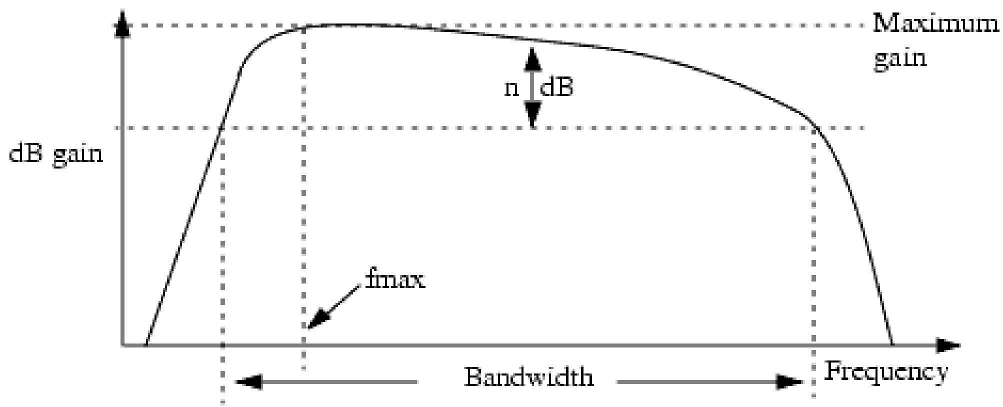
- When `band`, computes the band-pass bandwidth by:

  a. Determining the lowest frequency (f_{max}) at which the magnitude of the input waveform is maximized

  b. Determining the highest frequency less than f_{max} at which the input waveform magnitude drops *Db* decibels below the maximum

  c. Determining the lowest frequency greater than f_{max} at which the input waveform magnitude drops *Db* decibels below the maximum

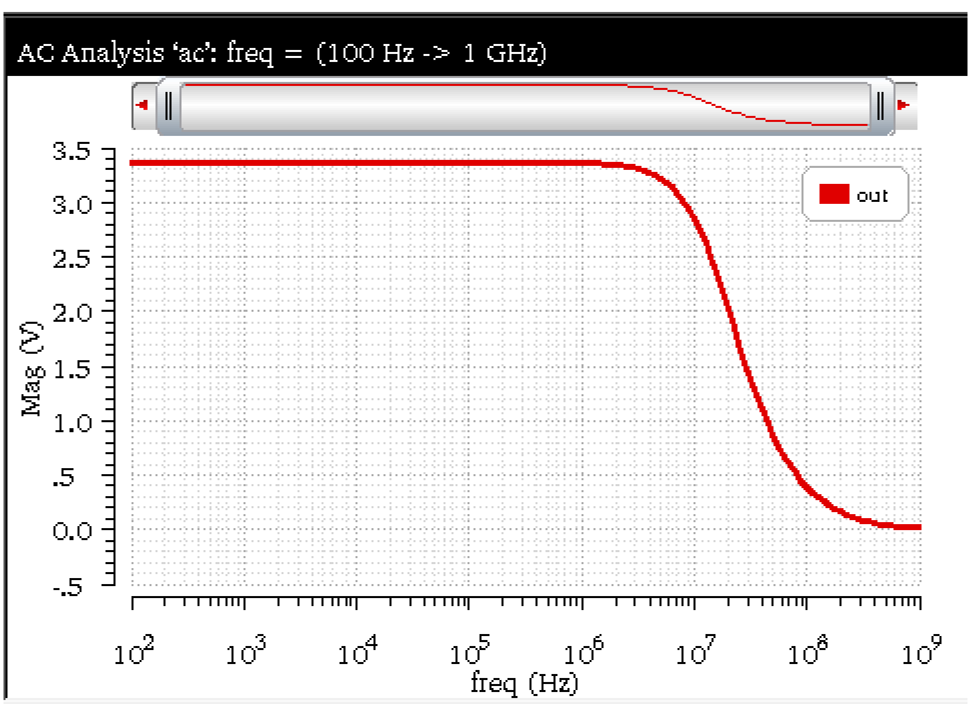
d. Subtracting the value returned by - *step b* from the value returned by step c. The value returned by step b or step c must exist.



Note: The bandwidth function includes the *Db* option; however, the signal is magnitude. This function modifies the magnitude signal to db scale internally.

Example

This example calculates the bandwidth of the following AC voltage signal, `v("out" ?result "ac")`



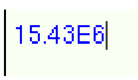
The following arguments are specified in this example:

- *Signal*—`v("out" ?result "ac")`
- *Db*—`3`
- *Type*—`low`

The following expression is created in the Buffer:

```
bandwidth(mag(v("out" ?result "ac" ?resultsDir "./ampTest.raw")) 3
"low" )
```

Now, when you evaluate this expression, the following output value is displayed in the Buffer as result:



Related OCEAN Function

The equivalent OCEAN function for `bandwidth` is:

```
bandwidth( o_waveform n_db t_type )
     => n_value/o_waveform/nil
```

For more information about this OCEAN function, see <u>bandwidth</u> in *OCEAN Reference*.

clip

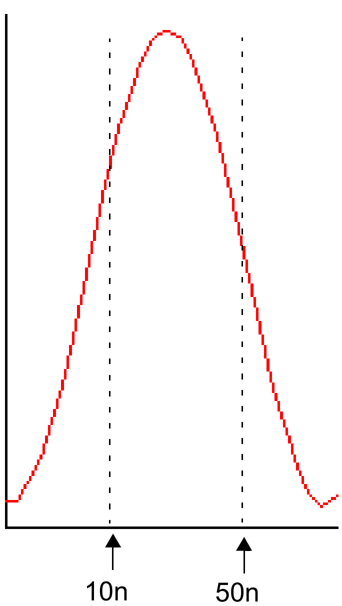
Returns the portion of a signal between two points along the X-axis.You can use the clip function to restrict the range of action of other special functions of the calculator such as `integ`, `rms`, and `frequency`.

- *Signal*—Displays the name of the signal.
- *From*—Starting point on the X-axis from where the clipping is to be started.
- *To*—Ending point on the X-axis where the clipping is to be ended.

The clip function in the SKILL mode is similar to the trim function in MDL mode.

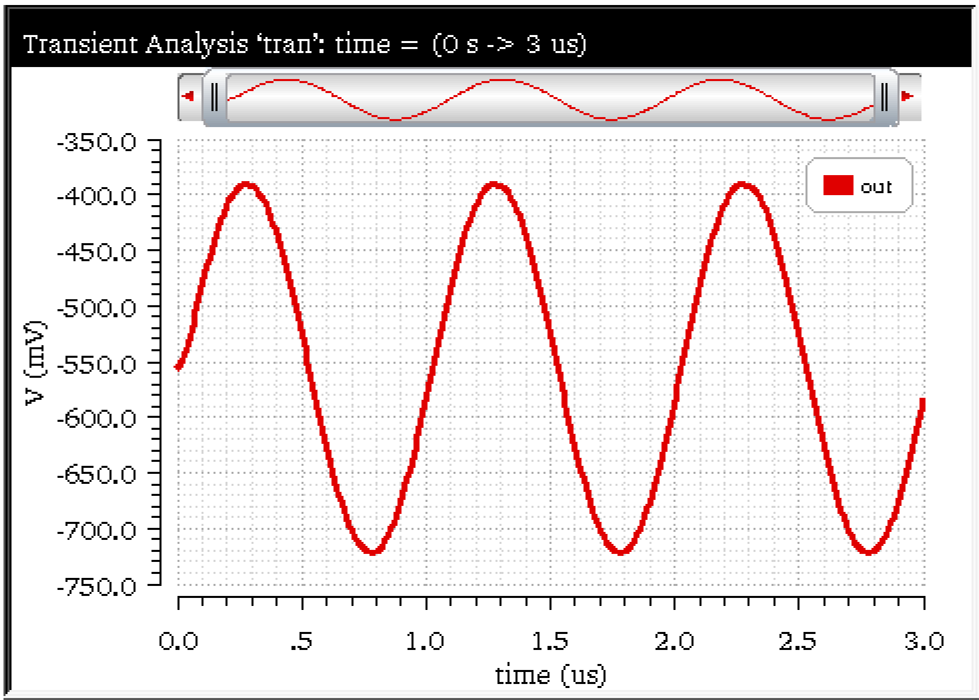
Example 1

The following diagram illustrates how the result with the values clip `(sig=V(sinewave), from=10n, to=50n )` is determined.



Example 2

This function shows the clipped waveform generated when you apply `clip` function on the following input signal:



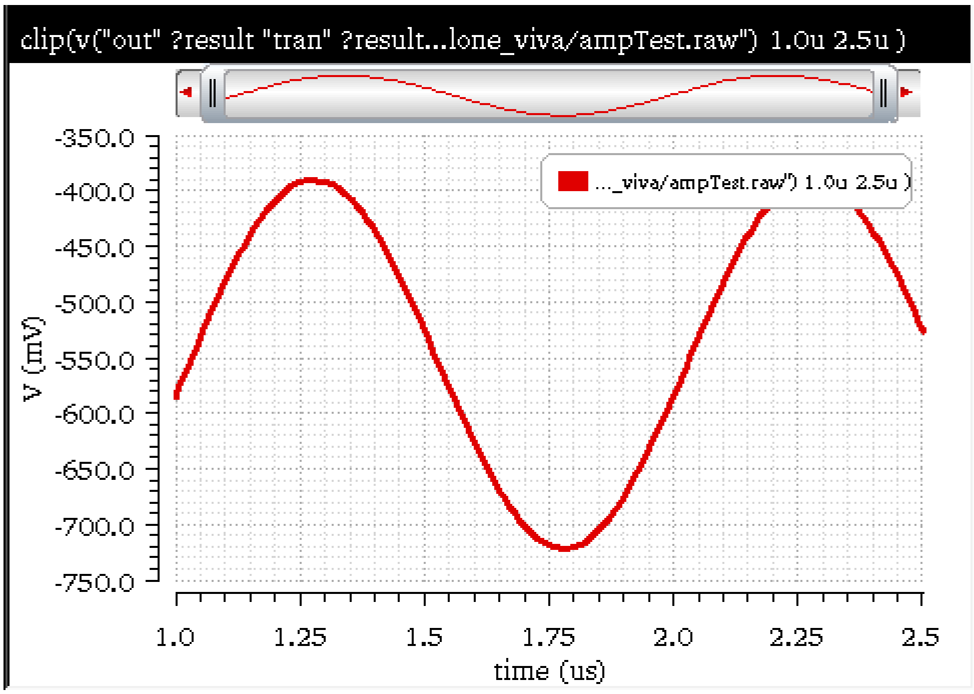
The following arguments are specified in this example:

- *Signal*—`v("out" ?result "tran")`
- *From*—`1.0u`
- *To*—`2.5u`

The following expression is created in the Buffer:

```
clip(v("out" ?result "tran" ?resultsDir "./ampTest.raw") 1.0u 2.5u )
```

Now, when you evaluate this expression, the following output is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN function for `clip` is:

```
clip( o_waveform n_from n_to )
     => o_waveform/nil
```

For more information about this OCEAN function, see clip in *OCEAN Reference*.

compare

Compares the two given waveforms based on the specified values for absolute and relative tolerances. This function compares only the sections of the two waveforms where the X or independent axes overlap.

- *Signal1*—Name of the first signal.
- *Signal2*—Name of the second signal.
- *Absolute Tolerance*—Specifies the absolute tolerance.
- *Relative Tolerance*—Specifies the relative tolerance.

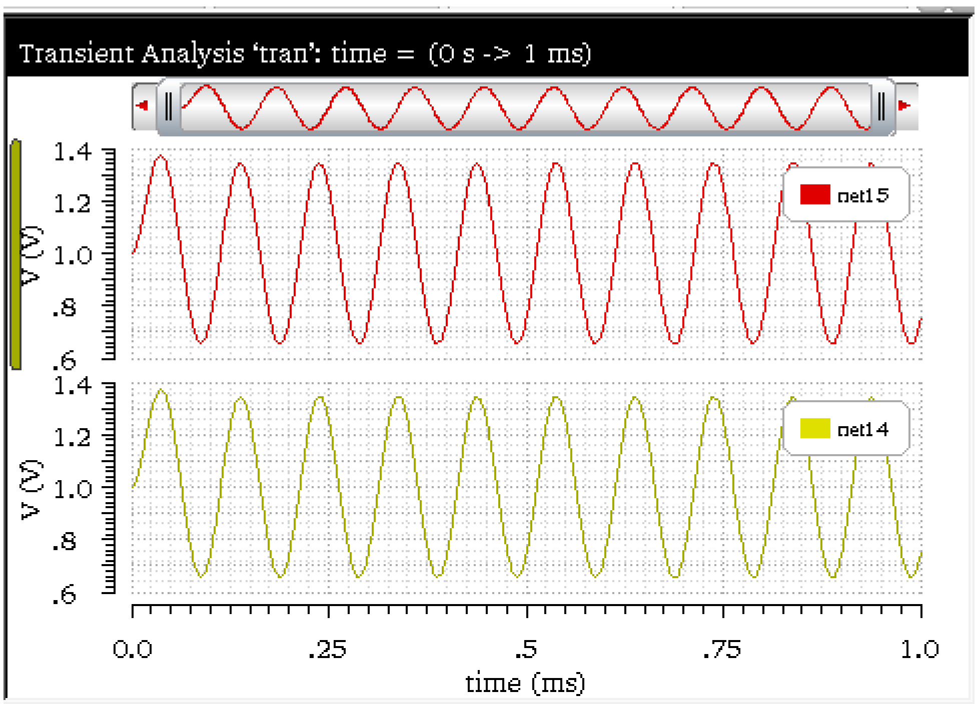
  The following situations are valid:

  - If neither relative nor absolute tolerance is specified, the function returns the difference of the two waveforms (Signal1 - Signal2).
  - If only the absolute tolerance is specified, the function returns the difference of the two waveforms only when the absolute value of the difference is greater than the absolute tolerance (|Signal1 - Signal2| > f_abstol); otherwise it returns a zero waveform.
  - If only the relative tolerance is specified, the function returns the difference of the two waveforms only when the absolute value of the difference is greater than the product of the relative tolerance and the larger of the absolute values of the two waveforms (|Signal1 - Signal2| > f_reltol * max(|Signal1|, |Signal2|)); otherwise it returns a zero waveform.
  - If both relative and absolute tolerances are specified, the function returns the difference of the two waveforms only when the absolute value of the difference is greater than the sum of the separately calculated tolerance components (|Signal1 - Signal2| > f_abstol + f_reltol * max(|Signal1|, |Signal2|)); otherwise it returns a zero waveform.

Note: The function also compares parametric waveforms. However, for a successful comparison of parametric waveforms, the family tree structures of the two input waveforms should be the same. For both the input waveforms, the number of child waveforms at each level should also be the same, except at the leaf level where the elements are simple scalars.

Example

This example shows the output waveform generated when you apply the `compare` function on the following input signals:



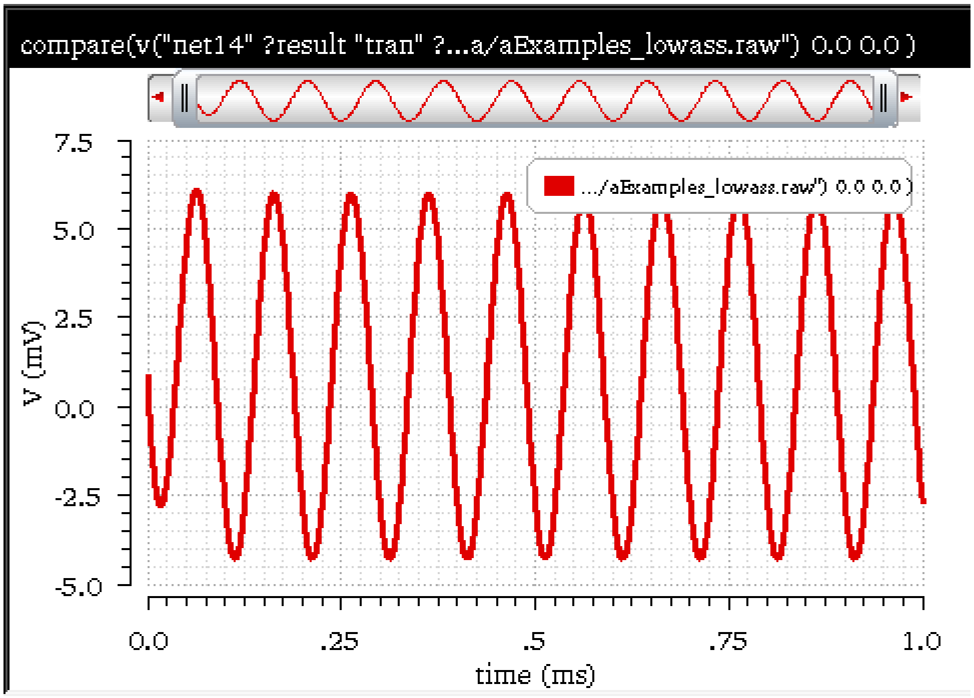
The following arguments are specified in this example:

- *Signal1*—`v("net14" ?result "tran")`
- *Signal2*—`v("net15" ?result "tran")`
- *Absolute Tolerance*—`0.0`
- *Relative Tolerance*—`0.0`

The following expression is created in the Buffer:

```
compare(v("net14" ?result "tran" ?resultsDir "./aExamples.raw")
v("net15" ?result "tran" ?resultsDir "./aExamples.raw") 0.0 0.0 )
```

Now, when you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN function for `compare` is:

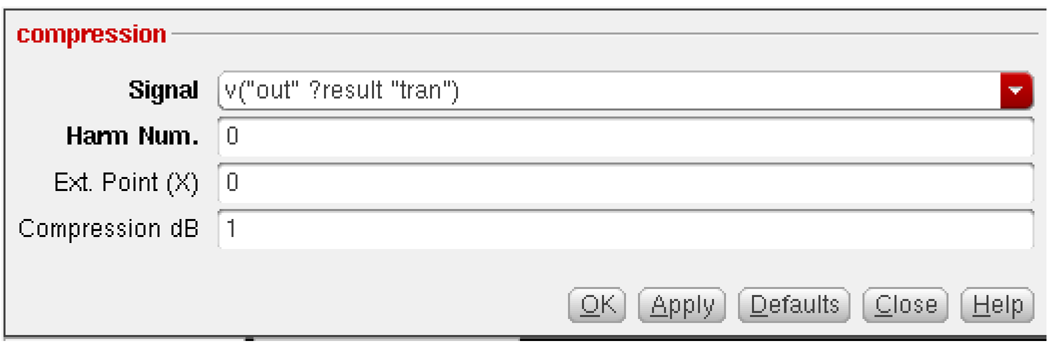
```
compare( o_waveform1 o_waveform1 [f_abstol [f_reltol]] )
     => o_comparisonWaveform/nil
```

For more information about this OCEAN function, see compare in *OCEAN Reference*.

compression

Returns the *N*th compression point value of a waveform at the specified extrapolation point. This function is available only in the SKILL mode.

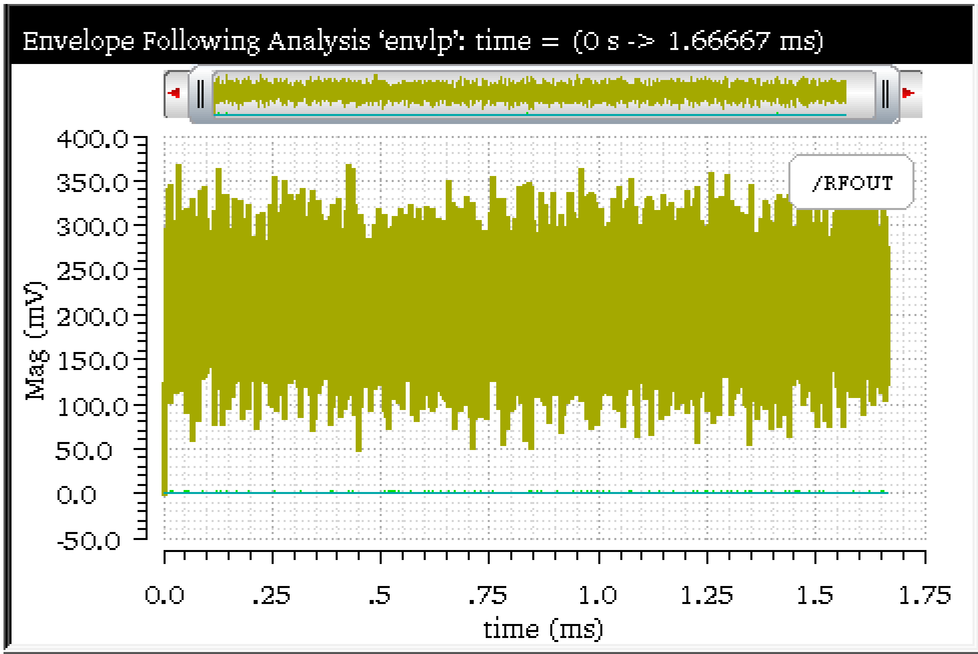
The compression function uses the power waveform to extrapolate a line of constant slope (dB/dB) according to a specified input or output power level. This line represents constant small-signal power gain (ideal gain). The function finds the point where the power waveform drops n dB from the constant slope line and returns either the X coordinate (input referred) value or the Y coordinate (output referred) value.



- *Signal*—Name of the signal.
- *Harm Num*—Harmonic number.
- *Ext. Point (X)*—Extrapolation point of the waveform. The extrapolation point is the X-axis value.
- *Compression dB*—Compression coefficient (N).

Example

This example shows the compression value generated when you apply the `compression` function on the following input waveform:



The following arguments are specified in this example:

- *Signal*—`v("/RFout")`
- *Harm Num.*—`1`
- *Ext. Point (X)*— `-30`
- *Compression dB*—`1`

The following expression is created in the Buffer:

```
compression(dB20(harmonic(mag(v("/RFOUT" ?result "envlp_fd"
?resultsDir "./viva/rfworkshop/simulation/EF_example_envlp/spectre/
schematic/psf"")), 1)), ?x -30, ?compress 1)
```

Now, when you evaluate this expression, the following output value is displayed in the Buffer:

```
21.96E-9
```

Related OCEAN Function

The equivalent OCEAN function for `compression` is:

```
compression( o_waveform [ ?x f_x ] [ ?y f_y ]
     [ ?compression f_compression ] [ ?io s_measure ] )
     => f_compPoint/nil
```

For more information about this OCEAN function, see <u>compression</u> in *OCEAN Reference*.

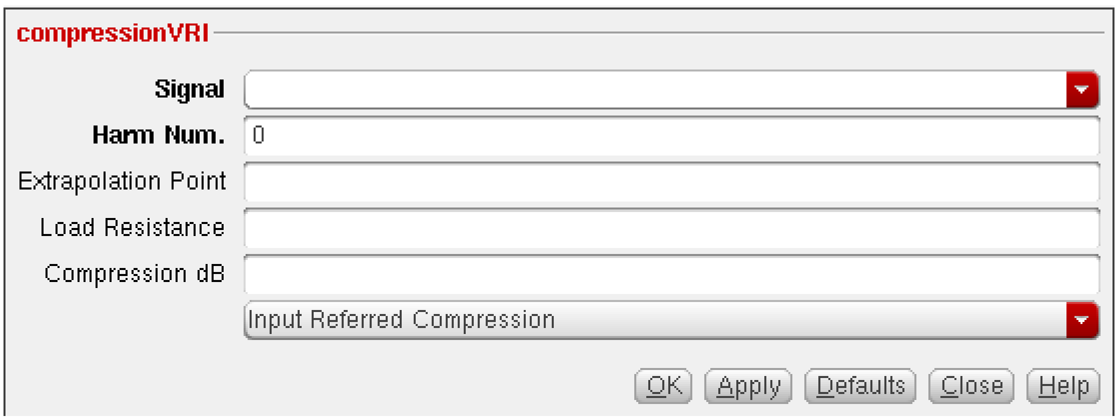
compressionVRI

Performs an *N*th compression point measurement on a power waveform. This function is available only in the SKILL mode.

You can use this function to simplify the declaration of a compression measurement.

This function extracts the specified harmonic from the input waveform(s), and uses dBm(spectralPower((i or v/r),v)) to calculate a power waveform. The function passes this power curve and the remaining arguments to the compression function to complete the measurement.

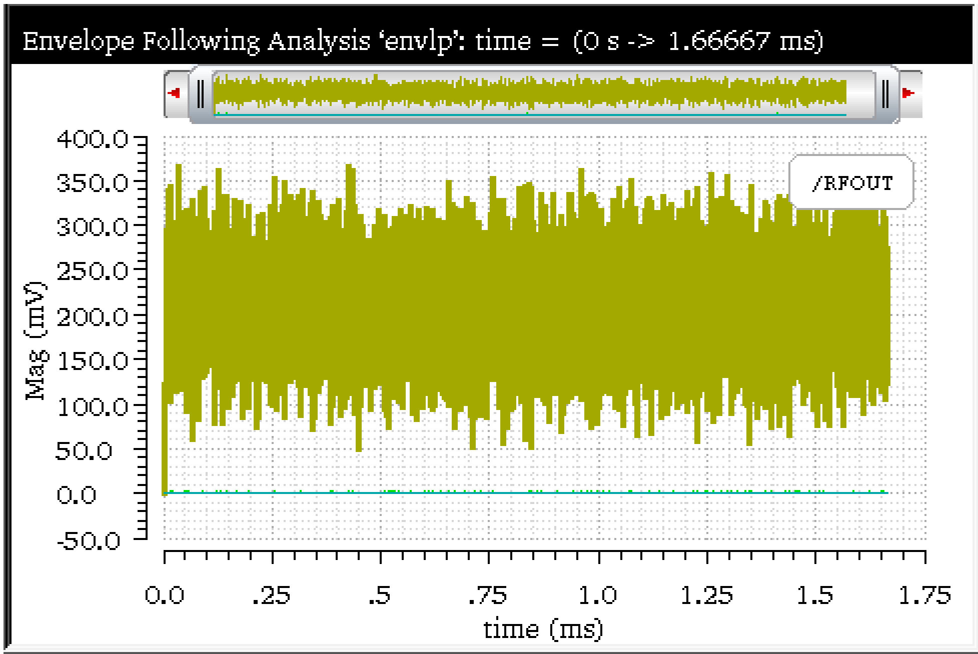
The compression function uses the power waveform to extrapolate a line of constant slope (dB/dB) according to a specified input or output power level. This line represents constant small-signal power gain (ideal gain). The function finds the point where the power waveform drops n dB from the constant slope line and returns either the X coordinate (input referred) value or the Y coordinate (output referred) value.



- *Signal*—Name of the signal.
- *Harm Num*—Harmonic index of the waveform.
- *Ext. Point (X)*—Extrapolation point for the waveform. The default value is the minimum x value of the input voltage waveform.
  The extrapolation point is the coordinate value in dBm that indicates the point on the output power waveform where the constant-slope power line begins. This point should be in the linear region of operation.
- *Load Resistance*—Resistance value. The default value is 50.
- *Compression dB*—Specifies the delta (in dB) between the power waveform and the ideal gain line that marks the compression point. The default value is 1.

Example

This example shows the compression value generated when you apply the `compressionVRI` function on the following input waveform:



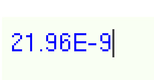
The following arguments are specified in this example:

- *Signal*—`v("/RFout")`
- *Harm Num.*—`1`
- *Extrapolation Point*— `-30`
- Load Resistance—`50`
- *Compression dB*—`1`

The following expression is created in the Buffer:

```
compressionVRI(mag(v("/RFOUT" ?result "envlp_fd" ?resultsDir "./
viva/rfworkshop/simulation/EF_example_envlp/spectre/schematic/
psf")) 1 ?gcomp 1 ?epoint -30 ?rport 50 )
```

Now, when you evaluate this expression, the following output value is displayed in the Buffer:



Related OCEAN Function

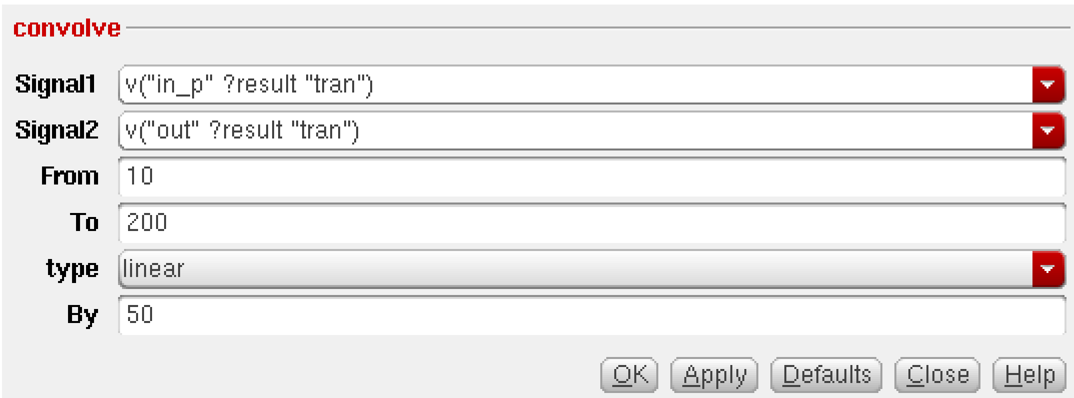
The equivalent OCEAN function for `compressionVRI` is:

```
compressionVRI( o_vport x_harm [?iport o_iport] [?rport f_rport]
     [?epoint f_epoint] [?gcomp f_gcomp] [?measure s_measure] )
     => o_waveform/n_number/nil
```

For more information about this OCEAN function, see compressionVRI in *OCEAN Reference*.

convolve

Returns a waveform consisting of the time domain convolution of two signals.



This function includes the following arguments in the SKILL mode:

- *Signal1*—Name of the first signal.
- *Signal2*—Name of the second signal.
- *From*—Starting point (X-axis value) of the integration range.
- *To*—End point (X-axis value) of the integration range.
- *type*—Specifies whether is the interpolation is linear or log.
- *By*—Specifies the step size.

In the MDL mode,

- *sig1* is the name of the first signal.
- *sig2* is the name of the second signal.
- *n_interp_steps* is the number of steps for interpolating waveforms.

If you want to specify an integration range like the SKILL mode, you can use the trim function.

Covolution is defined by the following equation:

```
ifft( fft(f1(s)) * fft(f2(s)) )
```

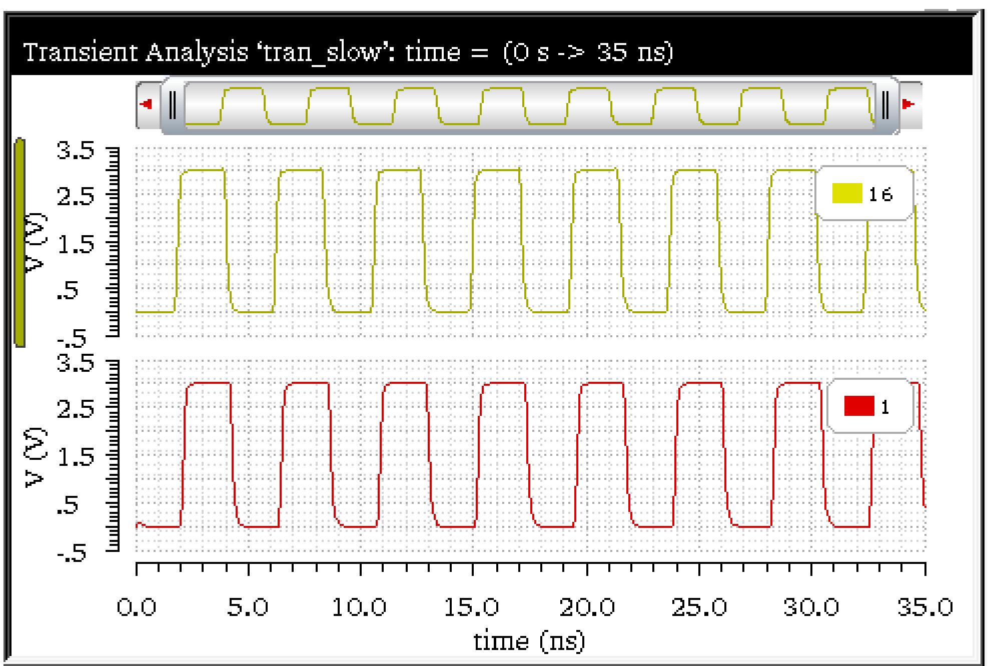
Additional Information

Covolution is defined by the following equation:

$$\int_{from}^{to} f1(s)f2(t-s)ds$$

Example

This example calculates the time domain convolution of the following two transient voltages, `v("/1")` and `v("/16")` and displays the output waveform generated:



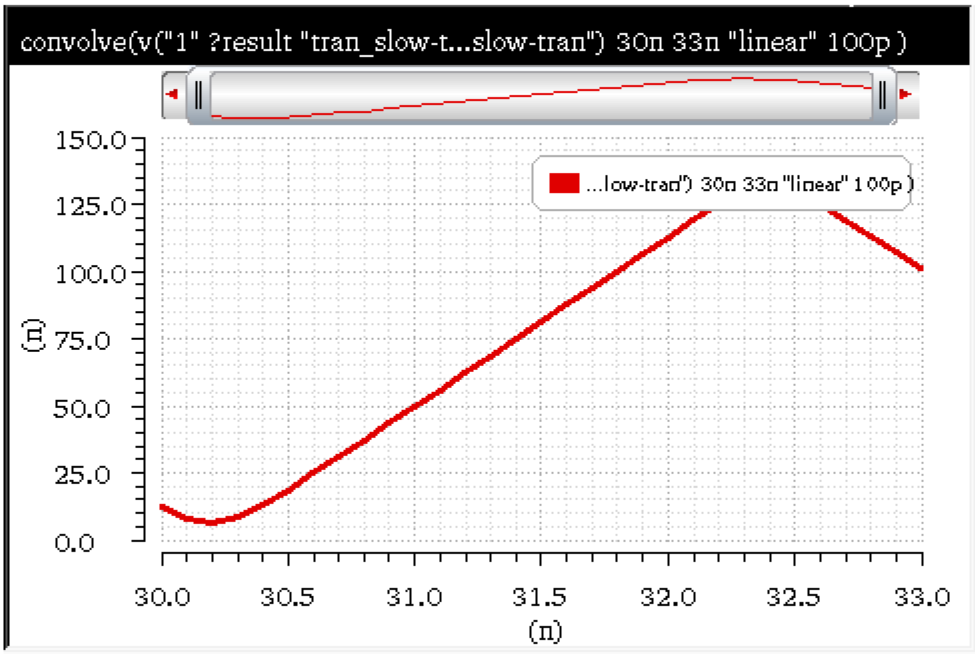
The following arguments are specified in this example:

- *Signal1*–`v("1" ?result "tran")`
- *Signal2*–`v("16" ?result "tran")`
- *From*–`30n`
- *To*– `33n`
- *Type*–`linear`
- *By*–`100p`

The following expression is created in the Buffer:

```
convolve(v("1" ?result "tran") v("16" ?result "tran") 30n 33n
"linear" 100p )
```

Now, when you evaluate this expression, the following output waveform showing the time domain convolution is displayed in the graph window:



Related OCEAN Function

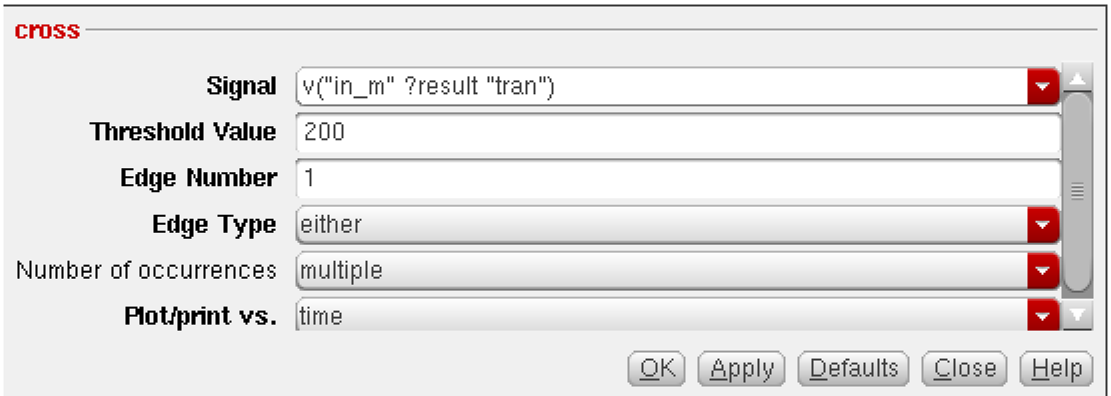
The equivalent OCEAN function for `convolve` is:

```
convolve( o_waveform1 o_waveform2 n_from n_to t_type n_by )
    => o_waveform/n_number/nil
```

For more information about this OCEAN function, see <u>convolve</u> in *OCEAN Reference*.

cross

Returns the X value where a signal crosses the threshold Y value.



This function includes the following arguments:

- *Signal*—Name of the signal.
- *Threshold Value*—Threshold to be crossed.
- *Edge Number*—Occurrence of the crossing. The first crossing is *Edge Number*=`1`, the second crossing is *Edge Number*=`2`, and so on. The value of *Edge Number* can be negative numbers: *Edge Number*=`-1` for the previous occurrence, *Edge Number*=`-2` for the occurrence before the previous occurrence, and so on.
- *Edge Type*—Direction of the crossing event. `rising` directs the function to look for crossings where the Y value is increasing, `falling` for crossings where the Y value is decreasing, and `either` for crossings in either direction.
- Number of occurrences—Select `single` to calculate one point or select `multiple` to calculate until the end of X range.
- Plot/print vs.—Specifies whether X-axis of the output waveform is time (or another X-axis parameter for non-transient data) or cycle.

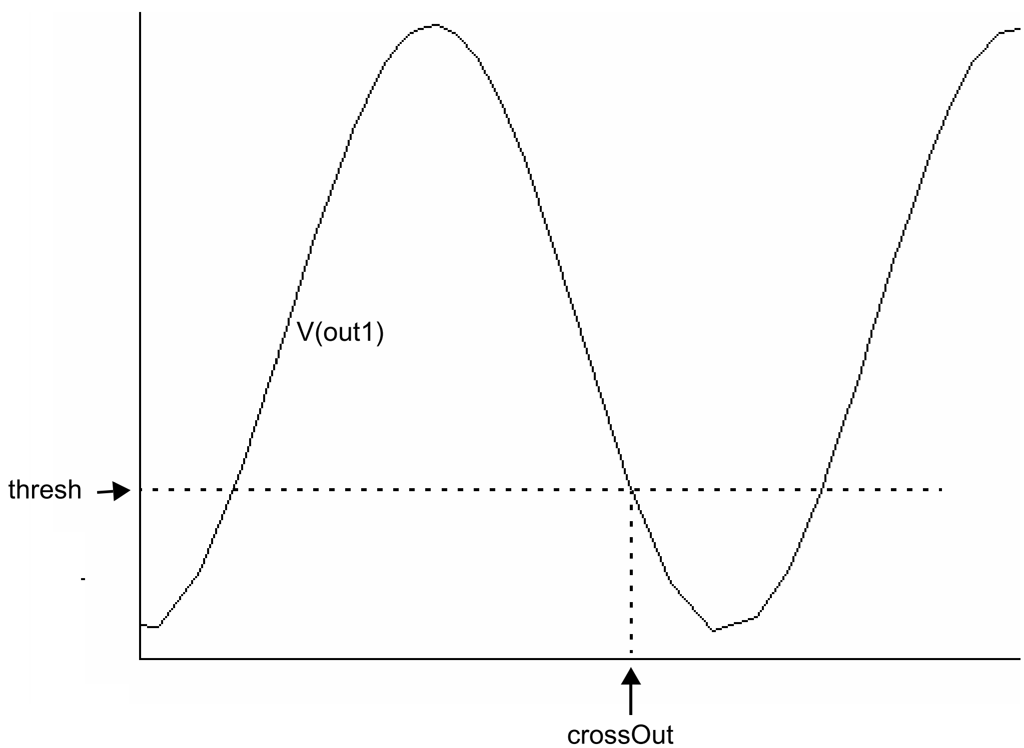
In the MDL mode,

- *sig*—Name of the signal.
- *dir*—Direction of the crossing event. `'rise` directs the function to look for crossings where the Y value is increasing, `'fall` for crossings where the Y value is decreasing, and `'cross` for crossings in either direction.

- *n*—Occurrence of the crossing. The first crossing is *n*=`1`, the second crossing is *n*=`2`, and so on. The value of n can be negative numbers: *n*=`-1` for the previous occurrence, *n*=`-2` for the occurrence before the previous occurrence, and so on.
- *thresh*—Threshold to be crossed.
- *start*—Time at which the function is enabled.
- *xtol*—Absolute tolerance in the X direction.
- *ytol*—Absolute tolerance in the Y direction.
- *accuracy*—Specifies whether the function should use interpolation, or use iteration controlled by the absolute tolerances to calculate the value. `'interp` directs the function to use interpolation, and `'exact` directs the function to consider the xtol and yval values.

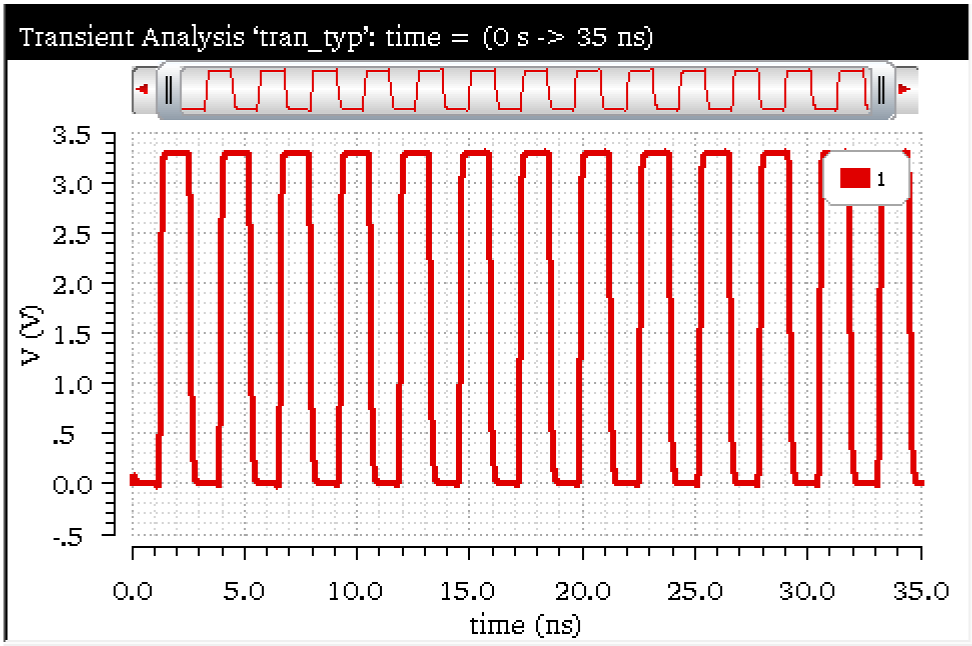
Example 1

The following diagram illustrates how the result is determined for the values *signal*=`V(out1)`, *Threshold Value*=`1`, *Edge Number*=`1`, and *Edge Type*=`falling`



Example 2

This example shows the output waveform generated when you apply the `cross` function on the following input waveform:



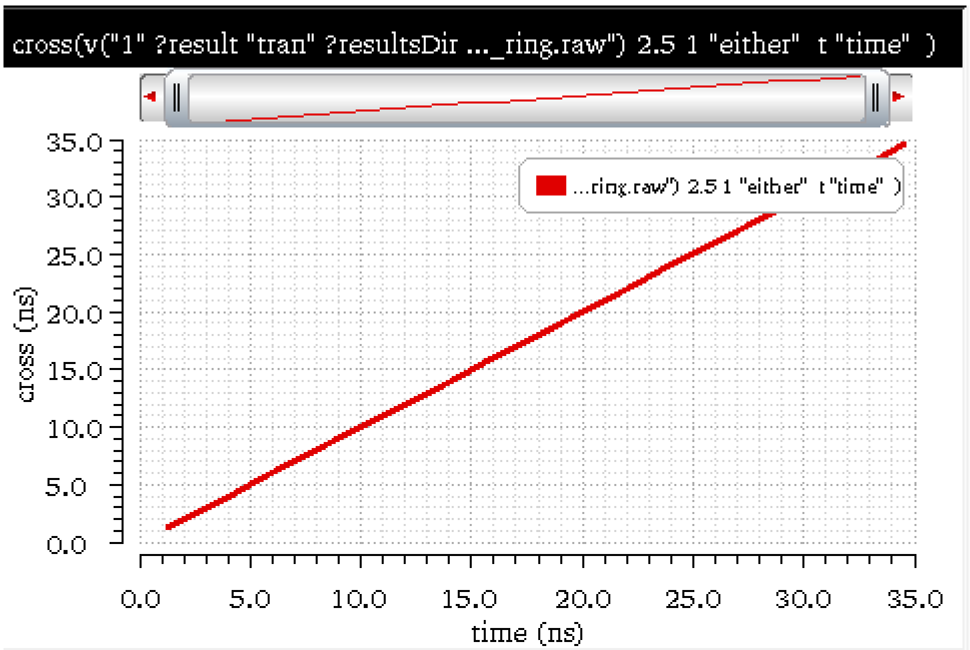
The following arguments are specified in this example:

- *Signal*—`v("1" ?result "tran")`
- *Threshold*—`2.5`
- *Edge Number*—`1`
- *Edge Type*—`either`
- *No of occurrences*—`multiple`
- *Plot/print vs.*—`time`

The following expression is created in the Buffer:

```
cross(v("1" ?result "tran") 2.5 1 "either"  t "time" )
```

Now, when you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

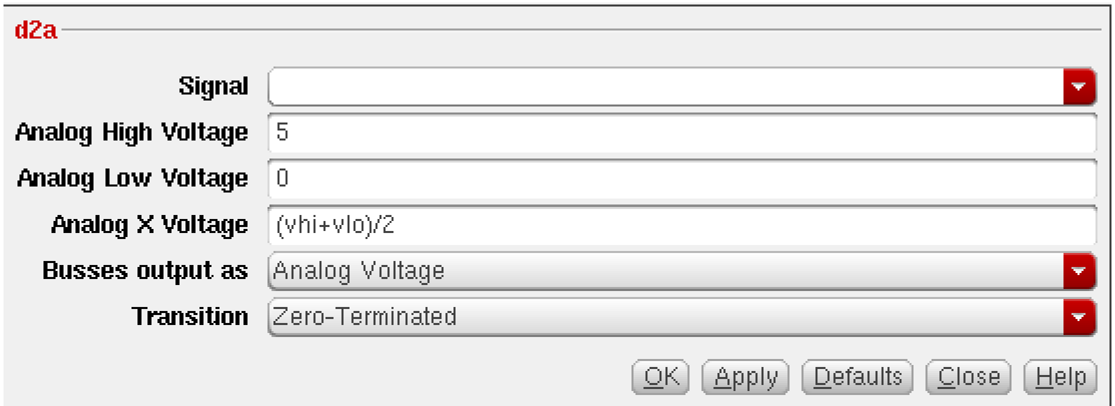
The equivalent OCEAN command for `cross` is:

```
cross( o_waveform n_crossVal x_n s_crossType [g_multiple [s_Xname]] )
     => o_waveform/g_value/nil
```

For more information about the OCEAN function, see <u>cross</u> in *OCEAN Reference*.

d2a

Returns the analog output from a given digital waveform.

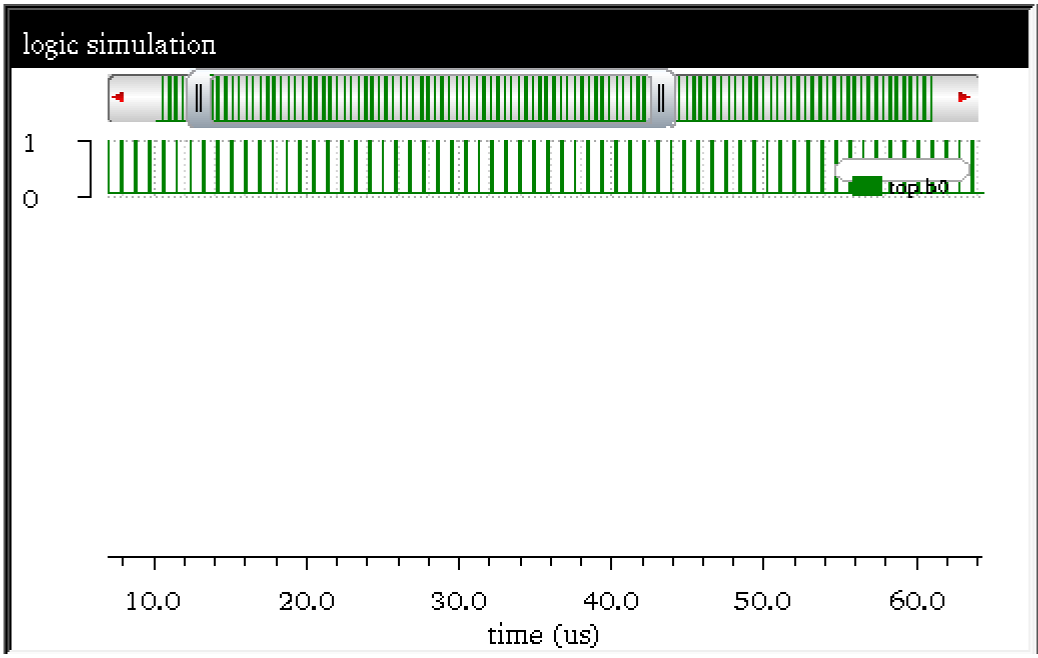


This function includes the following arguments:

- *Signal*—Name of the waveform, expression, or family of waveforms. The expression displayed in the Buffer can be added to this field by selecting *buffer*.
- *Analog High Voltage*—Highest voltage value when the digital signal is in the high-state (1)
- *Analog Low Voltage*—Lowest analog value when the digital signal is in the low-state (0)
- *Analog X Voltage*—Specify the corresponding analog value to which you want to convert the state X of the digital waveform. This value can be a number or a string expression of `vhi` and `vlo`, such as `(vhi+vlo)/2`
- *Busses output as*—Set this field to `Analog Voltage` or `Analog bits` if the given digital waveform is a bus.
- *Transition*—Set the transition to `Piecewise Linear` or `Zero-Terminated` if the given digital waveform is a bus.

Example

This example shows the conversion of digital waveform (logic signal) to an analog waveform. The input signal is as shown in the figure below:



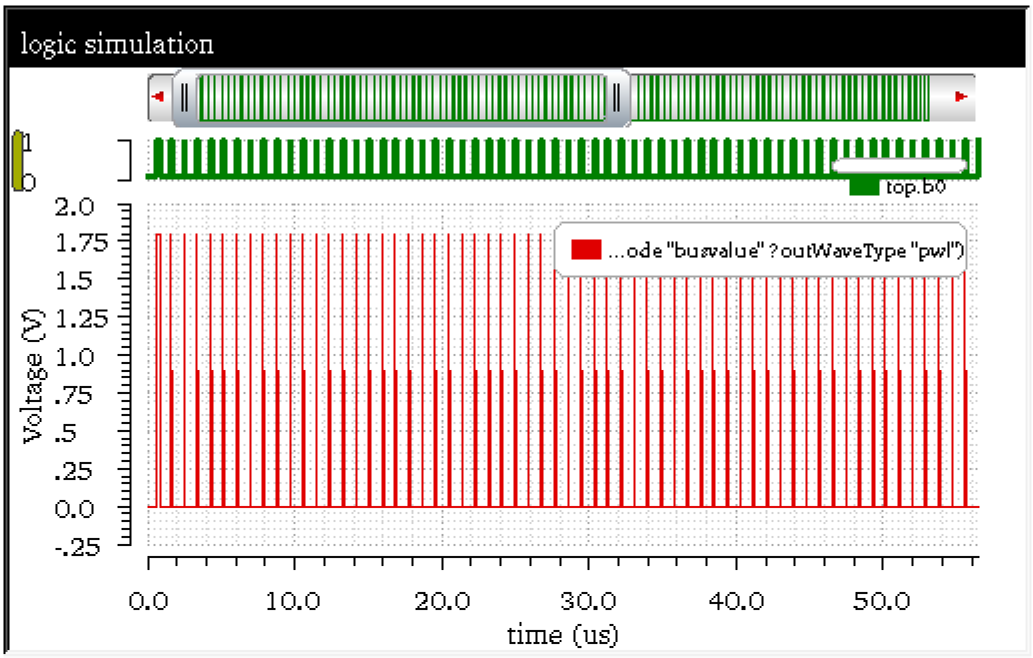
The following arguments are specified in this example:

- *Signal*—`getData(top.b0" ?result "waves.shm”)`
- *Analog High Voltage*—`1.8`
- *Analog Low Voltage*—`0`
- *Analog X Voltage*—`(vhi+vlo)/2`
- *Busses output as*—`Analog Voltage`
- *Transition*—`Piecewise-Linear`

The following expression is created in the Buffer:

```
awvDigital2Analog(getData("top.b0" ?result "waves.shm") 1.8 0
"(vhi+vlo)/2" ?mode "busvalue" ?outWaveType "pwl")
```

Now, when you evaluate this expression, the following output waveform, which is an analog signal, is displayed in the graph window.



Related SKILL Function

The equivalent SKILL command for `d2a` is:

```
awvDigital2Analog( o_waveform n_vhi n_vlo s_VX
     @key s_mode s_outWaveType s_vprevSTART )
     => o_waveform | nil
```

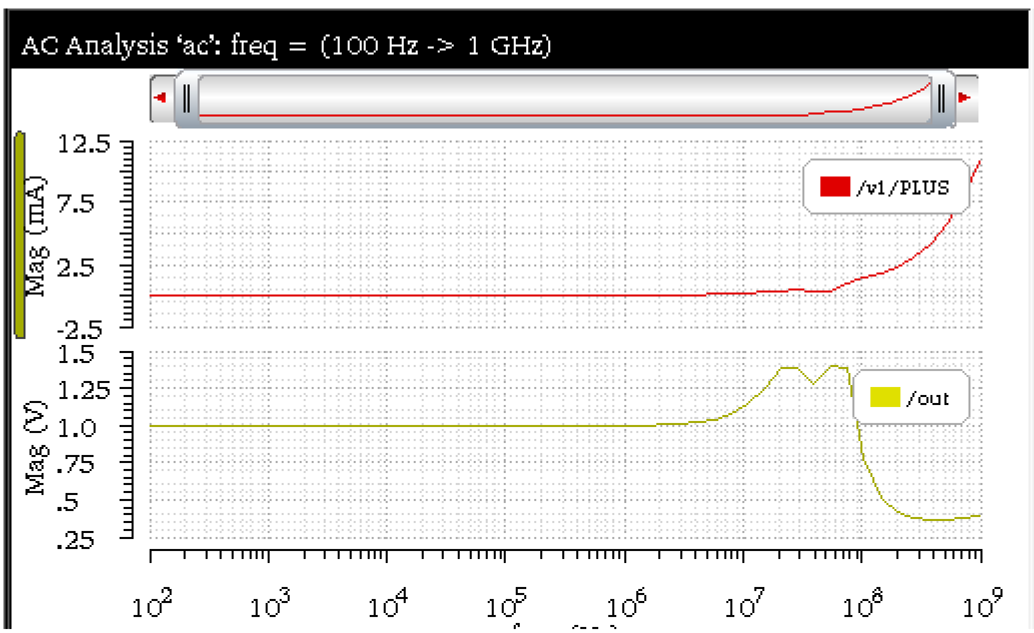
For more information about this SKILL function, see awvDigital2Analog in *Virtuoso Visualization and Analysis XL SKILL Reference*.

dBm

Returns 10 times the log10 of the specified waveform object plus 30. This function converts a signal, in watts, to dbm, where `dbm=10*log(x)+30`.

Example

This example shows the output waveform generated when you apply the `dBm` function on the following input waveforms:



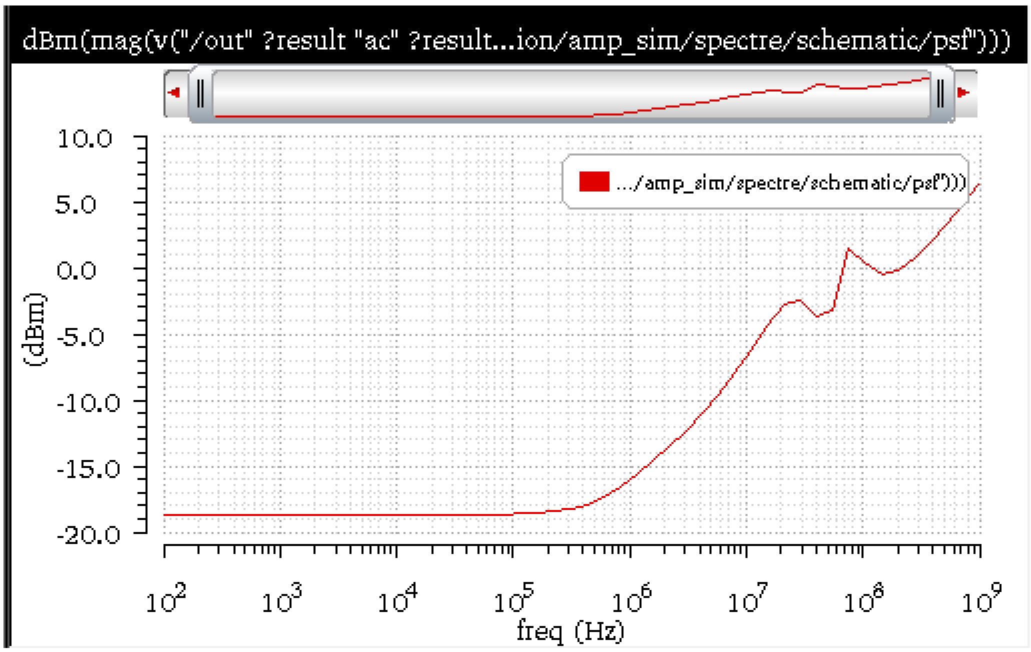
The above figure shows the current waveform, `/v1/PLUS` and voltage waveform, `/out`. If you want to calculate the `dBm` power of these signals, firstly you will have to multiply the terminal voltage and current signals.

```
mag(v("/out" ?result "ac"))*mag(i("/v1/PLUS" ?result "ac"
?resultsDir ./viva/ADE_ViVA/ADEviva/simulation/amp_sim/spectre/
schematic/psf))
```

When you apply the `dBm` function, the following expression is created in the Buffer:

```
dBm(mag(v("/out" ?result "ac"))*mag(i("/v1/PLUS" ?result "ac"
?resultsDir ./viva/ADE_ViVA/ADEviva/simulation/amp_sim/spectre/
schematic/psf)))
```

Now, when you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

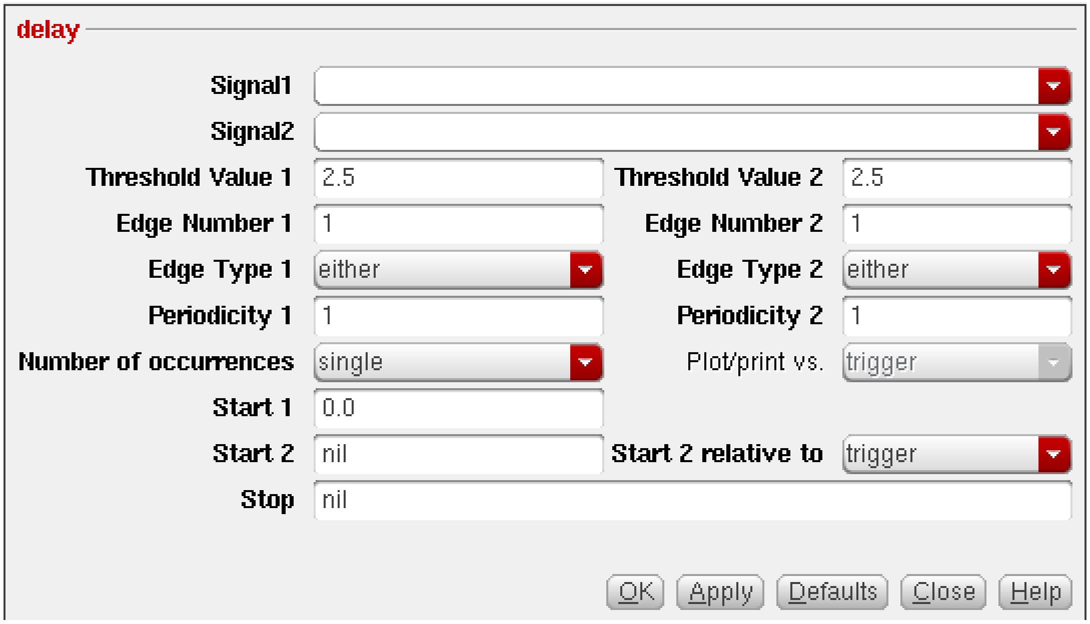
The equivalent OCEAN function for `dBm` is:

```
dbm( {o_waveform | n_number} )
     => o_waveform/n_number/nil
```

For more information about the OCEAN function, see dBm in *OCEAN Reference*.

delay

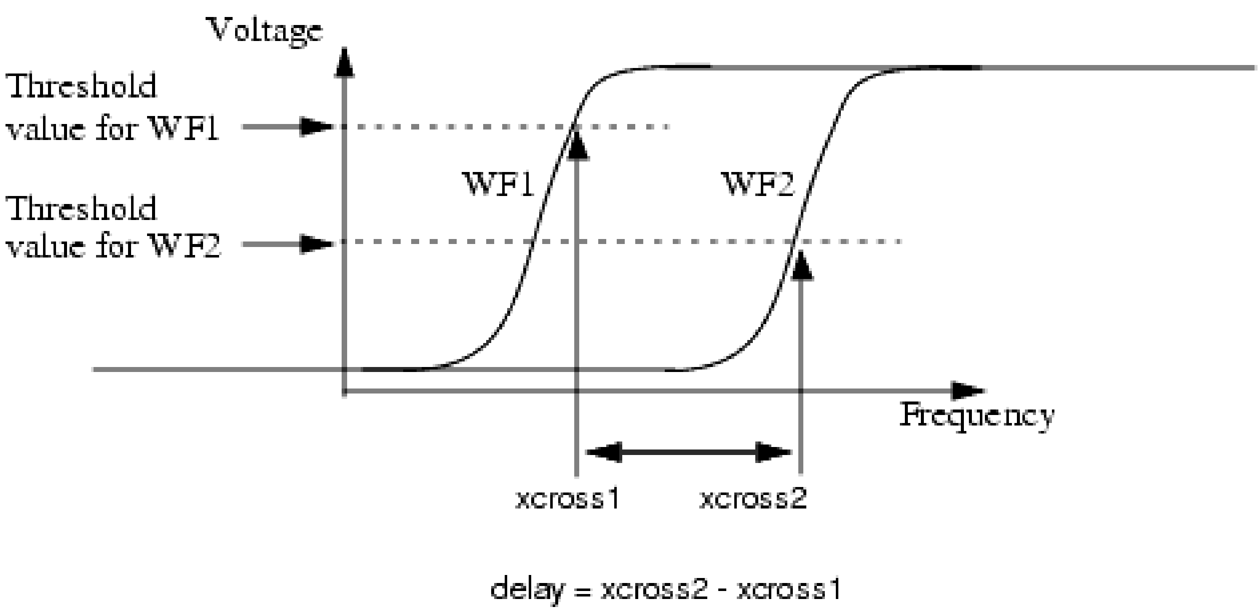
Computes the delay between two points or multiple sets of points in a waveform using the cross function. This function is available only in the SKILL mode.



This function includes the following arguments:

- *Signal1*—Name of the first signal.
- *Signal2*—Name of the second signal.
- *Threshold Value 1*—First threshold to be crossed.
- *Edge Number 1*—Number that specifies which crossing is to be the trigger event. For example, if *Edge Number1=2*, the trigger event is the second edge of the first waveform with the specified type that crosses *Threshold Value 1*.
- *Edge Type 1*—Direction of the first crossing event. `rising` directs the function to look for crossings where the Y value is increasing, `falling` for crossings where the Y value is decreasing, and `either` for crossings in either direction.
- *Periodicity 1*—Periodic interval for the first waveform. See Example 2 for details.
- *Threshold Value 2*—Second threshold to be crossed.
- *Edge Number* 2—Number that specifies which crossing is to be the trigger event. For example, if *Edge Number2=2*, the trigger event is the second edge of the second waveform with the specified type that crosses *Threshold Value 2*.

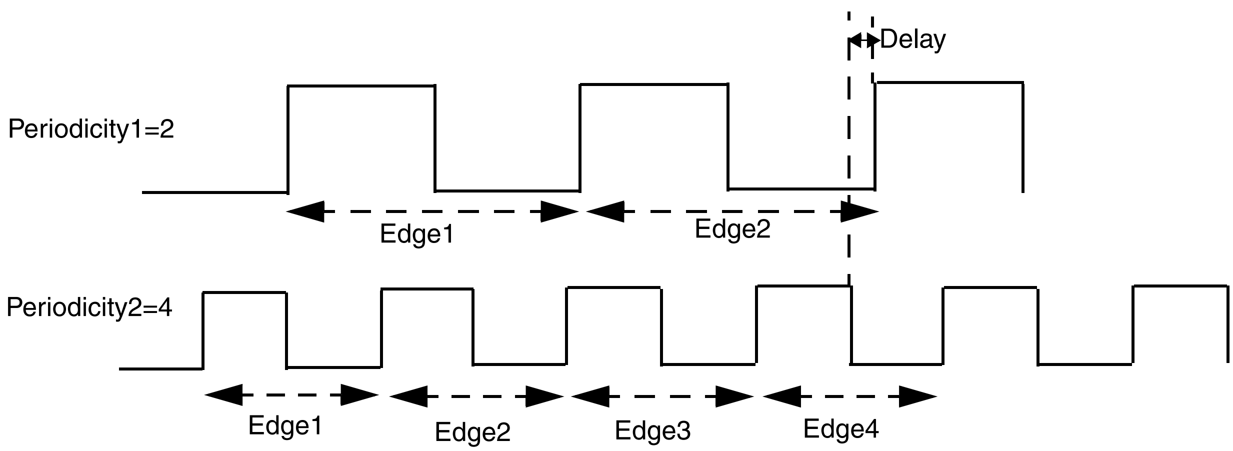
- *Edge Type 2*—Direction of the second crossing event. `rising` directs the function to look for crossings where the Y value is increasing, `falling` for crossings where the Y value is decreasing, and `either` for crossings in either direction.
- *Periodicity 2*—Periodic interval for the second waveform. See Example 2 for details.
- *Number of occurrences*—Specifies whether you want to retrieve only one occurrence of a delay event for the given waveform (*single*), or all occurrences of overshoot for the given waveform which you can later plot or print (*multiple*).
- *Plot/print vs.*—Specifies whether you want to retrieve delay data against *trigger* time, *target* time (or another X-axis parameter for non-transient data) or *cycle*. Cycle numbers refer to the n'th occurrence of the delay event in the input waveform.
  The value in this field is ignored when you specify *Number of Occurences* as *single*.
- *Start 1*—Time that specifies when the delay measurement is to be started.
- *Start 2*—Time to start observing the target event.
- *Start 2 relative to*—Specifies whether the Start 2 is relative to trigger or time.
- *Stop*—Time to stop observing the target event.



Example 1

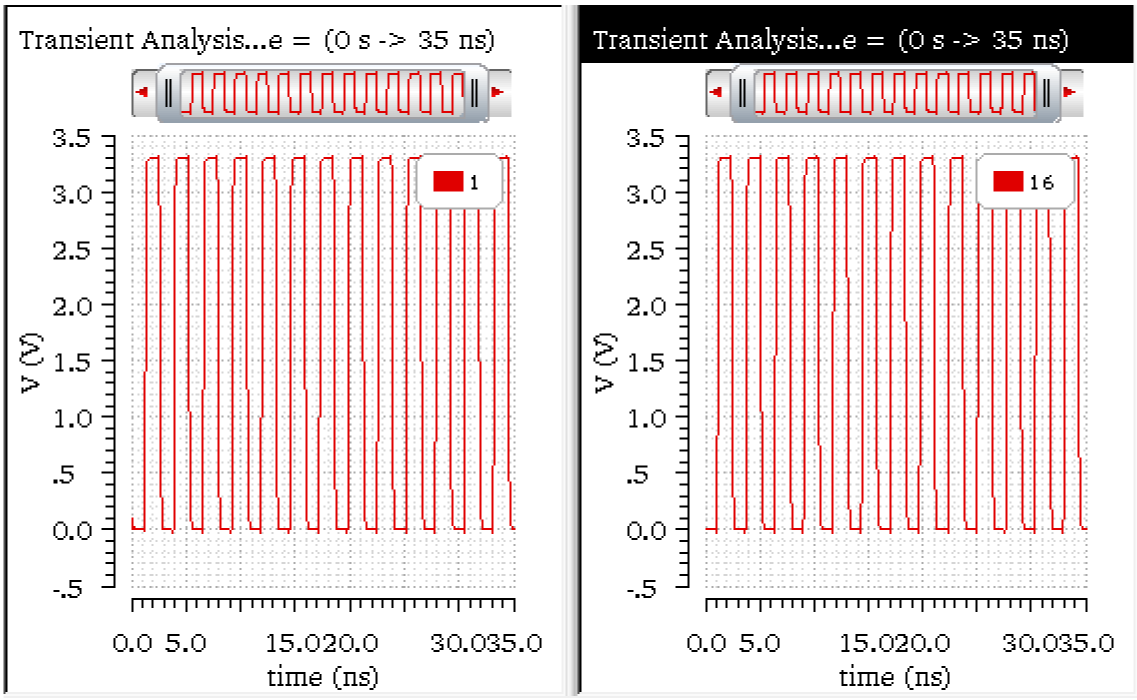
The delay algorithm for multiple occurrences returns the difference between the X points for the specified edges on the respective waveforms (with the specified periodicity). For example:

if the edge number specified for both the waveforms is 1 and periodicity is 2, then it will compute the difference between the 1,3,5,7 ... edges for both the waveforms.



Example 2

This examples shows the output waveform generated when you apply the `delay` function on the following input waveform:



The arguments specified in this example are:

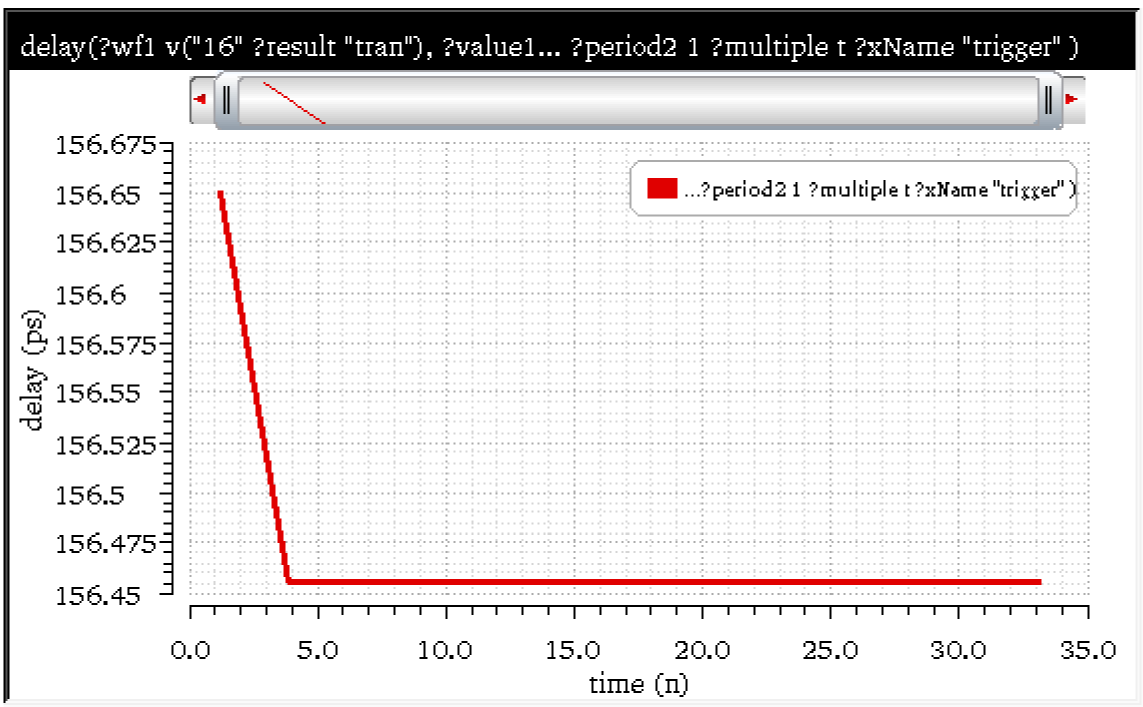
- *Signal1*—`v("1" ?result "tran")`
- *Signal2*—`v("16" ?result "tran")`
- *Threshold Value 1*—`2.5`
- Threshold Value 2—`2.5`
- *Edge Number 1*—`1`
- *Edge Number 2*—`1`
- *Edge Type 1*—`rising`
- *Edge Type 2*—`rising`
- *Periodicity* 1—`1`
- *Periodicity 2*—`1`
- *Number of occurrences*—`multiple`

- *Plot/print vs.*—`trigger`
- *Start 1*—`0.0`
- *Start 2*—`nil`
- *Start 2 relative to*—`trigger`
- *Stop*—`nil`

The following expression is created in the Buffer:

```
delay(?wf1 v("16" ?result "tran"), ?value1 2.5, ?edge1 "rising",
?nth1 1, ?td1 0.0, ?wf2 v("1" ?result "tran"), ?value2 2.5, ?edge2
"rising", ?nth2 1,  ?td2 nil , ?stop nil,  ?period1 1 ?period2 1
?multiple t ?xName "trigger" )
```

Now, when you evaluate this expression, the following output waveform showing the delay in two signals is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `delay` is:

```
delay( ?wf1 o_waveform1 ?value1 n_value1 ?edge1 s_edge1
     ?nth1 x_nth1 ?td1 n_td1
     ?wf2 o_waveform2 ?value2 n_value2 ?edge2 s_edge2
     ?nth2 x_nth2 {[?td2 n_td2]
     | [?td2r0 n_td2r0]} ?stop n_stop @rest args
     g_histoDisplay][x_noOfHistoBins])
     => o_waveform/n_value/nil
```

For more information about this OCEAN function, see delay in *OCEAN Reference*.

deriv

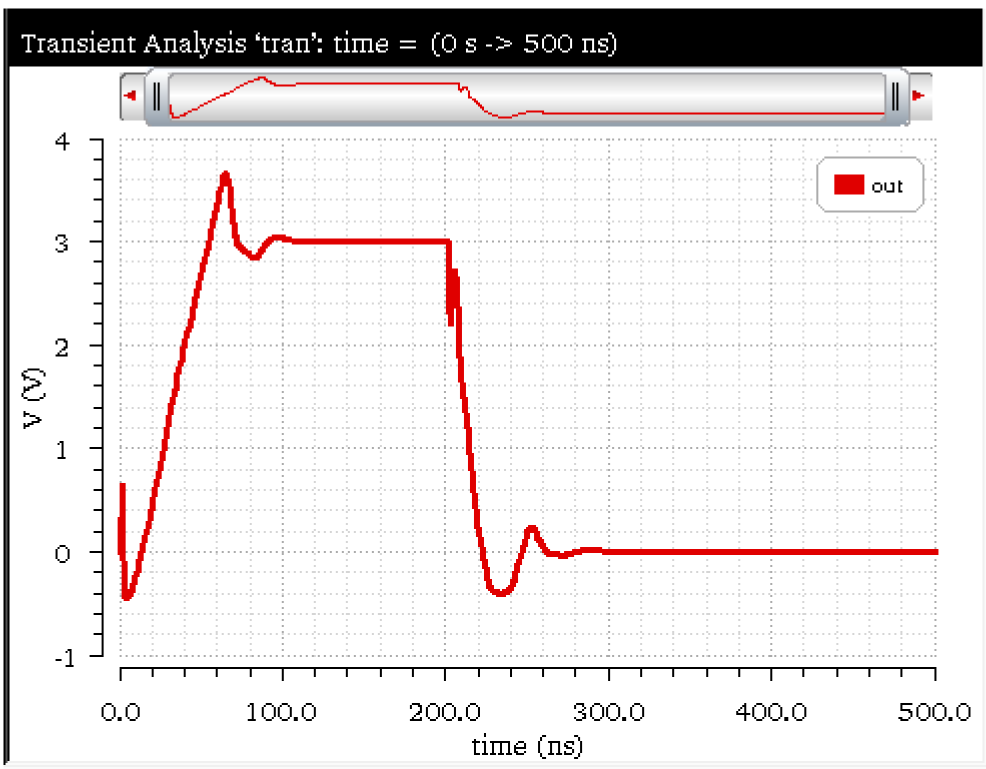
Returns the derivative of a signal with respect to the X-axis.

Note:

- After the second derivative, the results become inaccurate because the derivative is obtained numerically.
- Use the magnitude value instead of dB in frequency domain.

Example

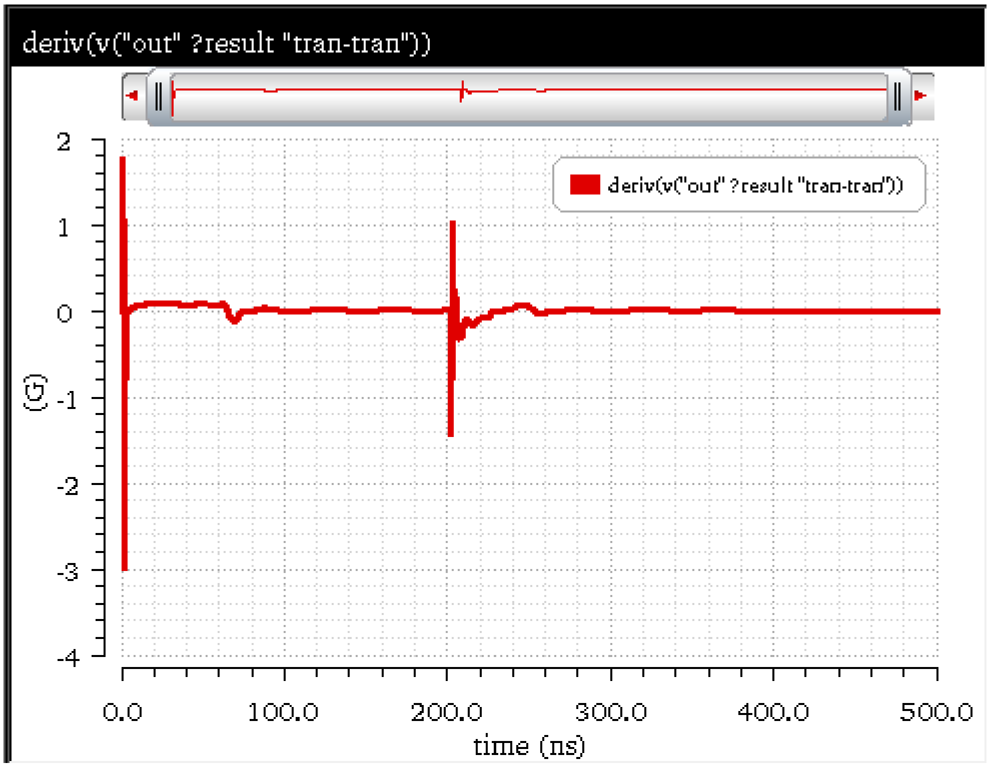
This example shows the output waveform generated when you apply the `deriv` function on the following input waveform, `(v("out" ?result "tran-tran"):`



When you apply the `deriv` function on this waveform, the following expression is created in the Buffer:

```
deriv(v("out" ?result "tran-tran")
```

Now, when you evaluate this expression, the following output waveform showing the derivative is displayed in the graph window.



Related OCEAN Function

The equivalent OCEAN command for `deriv` is:

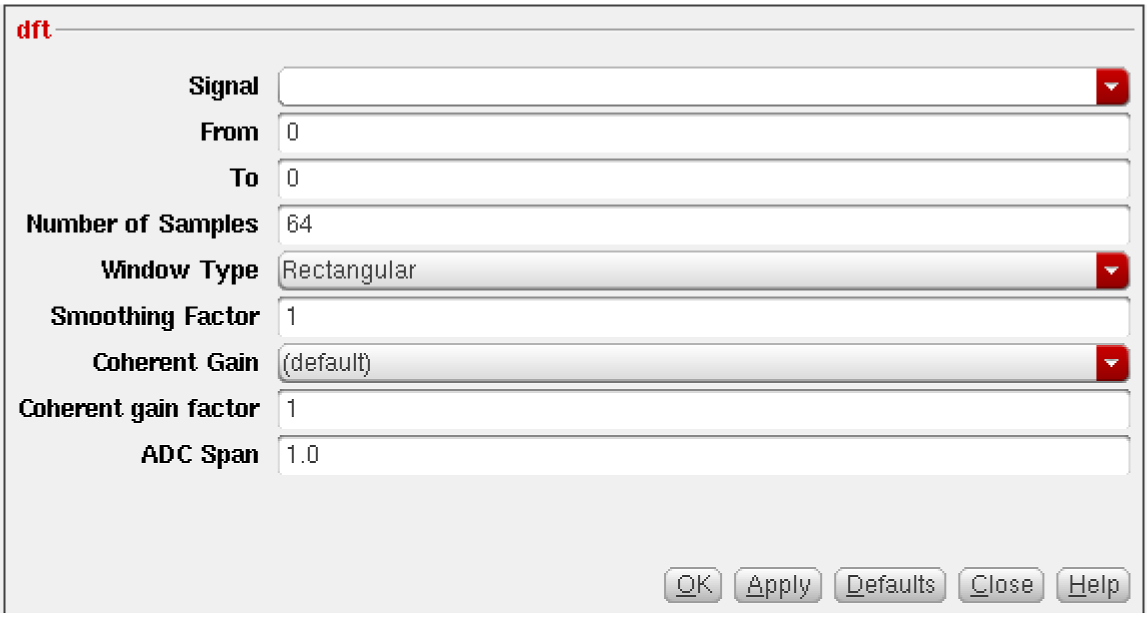
```
deriv( o_waveform )
     => o_waveform/nil
```

For more information about this OCEAN function, see <u>deriv</u> in *OCEAN Reference*.

dft

(Discrete Fourier Transform)

Computes the discrete Fourier transform of the Fourier Transform of a signal. This function is available only in the SKILL mode.



The tool which converts a temporal (time domain) description of a signal (real or complex) into one in terms of its frequency components is called the Fourier Transform. dft (Discrete Fourier Transform) is the discrete formulation of the Fourier Transform, which takes such regularly spaced data values (samples in time domain), and returns the value of the Fourier Transform for a set of values in frequency domain which are equally spaced. Most of the time, however, we work on real-valued signals only.

Consider a complex series (signal) `w(k)` with `N` samples of the form:

`w(0), w(1), w(2), ..., w(k), ..., w(N-1)`

Further, assume that the series outside the range `0`, `N-1` is extended `N`-periodic, that is, `w(k)=w(k+N)` for all `k`. The dft of this series will be denoted `W(n)`, will also have `N` samples and will be defined as:

$$W(n) = \frac{1}{N}\sum_{k-0}^{N-1} w(k)\left(e^{-2\pi ik\frac{n}{N}}\right)$$

where *n=0, ..., N-1*

- The first sample `W(0)` of the transformed series is the DC component, more commonly known as the average of the input series.
- The dft of a real series results in a symmetric series about the Nyquist frequency (described below).
- The highest positive (or negative) frequency sample is called the Nyquist frequency. This is the highest frequency component that should exist in the input series for the DFT to receive 'unpredictable' results. More specifically, if there are no frequencies above Nyquist frequency, the original signal can be exactly reconstructed from the samples. The Nyquist Theorem (or Shannon's Sampling Theorem) exactly specifies this – that for a band limited signal, you must sample at a frequency over twice the maximum frequency of the signal to reconstruct it from the samples.

While the dft transform above can be applied to any complex valued series, in practice for large series it can take considerable time to compute, the time taken being proportional to the square of the number of points (samples) in the series. A much faster algorithm has been developed by Cooley and Tukey called the FFT (Fast Fourier Transform). The only requirement of the most popular implementation of this algorithm (Radix-2 Cooley-Tukey) is that the number of points in the series be a power of `2` i.e. $N=2^n$.

Given *N* input points, the fft returns *N* frequency components, of which the first `(N/2 +1)` are valid. (The other components are mirror images and are considered invalid since the frequencies they represent do not satisfy the Nyquist Theorem above.) They start with the DC component, and are spaced apart by a frequency of `(1/(n deltaT))`. The magnitude of the complex number returned is the frequency's relative strength.

The dft function computes the discrete Fourier Transform of the buffer by fft algorithm where `deltaT=(t2-t1)/N`. The waveform is sampled at the following *N* timepoints:

*t1, t1 + deltaT, t1 + 2 * deltaT, º, t1 + (N - 1) * deltaT*

The output of `dft()` is a frequency waveform, `W(f)`, which has `(N/2+1)` complex values: the dc term, the fundamental, and `(N/2-1)` harmonics.

Note: The last time point, `(t1+(N - 1)*deltaT)`, is `(t2-deltaT)` rather than `t2`. The `dft` function assumes that `w(t1)` equals `w(t2)`.

This function includes the following arguments:

- *Signal*—Name of the signal.
- *From*—Starting point of the range over which you want to compute the transform.
- *To*—Ending point of the range over which you want to compute the transform.
  Be sure to cover at least one complete period of your slowest frequency.

- *Number of Samples*—Number of samples you want to take in expanding the Fourier transform.
  This number should be a power of 2. If it is not, the system increases the value to the next higher power of 2. Sample at a rate that is at least twice your highest frequency component (the Nyquist rate). Pick a sampling rate high enough that closely spaced frequency components can be resolved.
- *Window Type*—Window you want to use.

  Valid values: `Blackman`, `Cosine2`, `Cosine4`, `ExtCosBell`, `HalfCycleSine`, `HalfCycleSine3`, `HalfCycleSine6`, `Hanning`, `Hamming`, `Kaiser`, `Parzen`, `Rectangular`, and `Triangular`.
  Default value: `Rectangular`.
- *Smoothing Factor*—Smoothing factor applicable to the Kaiser window only.
  The Smoothing Factor field accepts values from 0 to 15. The value 0 implies no smoothing and is equivalent to a rectangular window.
- *Coherent Gain*—Scaling parameter. A non-zero value scales the power spectral density by 1/(f_cohGain). Valid values: 0 <= f_cohGain <= 1. You can use `1` if you do not want the scaling parameter to be used. Default value: `1`.
- *Coherent gain factor*—When you do a dft, the applied *Window Type* (aside from the rectangular type) changes the signal's amplitude. Applying a coherent gain factor is a way to get consistent results regardless of the window type.
- *ADC Span*—Full-scale span ignoring any DC offsets. If this value is given, the input waveform is first divided by the given adcSpan value then discrete fourier transform is calculated.

When you run the transient analysis, keep the maximum time step small enough to represent the highest frequency component accurately. The maximum time step should be smaller than the sampling period that you use for the dft of the time domain waveform. The samples in the dft will either hit a data point (calculated exactly by the simulator) or an interpolated point between two data points.

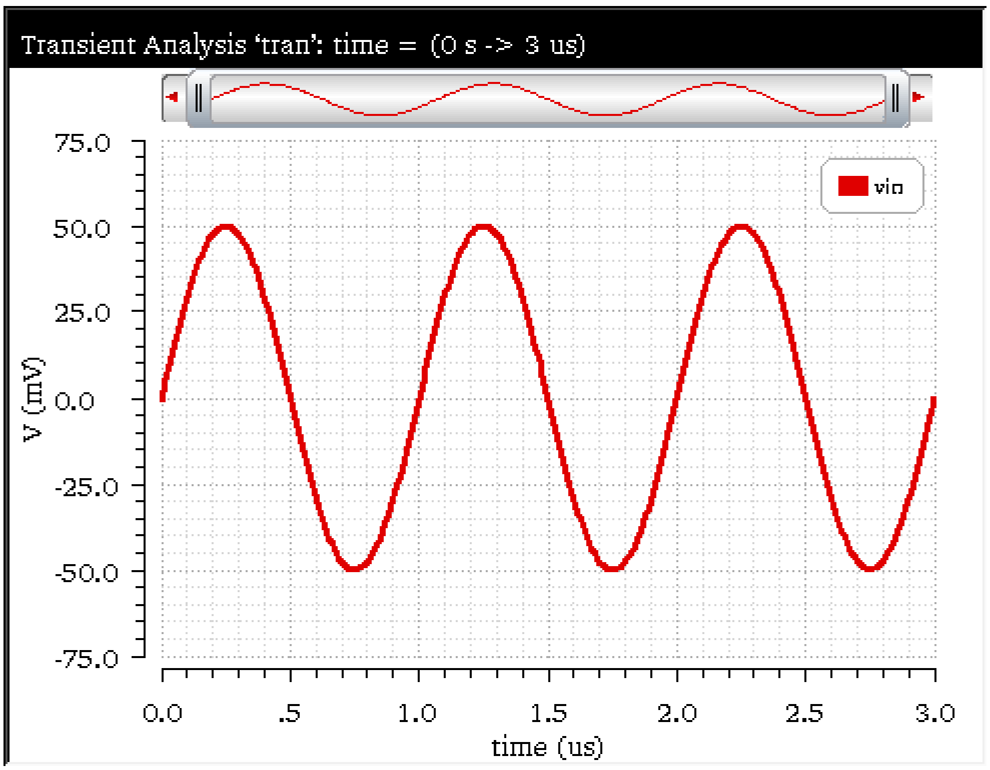
Choosing a maximum timestep during transient simulation that is smaller than the dft sampling period ensures that sampling maintains a resolution at least equal to that of the transient time-domain waveform.

The start and stop times should not coincide with the boundaries of the time-domain waveform. The boundary solutions might be imprecise and generate incorrect results if used in other calculations.

One of the uses of fast Fourier Transform windowing is to reduce discontinuities at window edges caused by having a non integral number of periods of a signal in a window. This removes the abrupt edges, making them fall off smoothly to zero, and can improve the validity of the fft components obtained. You can also use fft windowing to 'dig out' the details of signal components that are very close Gin frequency or that consist of both large and small amplitudes.

Example

This example shows the DFT output generated when you apply the `dft` function on the following input signal:



The following arguments are specified in this example:

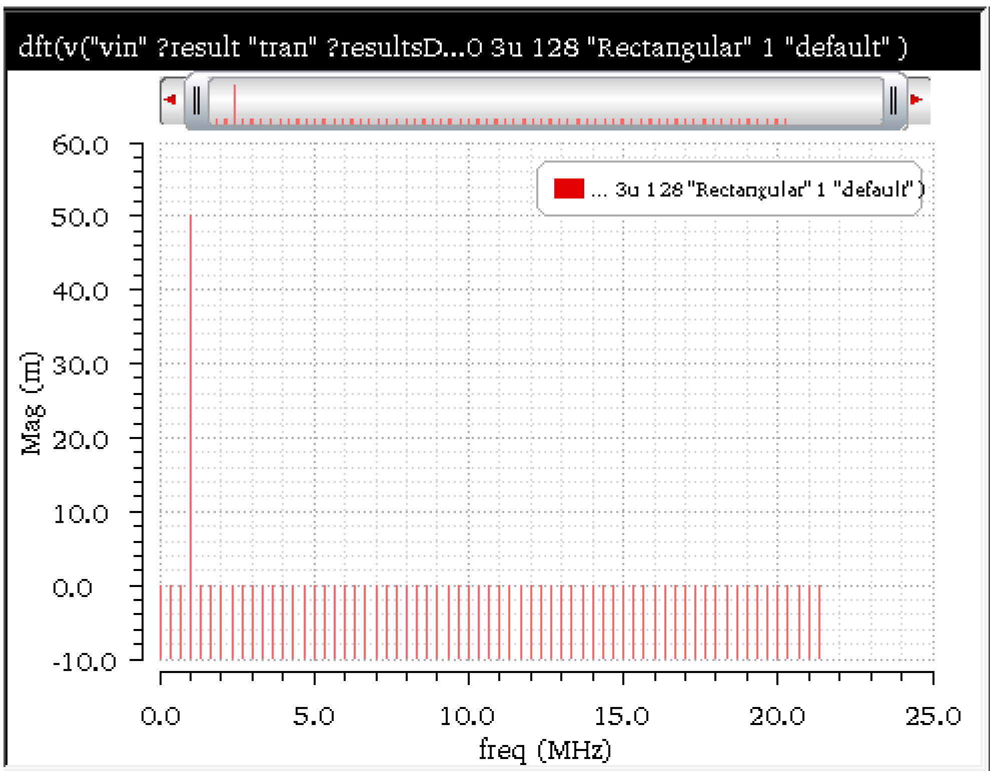
- *Signal*—`v("vin" ?result "tran")`
- *From*—`0`
- To—`3u`
- *Number of Samples*—`128`
- *Window Type*—`Rectangular`
- *Smoothing Factor*—`1`

- *Coherent Gain*—`(default)`
- *Coherent gain factor*—`1`

The following expression is created in the Buffer:

```
dft(v("vin" ?result "tran") 0 30u 128 "Rectangular" 1 "default" )
```

Now, when you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `dft` is:

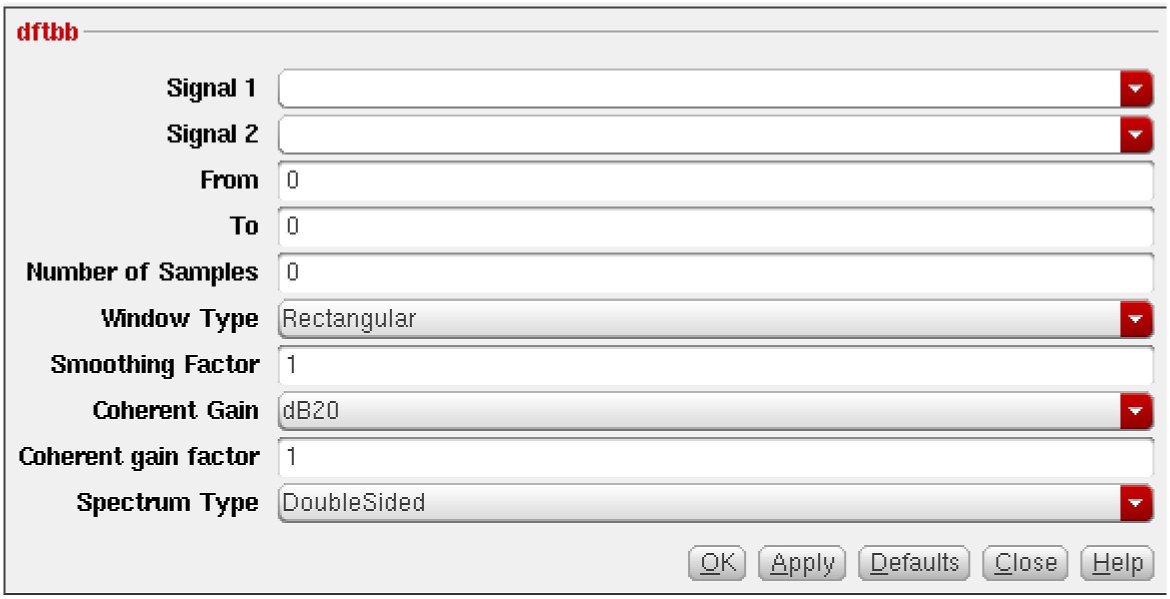
```
dft( o_waveform n_from n_to x_num [t_windowName [n_param1]] )
     => o_waveform/nil
```

For more information about this OCEAN function, see dft in *OCEAN Reference*.

dftbb

(Discrete Fourier Transform Baseband)

Computes the Discrete Fourier Transform (fast Fourier transform) of a complex signal. This function is available only in the SKILL mode.



This function includes the following arguments:

- *Signal1*—First waveform.
- *Signal2*—Second waveform.
- *From*—Starting point of the range over which you want to compute the transform.
- *To*—Ending point of the range over which you want to compute the transform.
  Be sure to cover at least one complete period of your slowest frequency.
- *Number of Samples*—Number of samples you want to take in expanding the Fourier transform.
  This number should be a power of 2. If it is not, the system increases the value to the next higher power of 2. Sample at a rate that is at least twice your highest frequency component (the Nyquist rate). Pick a sampling rate high enough that closely spaced frequency components can be resolved.
- *Window Type*—Window you want to use.
  Only the *Rectangular* window type is supported. Virtuoso Visualization and Analysis XL

issues a warning if you create an expression by hand for a different window type and switches to the default *Rectangular* window type.

- *Smoothing Factor*—Smoothing factor applicable to the Kaiser window only.
The Smoothing Factor field accepts values from 0 to 15. The value 0 implies no smoothing and is equivalent to a rectangular window.
- *Coherent Gain*—Scaling parameter. A non-zero value scales the power spectral density by 1/(f_cohGain). Valid values: 0 <= f_cohGain <= 1. You can use `1` if you do not want the scaling parameter to be used. Default value: `1`.
- *Coherent gain factor*
When you do a dftbb, the applied *Window Type* (aside from the rectangular type) changes the signal's amplitude. Applying a coherent gain factor is a way to get consistent results regardless of the window type.
- *Spectrum Type*—Specifies the type of spectrum that can be either singleSided or doubleSided. When *Spectrum Type* is single-sided, the resultant waveform is only on one side of the Y-axis starting from 0 to `N-1`. When it is double- sided, the resultant waveform is symmetric to the Y-axis from `-N/2` to `N/2`.

Additional Information

Complex signal `z(t)= x(t) + j*y(t)`:

```
N-1
Z(n)= ReZ(n) +j*ImZ(n)=SUM[ z(k)*exp(-j*theta*n*k)],
k=0
```

where,

```
theta=2*Pi/N; n=0, 1, ..., N-1.
```

Both waveforms are sampled at the following `N` timepoints:

```
t1, t1 + deltaT, t1 + 2 * deltaT, ..., t1 + (N - 1) * deltaT
```

The output of dftbb`(waveform1, waveform2)` are `N` complex values.

The dftbb function is required because the dft function gives out the amplitudes `(sqrt(Re**2+Im**2))` of dfts of real signals only – not Re and Im. Therefore, you cannot replace one dft of the complex signal `z(t)= i(t) + j*q(t)` with two dfts of two real signals `i(t)` and `q(t)`:

```
N-1
I(n)= ReI(n) +j*ImI(n)= SUM[ i(k)*exp(-j*theta*n*k)],
k=0
N-1
```

```
Q(n)= ReQ(n) +j*ImQ(n)= SUM[ q(k)*exp(-j*theta*n*k)],
k=0
```

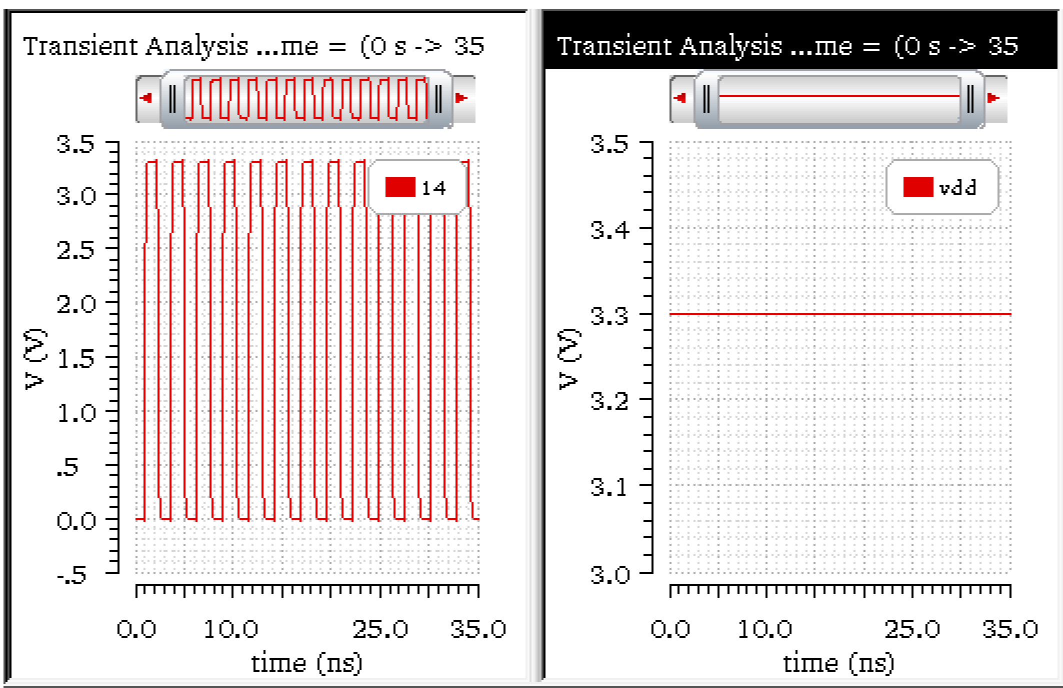
and then compute:

```
ReZ(n)= ReI(n) - ImQ(n);
ImZ(n)= ImI(n) + ReQ(n); for n=0, 1, ..., N-1.
```

The above definition is for single-sided output waveforms. This holds true for double-sided output waveforms except that the previous output waveform is translated so that `n` varies from `-N/2 to (N/2)-1.`

Example

This example displays the output DFT waveform when you apply the `dftbb` function on the following input waveforms:



The following arguments are specified in this example:

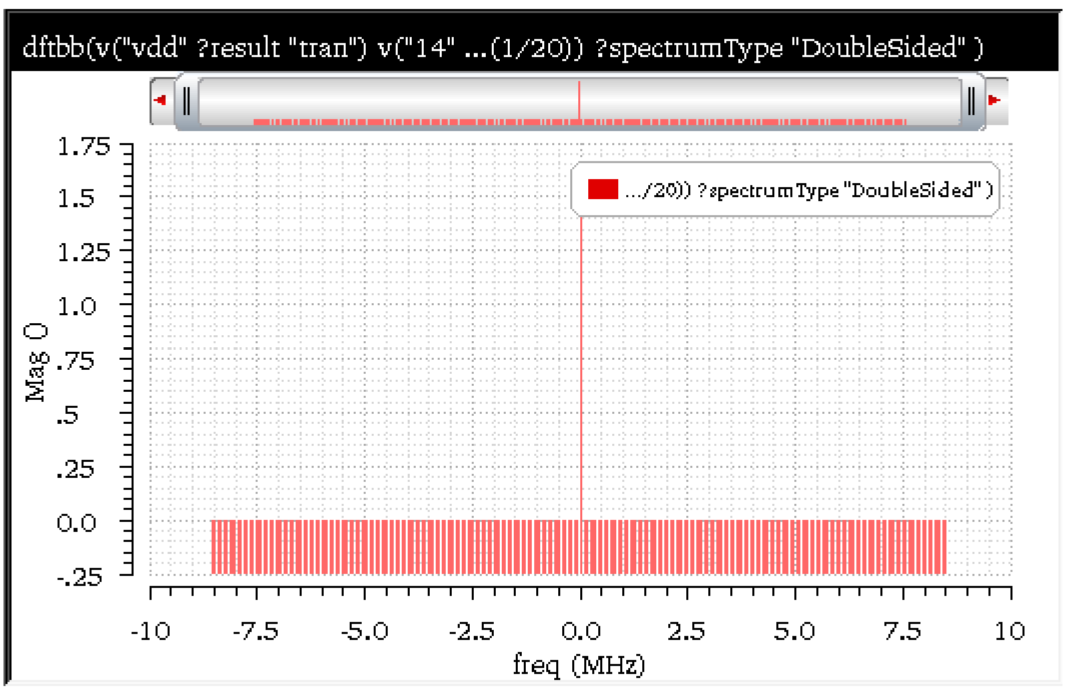
- *Signal1*—`v("14" ?result "tran")`
- *Signal2*—`v("vdd" ?result "tran")`
- *From*—`0`
- *To*—`30u`

- *Number of Samples*—`512`
- *Window Type*—`Rectangular`
- *Smoothing Factor*—`1`
- *Coherent Gain*—`dB20`
- *Coherent gain factor*—`1`
- *Spectrum Type*—`DoubleSided`

The following expression is created in the Buffer:

```
dftbb(v("14" ?result "tran" ?resultsDir "nand2_ring.raw") v("vdd"
?result "tran" ?resultsDir "nand2_ring.raw") 0 30u 512 ?windowName
"Rectangular" ?smooth 1 ?cohGain (10**(1/20)) ?spectrumType
"DoubleSided" )
```

Now, when you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `dftbb` is:

```
dftbb( o_waveform1 o_waveform2 f_timeStart f_timeEnd x_num
     ?windowName t_windowName ?smooth x_smooth ?cohGain f_cohGain
     ?spectrumType s_spectrumType)
     => o_waveformComplex/nil
```

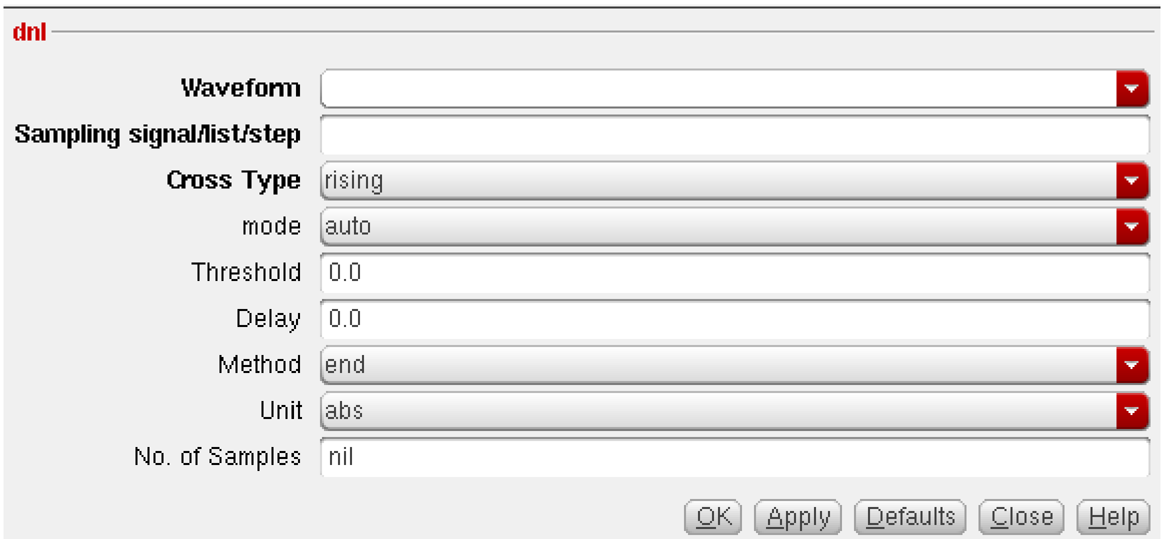
For more information about this OCEAN function, see dftbb in *OCEAN Reference*.

dnl

(Differential Non-Linearity)

Computes the differential non-linearity of a transient simple or parametric waveform. DNL can be calculated as:

$$DNL = \frac{V_{out}(i+1) - V_{out}(I)}{\text{Ideal LSB step height}} - 1$$



This function includes the following arguments:

- *Waveform*—Name of the signal.
- *Sampling signal/list/step*—Signal used to obtain the points for sampling the *Waveform* (points at which the waveform crosses the threshold while either `rising` or `falling` as specified in the *cross Type* field with the *delay* added to them), list of domain values at which the sample points are obtained from the *Waveform,* or the sampling interval.
- *Cross Type*—Specifies the points at which the curves of the waveform intersect with the threshold. While intersecting, the curve may be either rising (`rising`) or falling (`falling`).
- *mode*—Specifies the mode used to calculate the threshold value. If you want to specify the threshold value, select `user`. If you want that the Virtuoso Visualization and Analysis XL calculates the threshold value, select `auto`. The auto threshold is calculated as:

Auto Threshold Value = integral of the waveform divided by the X range.

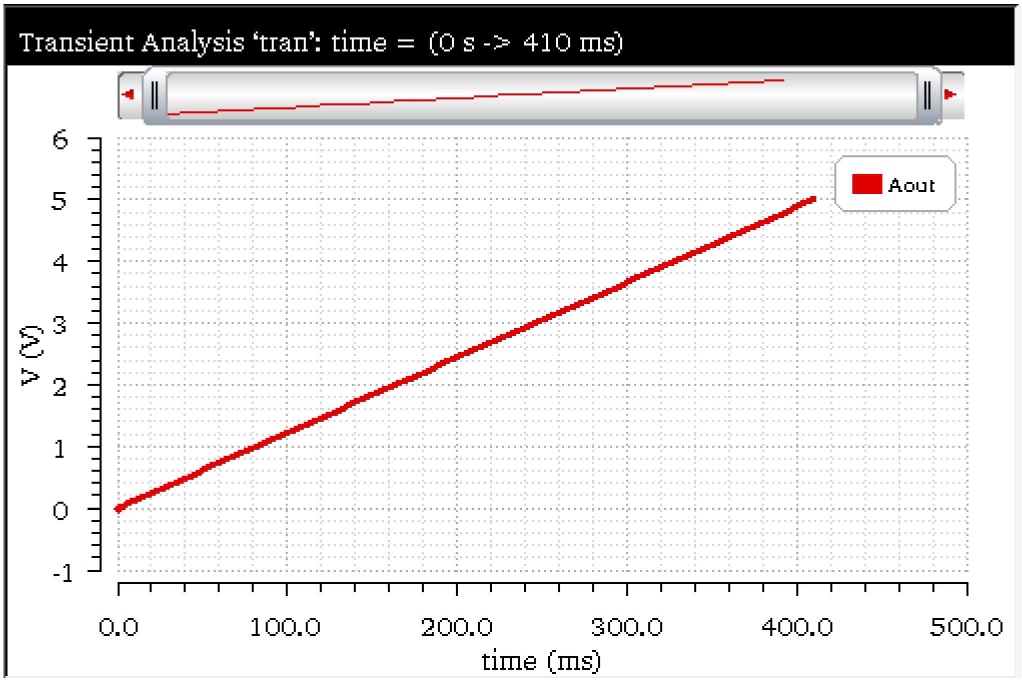
- *Threshold*—Threshold value against which the frequency is to be calculated. You need to specify the threshold value only if the *mode* is `auto`.
- *Delay*—Delay time after which the sampling begins.
- *Method*—Specifies whether the end-to-end (`end`) or straight line (`fit`) method is used
- *Unit*—Specifies whether the output waveform is to be output as an absolute value (`abs`) or multiples of least significant bit (`lsb`).
- *No. of Samples*—Specifies the samples used for calculating the non-linearity. If not specified, the samples are taken against the entire data window.

Note: For each of the three ways in which the sample points can be specified, only a few of the other optional arguments are significant,

- For *Sampling signal*, the fields *Cross Type*, *mode*, *Threshold*, *Delay*, *Method*, and *Units* are significant.
- For *list*, the fields *Method* and *Units* are significant.
- For *step*, the fields *Method*, *Units*, and *No. of samples* are significant.

Example

This example shows the output waveform generated when you apply the `dnl` function on the following input signal:



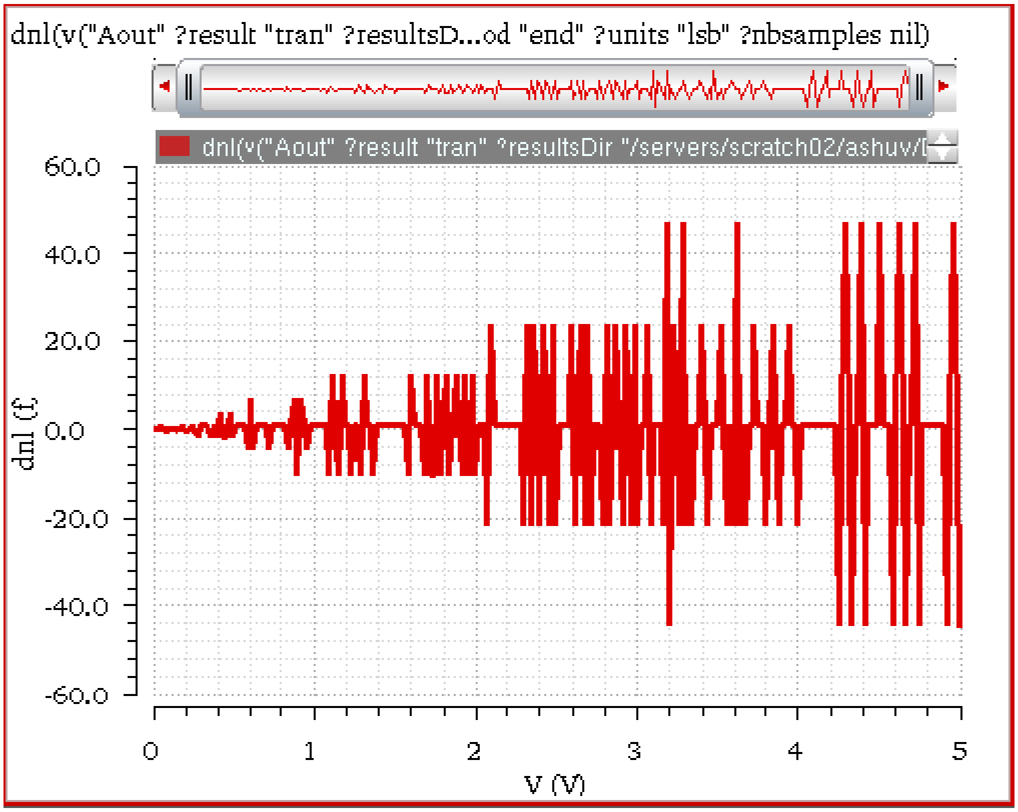
The following arguments are specified in this example:

- *Waveform*—`v("Aout" ?result "tran")`
- *Sampling signal/list/step*—`0.0016`
- *Cross Type*—`rising`
- *mode*—`auto`
- *Threshold*—`0.0`
- *Delay*—`800u`
- *Method*—`end`
- *Unit*—`lsb`
- *No. of Samples*—`nil`

The following expression is created in the Buffer with the specified arguments:

```
dnl(v("Aout" ?result "tran" ?resultsDir psf) 0.0016  ?mode "auto"
?crossType "rising" ?delay 800u ?method "end" ?units "lsb" ?nbsamples
nil)
```

Now, when you evaluate this expression, the following output waveform is displayed in the graph window. Notice that the X-axis label of this waveform is same as that of the Y-axis label of the input waveform.



Related OCEAN Function

The equivalent OCEAN command for `dnl` is:

```
dnl( o_dacSignal o_sample|o_pointList|n_interval
     [?mode t_mode] [?threshold n_threshold] [?crossType t_crossType]
     [?delay f_delay] [?method t_method][?units x_units]
     [?nbsamples n_nbsamples] )
     => n_dnl/nil
```

For more information about this OCEAN function, see dnl in *OCEAN Reference*.

dutycycle

Calculates the ratio of the time for which the signal remains high to the period of the signal. This function should be used for the periodic signals only.

This function includes the following arguments:

In the SKILL mode,

- *Waveform*—Name of the signal, expression, or family of waveforms.
- *mode*—Specifies the mode used to calculate the threshold value. If you want to specify the threshold value, select `user`. If you want that the Virtuoso Visualization and Analysis XL calculates the threshold value, select `auto`. The auto threshold is calculated as:

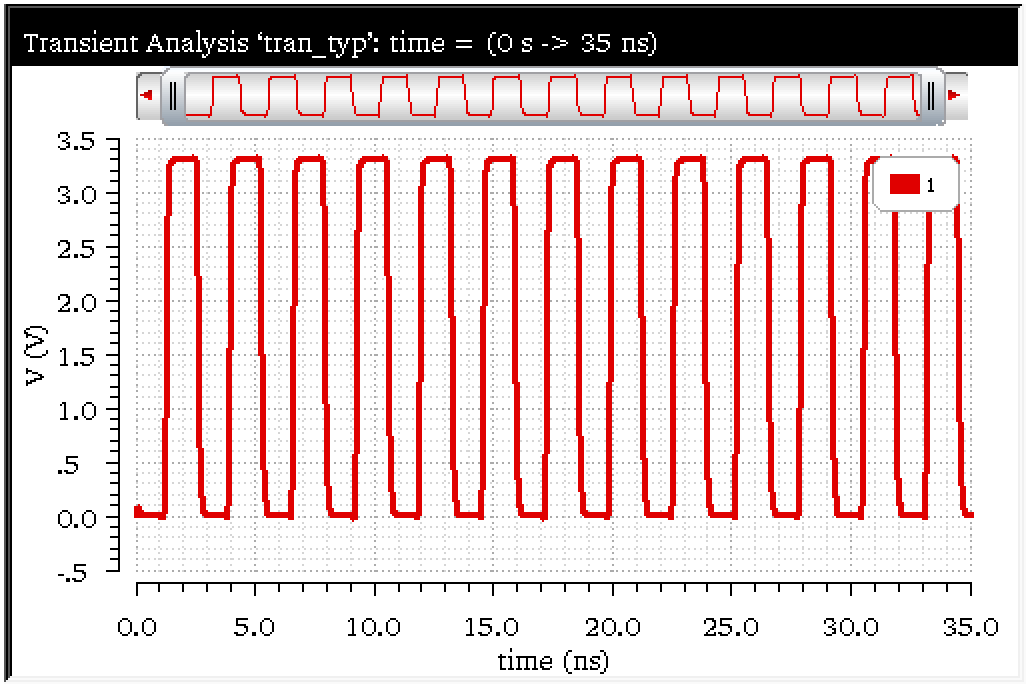
  Auto Threshold Value = integral of the waveform divided by the X range.
- *threshold*—Threshold value. You need to specify the threshold value only if the *mode* is `user`.
- *Plot/print vs.*— Specifies whether X axis of the output waveform is *time* (or another X-axis parameter for non-transient data) or *cycle*.
- *Output Type*—Type of output. If set to `plot`, the output is a waveform; if set to `average`, the output is an average value. The default value is `plot`.

In the MDL mode.

- *sig*—Name of the signal.
- *theta*—Percentage that defines the logic high of the signal. A threshold value is calculated as follows:
  yThresh=theta/100*(Ymax+Ymin)
  The portion of the signal above yThresh is taken as high.

Example

This example shows the output waveform generated when you apply the `dutyCycle` function on the following input waveform:



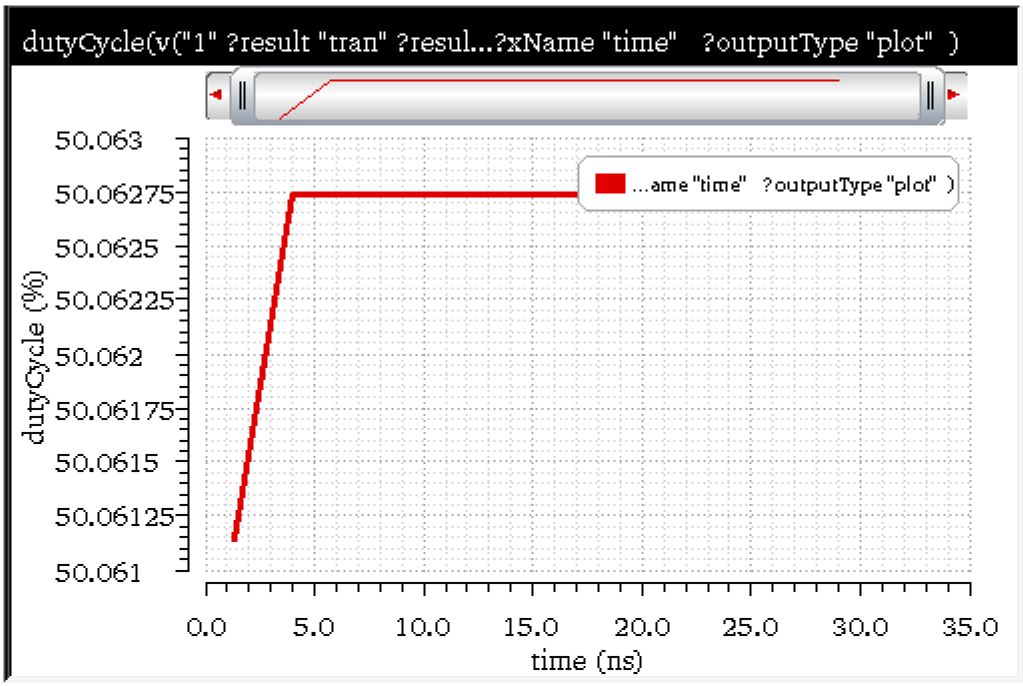
The following arguments are specified in this example:

- *Waveform*—`v("1" ?result "tran")`
- *mode*—`auto`
- *threshold*—`0.0`
- *Plot/print vs.*—`time`
- *Output Type*—`plot`

The following expression is created in the Buffer:

```
dutyCycle(v("1" ?result "tran" ?resultsDir "nand2_ring.raw") ?mode
"auto" ?xName "time"   ?outputType "plot"   )
```

Now, when you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `dutyCycle` is:

```
dutyCycle( o_waveform
     [?mode t_mode] [?threshold n_threshold]
     [?xName t_xName] [?outputType t_outputType] )
     => o_waveform/f_average/nil
```

For more information about this OCEAN function, see dutyCycle in *OCEAN Reference*.

evmQAM

(Error Vector Magnitude Quadrature Amplitude Modulation)

Processes the I and Q waveform outputs from the transient simulation run to calculate the Error Vector Magnitude (EVM) for multi-mode modulations. The function plots the I versus Q scatterplot. EVM is a useful measurement to describe the overall signal amplitude and phase modulated signal quality. It is based on a statistical error distribution normalized from an ideal digital modulation. Quadrature Amplitude Modulation (QAM) is a typical modulation scheme where EVM is useful. The EVM is calculated by detecting the I and Q signal levels corresponding to the four possible I and Q symbol combinations and calculating the difference between the actual signal level and the ideal signal level.

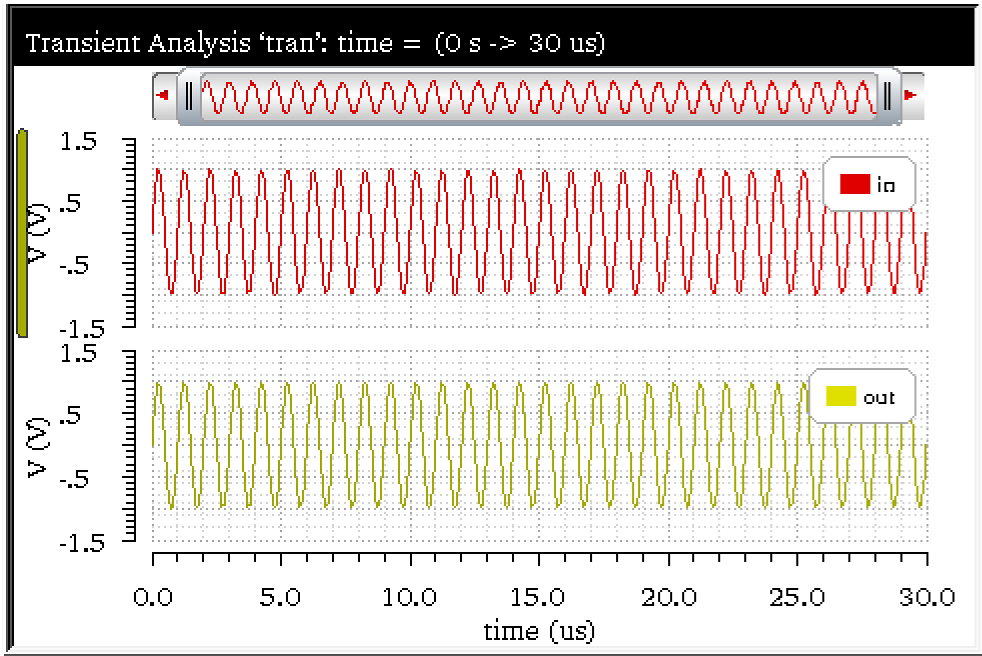
Note: This function is not supported for families of waveforms. The evmQAM function is available only in the SKILL mode.

This function includes the following arguments:

- *I-Signal*—Waveform for the I signal.
- *Q-Signal*—Waveform for the Q signal.
- *Symbol Start*—Start time for the first valid symbol.
- *Symbol period*—Period for the symbol. Each period is represented by a data rate. The data rate at the output is determined by the particular modulation scheme being used. For example, if the data rate selected is 5.5 Mbps, it corresponds to a period of 181.8 ns.
- *Modulation Level*—Modulation level. Valid values are 4, 16, 64, and 256.
- *Normalize Display*—Normalizes the scatter plot to the ideal values +1 and -1 (for example, when superimposing scatter plots from different stages in the signal flow, where the levels may be quite different but the you want to see relative degradation or improvement in the scatter).

Example

This example shows the scatter plot generated when you apply `evmQpsk` function on the following input waveforms:



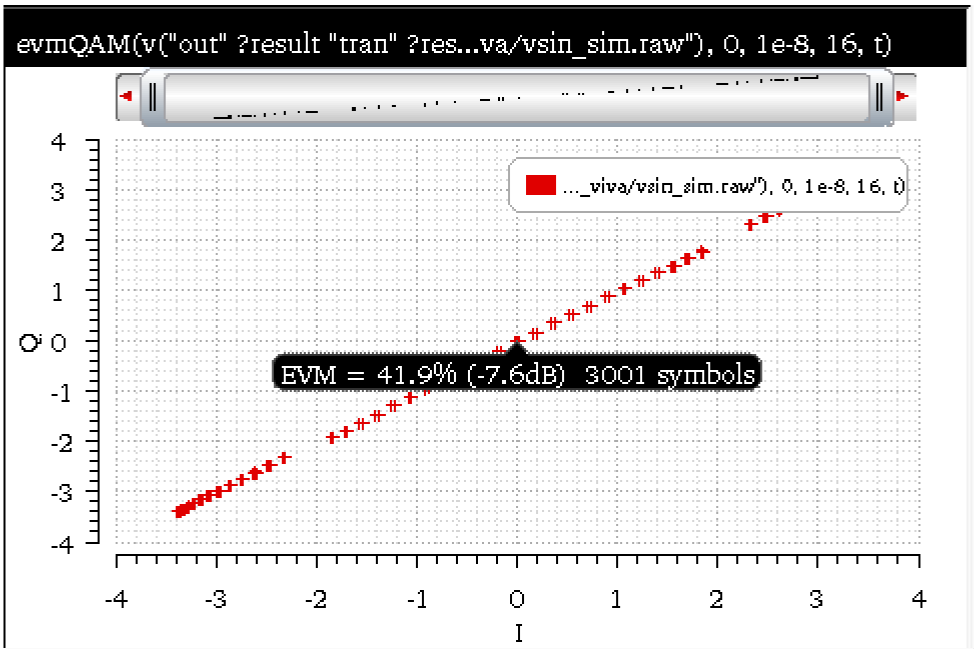
The following arguments are specified in this example:

- *I-Signal*—`v("in" result "tran")`
- *Q-Signal*—`v("out" result "tran")`
- *Symbol Start*—`0`
- *Symbol period*—`1e-8`
- *Modulation Level*—`16`
- *Normalize Display*—`on`

The following expression is created in the Buffer:

```
evmQAM(v("in" ?result "tran" ?resultsDir "./vsin_sim.raw"), v("out"
?result "tran" ?resultsDir "./vsin_sim.raw"), 0, 1e-8, 16, t)
```

Now, when you evaluate this expression, the following scatter plot is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN function for `evmQAM` is:

```
evmQAM( o_waveformI o_waveformQ n_tDelay n_samplingT x_levels g_normalize )
     => o_waveform/nil
```

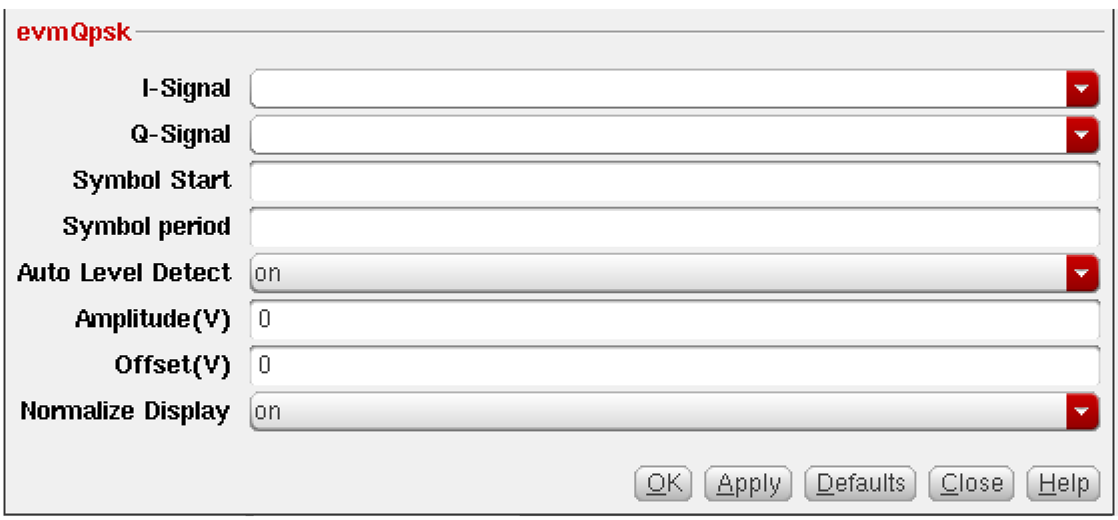
For more information about this OCEAN function, see evmQAM in *OCEAN Reference*.

evmQpsk

(Error Vector Magnitude Quadrature Phase Shift Keying)

Processes the I and Q waveform outputs from the transient simulation run to calculate the Error Vector Magnitude (EVM) and plot the I versus Q scatter plot. EVM is a useful measurement to describe the overall signal amplitude and phase modulated signal quality. It is based on a statistical error distribution normalized from an ideal digital modulation. Quadrature Phase Shift Keying (QPSK) is a typical modulation scheme where EVM is useful. The EVM is calculated by detecting the I and Q signal levels corresponding to the four possible I and Q symbol combinations and calculating the difference between the actual signal level and the ideal signal level.

Note: This function is not supported for families of waveforms. This function is available only in the SKILL mode.



This function includes the following arguments:

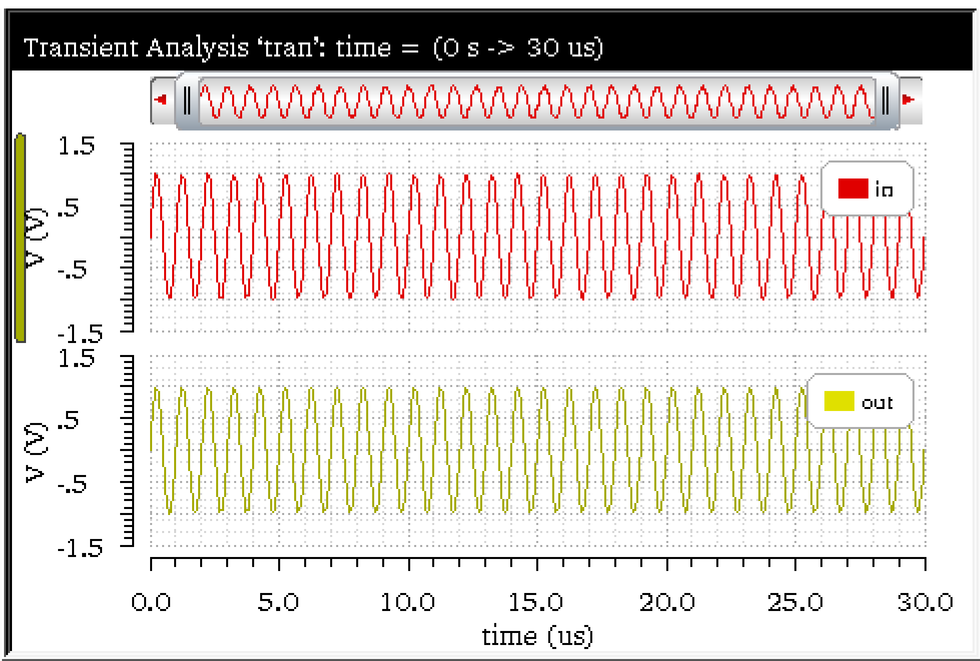
- *I-Signal*—Waveform for the I signal.
- *Q-Signal*—Waveform for the Q signal.
- *Symbol Start*—Start time for the first valid symbol.
- *Symbol period*—Period for the symbol. Each period is represented by a data rate. The data rate at the output is determined by the particular modulation scheme being used. For example, if the data rate selected is 5.5 Mbps, it corresponds to a of 181.8 ns.
- *Auto Level Detect* `on` indicates that you want the amplitude (*Amplitude*) and DC offset (*Offset*) to be calculated automatically. *Amplitude* is calculated by averaging the rectified voltage level of the signal streams and DC *Offset* by averaging the sum of an

equal number of positive and negative symbols in each signal stream. These values are used to determine the EVM value. If *Auto Level Detect* is set to `off`, you must specify values for the *Amplitude* and *Offset* fields.

- *Amplitude(V)* is the amplitude of the signal. You need to specify a value in this field only if *Auto Level Detect* is set to `off`.
- *Offset(V)* is the DC offset value. You need to specify a value in this field only if *Auto Level Detect* is set to `off`.
- *Normalize Display* normalizes the scatter plot to the ideal values +1 and -1 (for example, when superimposing scatter plots from different stages in the signal flow, where the levels may be quite different but the you want to see relative degradation or improvement in the scatter).

Example

This example shows the scatter plot generated when you apply `evmQpsk` function on the following input waveforms:



The following arguments are specified in this example:

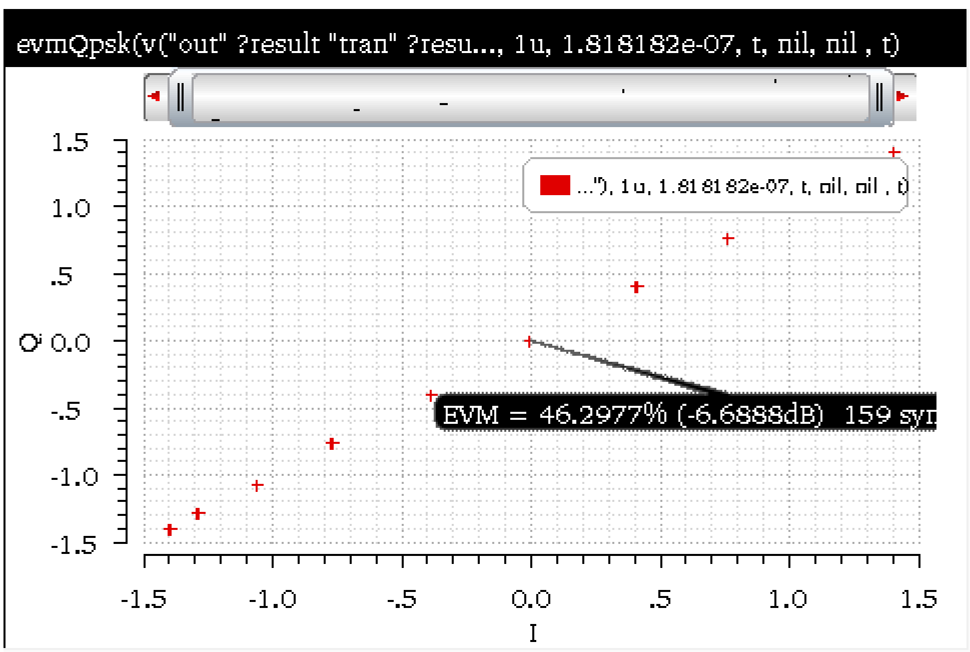
- *I-Signal*—`v("in" result "tran")`
- *Q-Signal*—`v("out" result "tran")`

- *Symbol Start*—`1u`
- *Symbol period*—`1.818182e-07`
- *Auto Level Detect*—`on`
- *Amplitude(V)*—`0`
- *Offset(V)*—`0`
- *Normalize Display*—`on`

The following expression is created in the Buffer:

```
evmQpsk(v("in" ?result "tran" ?resultsDir "./vsin_sim.raw"), v("out"
?result "tran" ?resultsDir "./vsin_sim.raw"), 1u, 1.818182e-07, t,
nil, nil , t)
```

Now, when you evaluate this expression, the following scatter plot is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN function for `evmQpsk` is:

```
evmQpsk( o_waveform1 o_waveform2 n_tDelay n_sampling
     g_autoLevelDetect n_voltage n_offset g_normalize )
     => o_waveform/nil
```

For more information about this OCEAN function, see evmQpsk in *OCEAN Reference*.

eyeDiagram

Gives an eye-diagram plot in which the waveform signal is divided into fixed time periods, which are then superimposed on each other. The result is a plot that has many overlapping lines enclosing an empty space known as the eye. The quality of the receiver circuit is characterized by the dimension of the eye. An open eye means that the detector can distinguish between `1's` and `0's` in its input, while a closed eye means that a detector placed on `Vout` is likely to give errors for certain input bit sequences.

This function is available only in the SKILL mode.

This function includes the following arguments:

- *Signal*—Name of the signal.
- *Start Time*—X-axis value from where the eye-diagram plot is to begin.
- *Stop Time*—X-axis value where the eye-diagram plot is to end.
- *Period*—Time period for the eye diagram.
- *Adv. Options*—Specifies whether the vertical (`Max Vertical Opening`) or horizontal opening (`Max Horizontal Opening`) of the eye is to be calculated.

Additional Information

Calculating Horizontal and Vertical Eye Width:

The waveform is folded on the X-axis between the start time (`n_start`) and stop time (`n_stop`) by the length `n_period`.

The function performs the following steps to calculate the horizontal eye opening:

- Calculates all the X points are where the folded waveform intersects the line `y = yMid` where, `yMid = ymax(o_waveform)+ min(o_waveform))/2`

- From these calculated X points, returns the two consecutive X points, which have the maximum distance between them, as the horizontal eye opening.

  `X[k]` and `X[k-1]` for which the (`X[k]-X[k-1]`) value is maximum.

The function performs the following steps to calculate vertical eye width:

- Calculates the horizontal eye width to find the consecutive X points, `X[k]` and `X[k-1]` having maximum distance between them.
- Calculates all Y points where the folded waveform intersects the following line:

  `x = (X[k]-X[k-1])/2`

- From these calculated Y points, returns two consecutive Y points, which have the maximum distance between them, as the vertical eye opening.

  `Y[k]` and `Y[k-1]` for which (`Y[k]-Y[k-1]`) is maximum.

Assumptions

The following assumptions have been made while calculating the advance option values:

- The opening of an eye approximately lies on following:

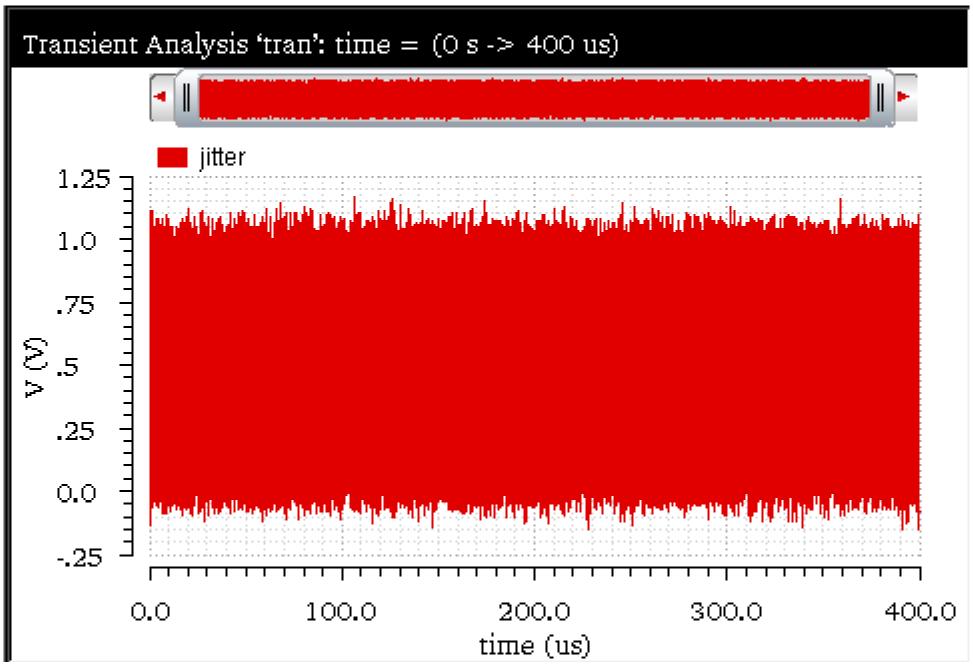
  `(ymax(o_waveform)+ymin(o_waveform))/2`

- Only one eye opening exists in the area mentioned above in which the waveform is folded.

For more information about eye diagram, see the <u>Eye Diagram</u> assistant.

Example

This example shows the eye diagram plot generated when you apply the `eyeDiagram` function on the following input signal:



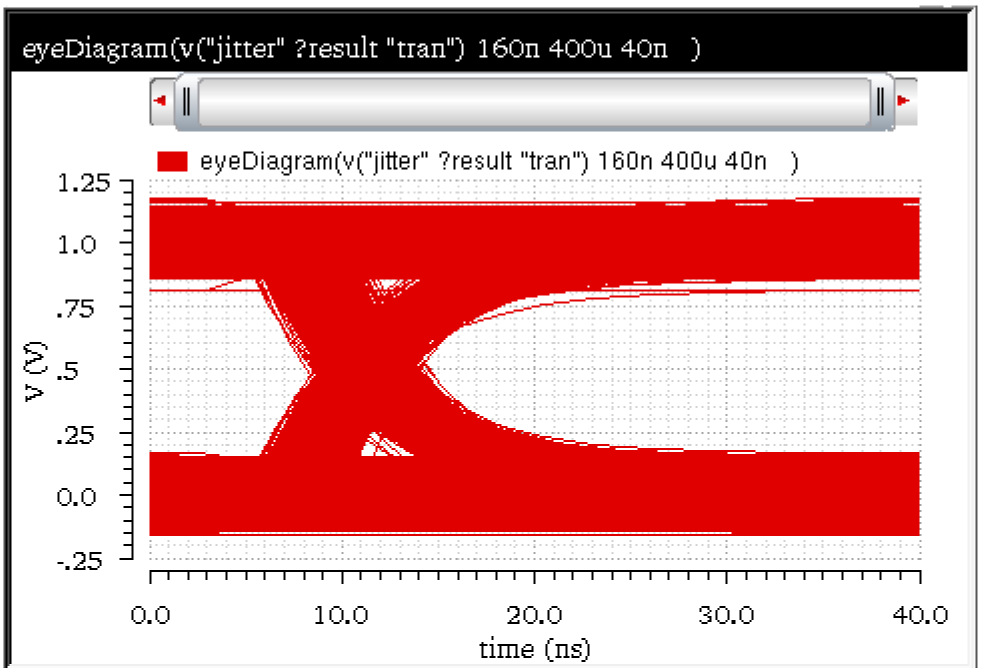
The following arguments are specified in this example:

- *Signal*—`v("jitter" ?result "tran")`
- *Start Time*—`160n`
- *Stop Time*—`400u`
- *Period*—`40n`
- *Adv. Options*—`None`

The following expression is created in the Buffer:

```
eyeDiagram(v("jitter" ?result "tran") 160n 400u 40n )
```

Now, when you evaluate this expression, the following eye diagram plot is displayed in the graph window:



Related OCEAN Function

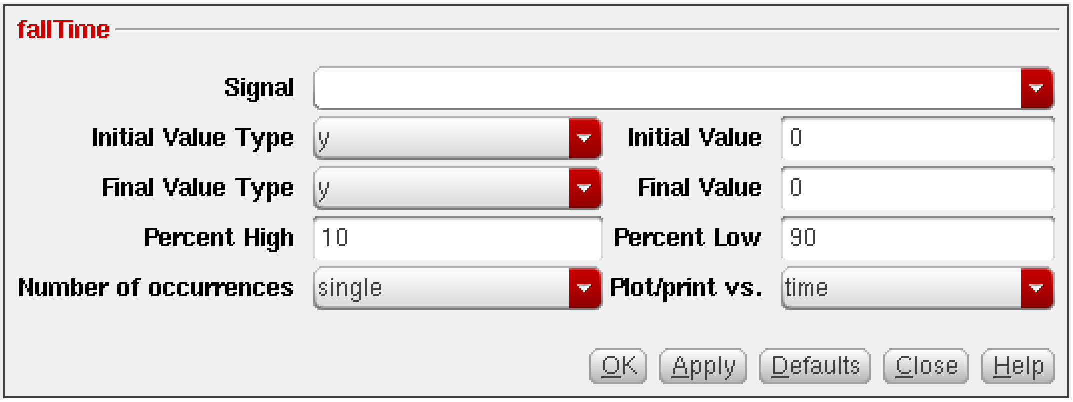
The equivalent OCEAN command for `eyeDiagram` is:

```
eyeDiagram ( o_waveform n_start n_stop n_period ?advOptions t_advOptions )
     => o_waveform/nil
```

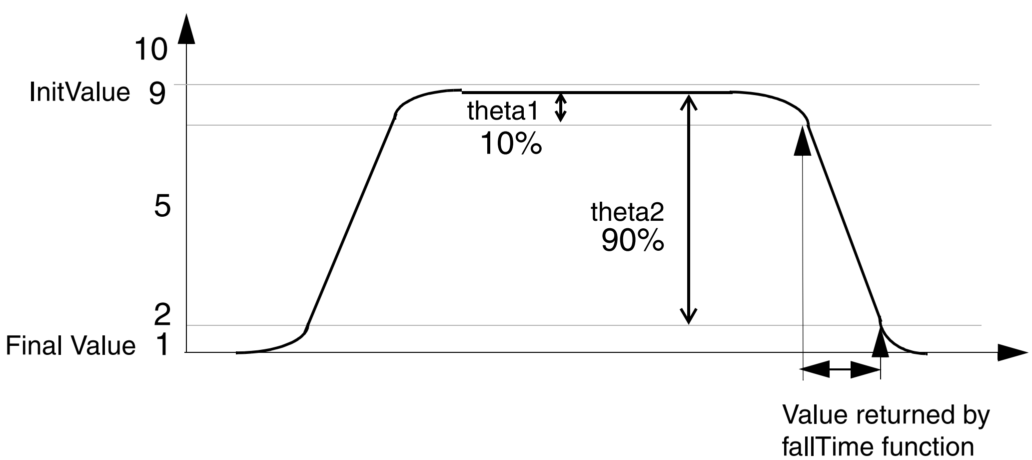
For more information about this OCEAN function, see eyeDiagram in *OCEAN Reference*.

fallTime

Returns the fall time measured between *`theta1`* (percent high) to *`theta2`* (percent low) of the difference between the initial value and the final value.



The `fallTime` function can also be used to compute the rise time if *`initVal`* is lower than *`finalVal`*.



This function includes the following fields:

- *Signal*—Name of the waveform.
- *Initial Value Type*—Initial value type used to start the computation. The options are y at x or y.
- *Initial Value*—Initial value at which the computation is to be started.

- *Final Value Type*—Final value type used to end the computation. The options are y at x or y.
- *Final Value*—Final value at which the computation is to be ended.
- *Percent High*—Percentage high difference between initial and final values.
- *Percent Low*—Percentage low difference between initial and final values.
- *Number of occurrences*—Number of occurrences of falls to be calculated. It can be single or multiple.
- *Plot/print vs.*—All falltime values to be calculated per cycle or time.

Additional Information

Consider the following equations:

```
Val1 = theta1 / 100.0 * diff + initVal

Val2 = theta2 / 100.0 * diff + initVal
```

The following table shows how the *fallTime* function works when you apply the above equations:

Function *FallTime(w initVal nil finalVal nil theta1 theta2)*	**initVal**	**finalVal**	**theta1**	**theta2**	**Val1**	**Val2**	**Output**
Case 1	0	1	10	80	0.1	0.8	1.4n (time taken to rise from 200.0p ns to 1.6ns) See Figure for Case 1 on page 679.
Case 2	0	1	80	10	0.1	0.8	0.7n (time taken to fall from 4.2ns to 4.9ns) See Figure for Case 2 on page 680.
Case 3	1	0	10	80	0.9	0.2	0.7n (time taken to fall from 4.1ns to 4.8ns) See Figure for Case 3 on page 681.
Case 4	1	0	80	10	0.2	0.9	1.4n (time taken to rise from 400.0p s to 1.8ns) See Figure for Case 4 on page 682

Figure for Case 1

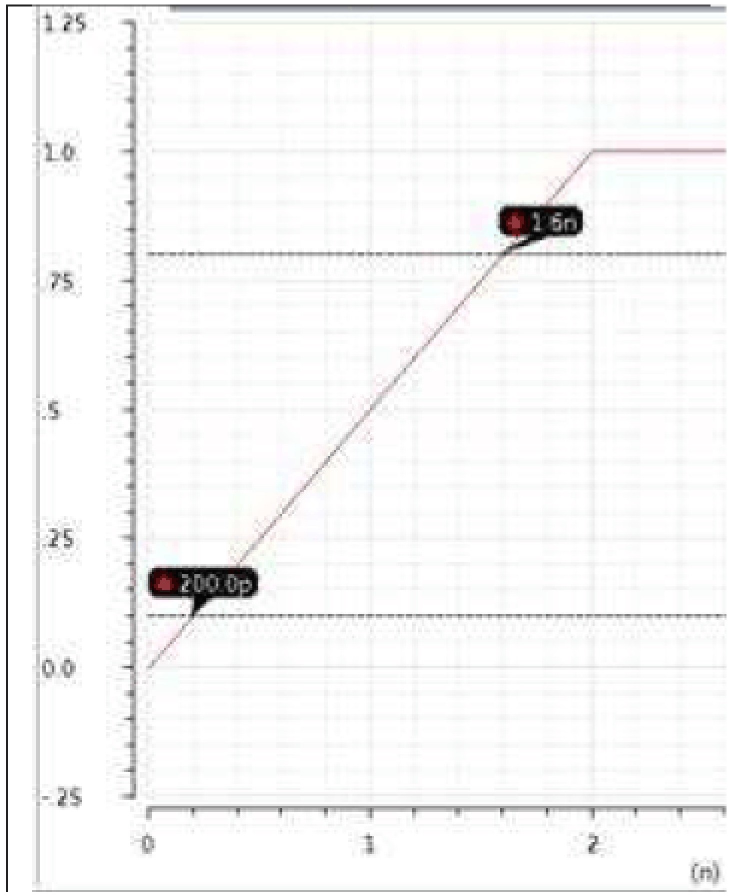


Figure for Case 2

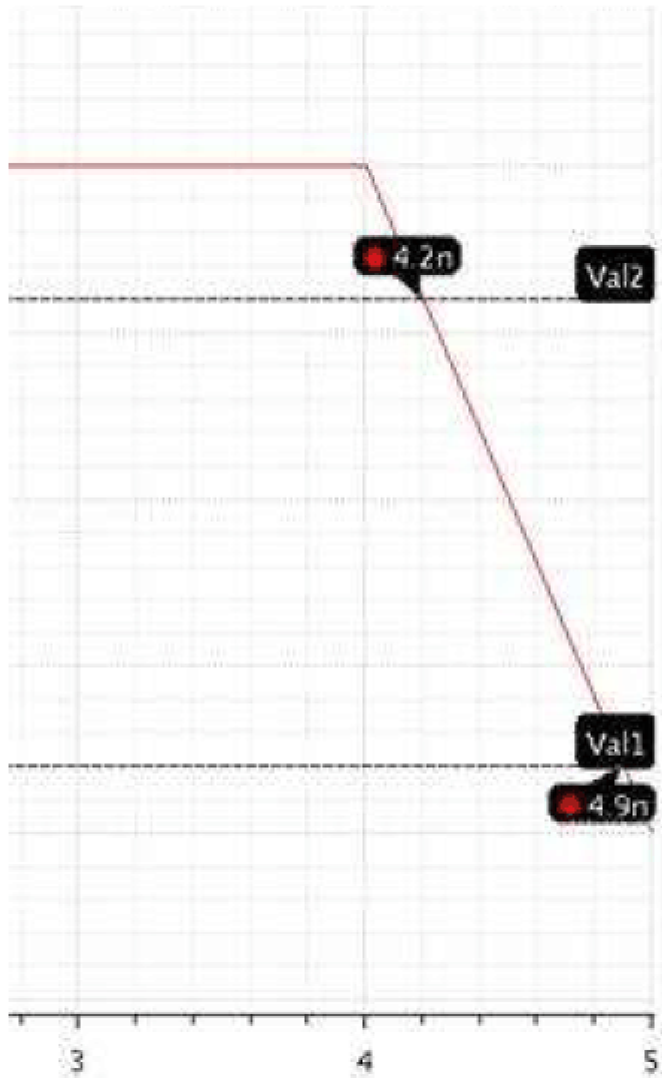


Figure for Case 3

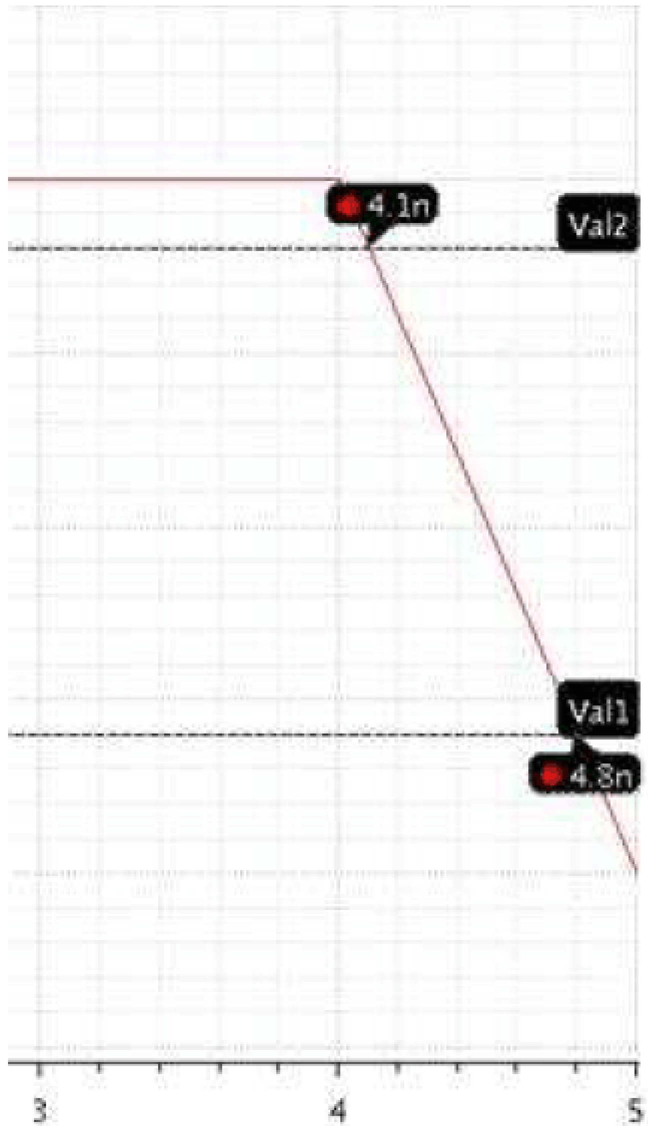
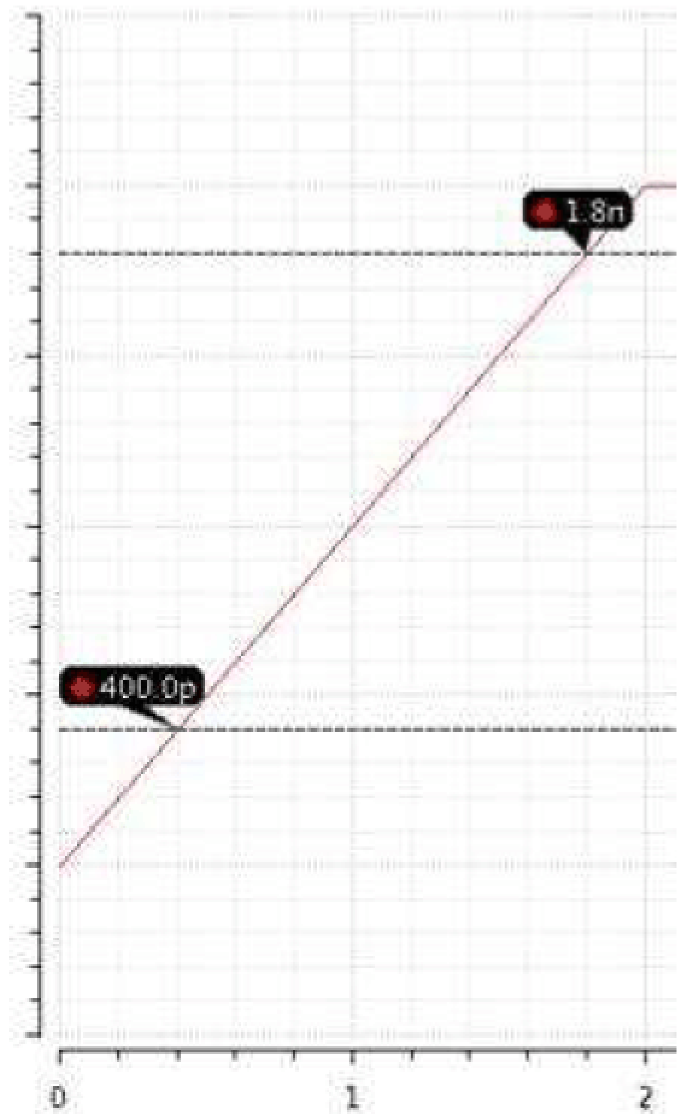
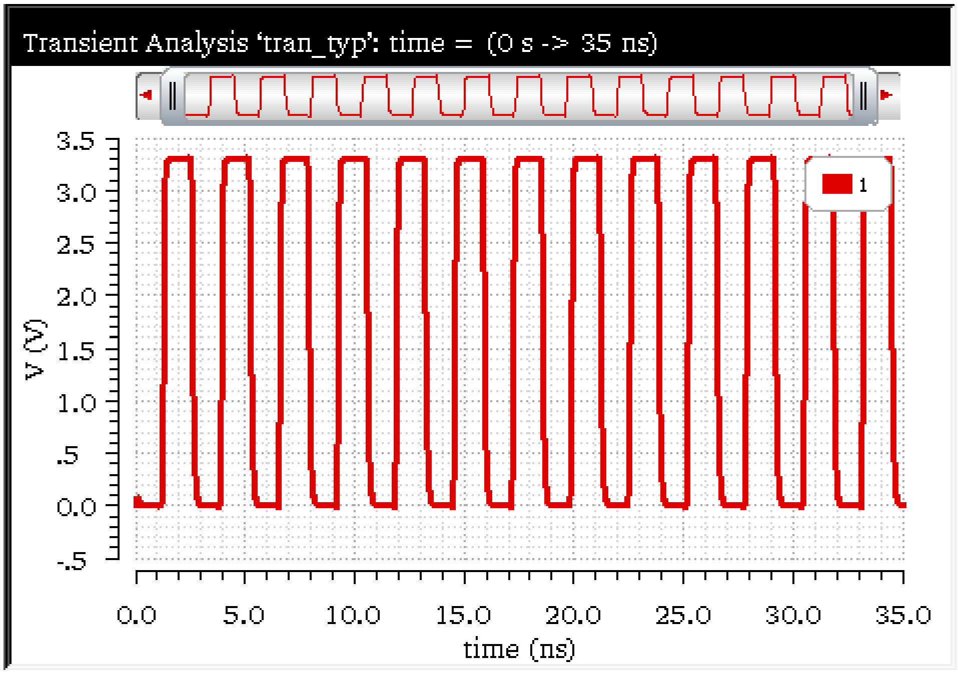


Figure for Case 4



Example

This example shows the output waveform genearted when you apply the `fallTime` function on the following input signal:



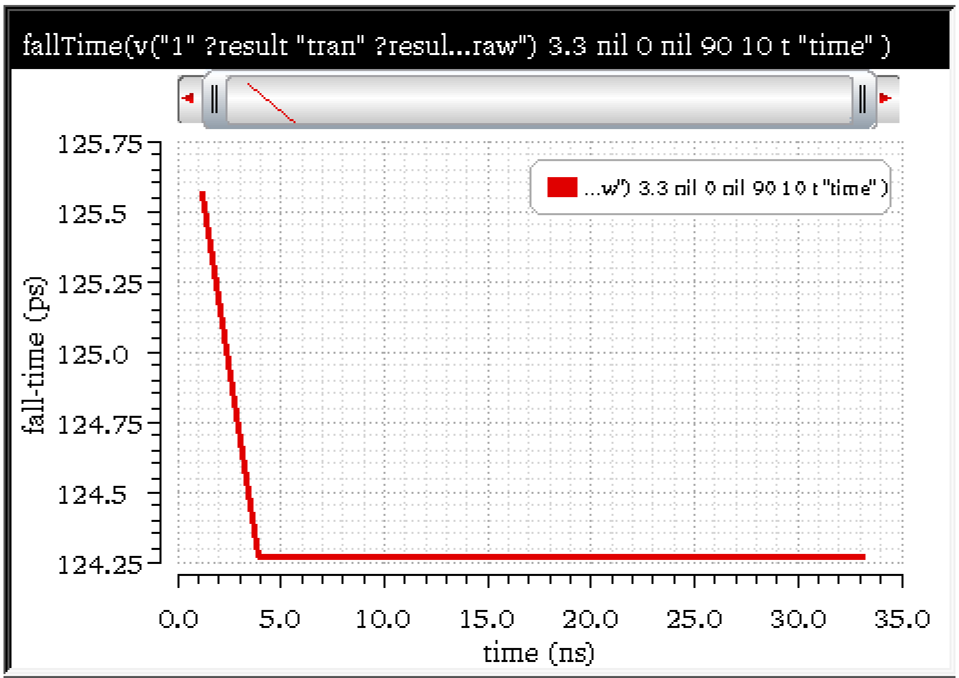
The following arguments are specified in this example:

- *Signal*—`v("1" ?result "tran")`
- *Initial Value Type*—`y`
- *Initial Value*—`3.3`
- *Final Value Type*—`y`
- *Final Value*—`0.0`
- *Percent High*—`90`
- *Percent Low*—`10`
- *Number of occurrences*—`multiple`
- *Print/plot vs.*—`time`

The following expression is created in the Buffer:

```
fallTime(v("1" ?result "tran" ?resultsDir "./nand2_ring.raw") 3.3 nil
0 nil 90 10 t "time" )
```

Now, when you evaluate this expression, the following output waveform showing the falltime is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `fallTime` is:

```
fallTime( o_waveform n_initVal g_initType
     n_finalVal g_finalType n_theta1 n_theta2
     [g_multiple [s_Xname][g_histoDisplay][x_noOfHistoBins] ] )
     => o_waveform/n_value/nil
```

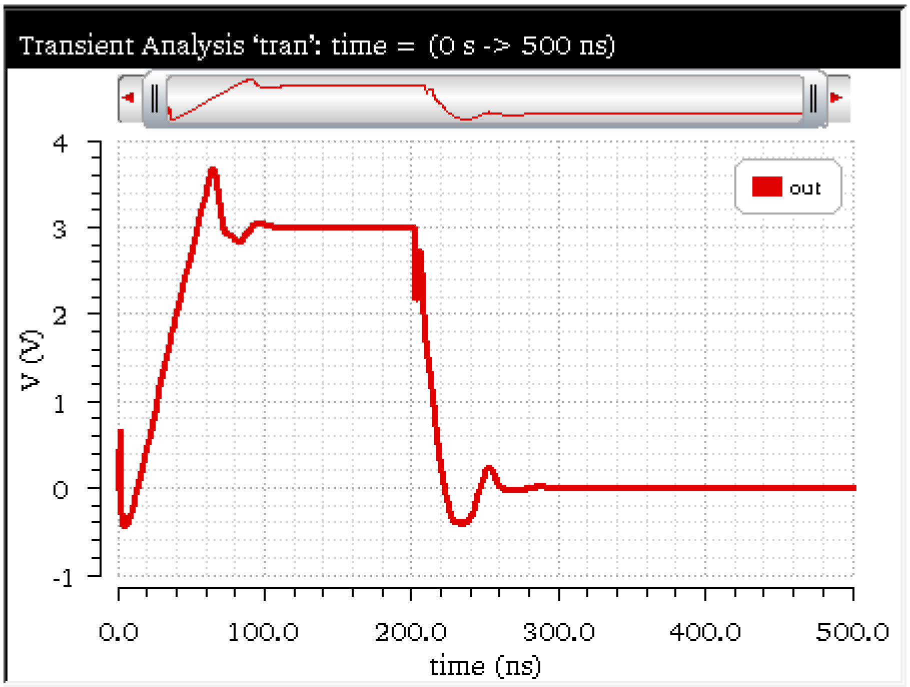
For more information about this OCEAN function, see fallTime in *OCEAN Reference*.

flip

Returns a reversed version of a signal (rotates the signal along the Y-axis).

Example

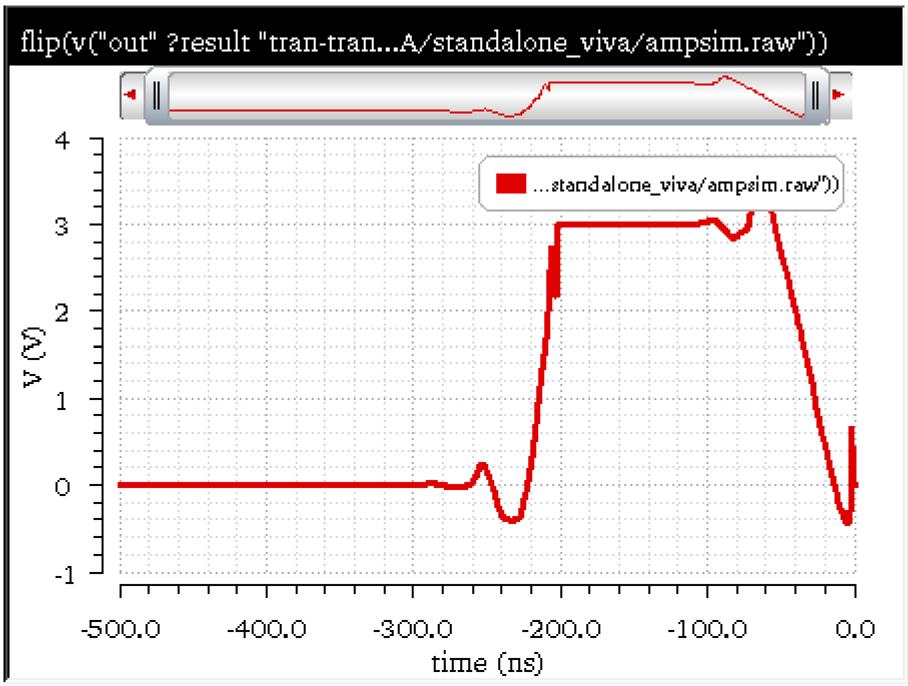
This example displays the output waveform generated when you apply the `flip` function on the following input signal:



When you apply the `flip` function on this signal, the following expression is created in the Buffer:

```
flip(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

When you evaluate this expression, the following output waveform, which is the inverse of the input signal, is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for the `flip` function is:

```
flip( o_waveform )
     => o_waveform/nil
```

For more information about the OCEAN function, see flip in *OCEAN Reference*.

fourEval

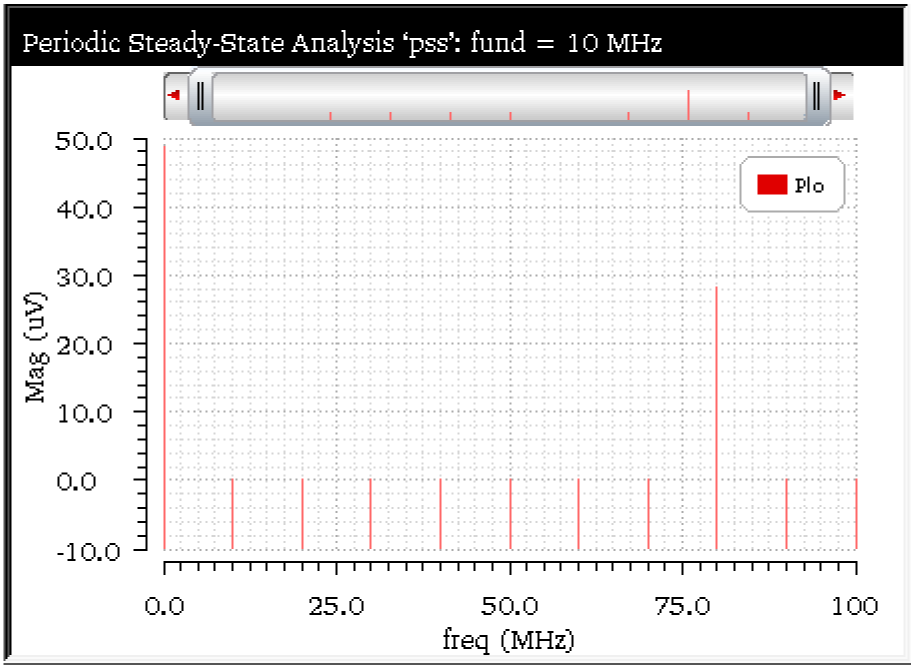
Evaluates the Fourier series represented by the buffer expression. This function is an inverse Fourier transformation and thus the inverse of the dft (discrete fourier transform) function. It transforms the buffer expression from the frequency domain to the time domain.This function is available only in the SKILL mode.

This function includes the following arguments:

- *Signal*—Name of the signal.
- *From*—Time at which you want to begin evaluating the series.
- *To*—Time till which you want to evaluate the series.
- *By*—Increment.
- *Baseband*—Evaluates the baseband version of the inverse of the `dft` function by converting the unsymmetrical spectrum to a symmetrical one (when set to `on`).

Example

This example calculates the Fourier series and shows the output plot generated when you apply the `fourEval` function on the following spectrum signal, `v(Plo)`:
;:



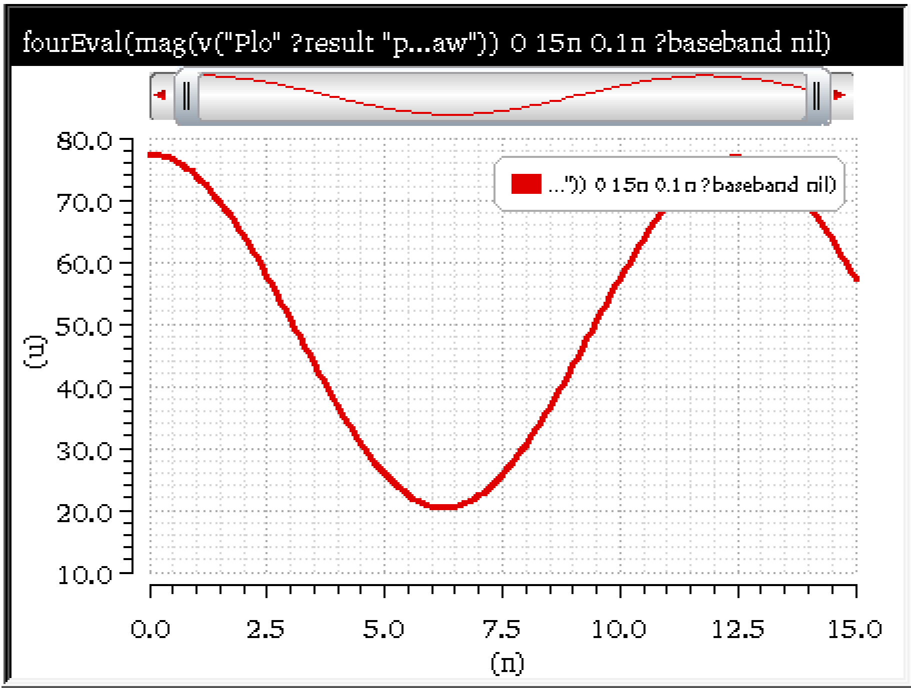
The following argument values are specified in this example:

- *Signal*—`v("Plo" ?result "pss_fd" ?resultsDir "./pss_ne600.raw")`
- *From*—`0n`
- *To*—`15n`
- *By*—`0.1n`
- *Baseband*—`off`

The following expression is created in the Buffer:

```
fourEval(mag(v("Plo" ?result "pss_fd" ?resultsDir "./
pss_ne600.raw")) 0 15n 0.1n ?baseband nil)
```

When you evaluate this expression, the output waveform for the Fourier series is displayed in the graph window (as shown in the figure below):



Related OCEAN Function

The equivalent OCEAN command for `fourEval` function is:

```
fourEval( o_waveform n_from n_to n_by [?g_baseBand] )
     => o_waveform/nil
```

For more information about the OCEAN function, see fourEval in *OCEAN Reference*.

freq

In the SKILL mode, `freq` returns a waveform representing the frequency of a signal versus time or cycle.

- *Signal*—Name of the input signal.
- *Edge Type*—Direction of the crossing event.
  `rising` directs the function to look for crossings where the Y value is increasing
  `falling` directs the function to look for crossings where the Y value is decreasing
- *Plot vs.*—Specifies whether you want to retrieve frequency against *time* (or another X-axis parameter for non-transient data) or *cycle*. Cycle numbers refer to the n'th occurrence of the delay event in the input waveform.
- *Threshold Mode*—Specifies the mode used to calculate the threshold value. If you want to specify the threshold value, select `user`. If you want that the Virtuoso Visualization and Analysis XL calculates the threshold value, select `auto`. The *auto* threshold is calculated as:

  Auto Threshold Value = integral of the waveform divided by the X range.
- *User Threshold Value*—Threshold Y-axis value to be crossed. In SKILL mode, this field is available only if the *Threshold Mode* is *auto*.

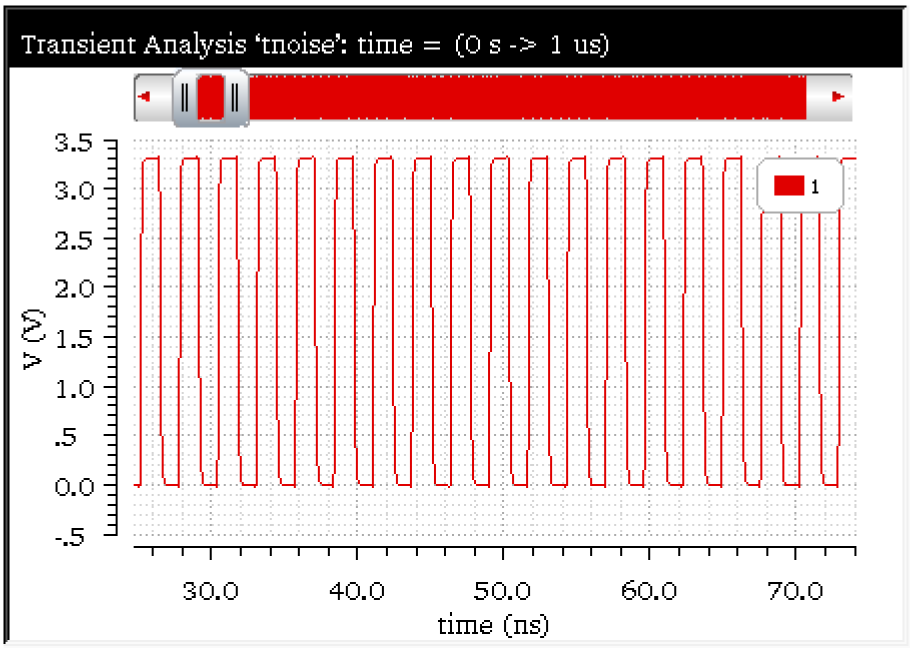
In the MDL mode, `freq` returns a waveform representing the frequency of a signal versus time.

- *sig* is the signal.
- *thresh* is the threshold Y-axis value to be crossed.
- *dir* is the direction of the crossing event. rising directs the function to look for crossings where the Y value is increasing and falling for crossings where the Y value is decreasing.

- the *(thresh)* is the threshold Y-axis value to be crossed. In SKILL mode this field is used only if the *Threshold Mode* is *auto*.

Example

This example shows the output generated when you apply the `freq` function on the input signal plotted in the graph below:



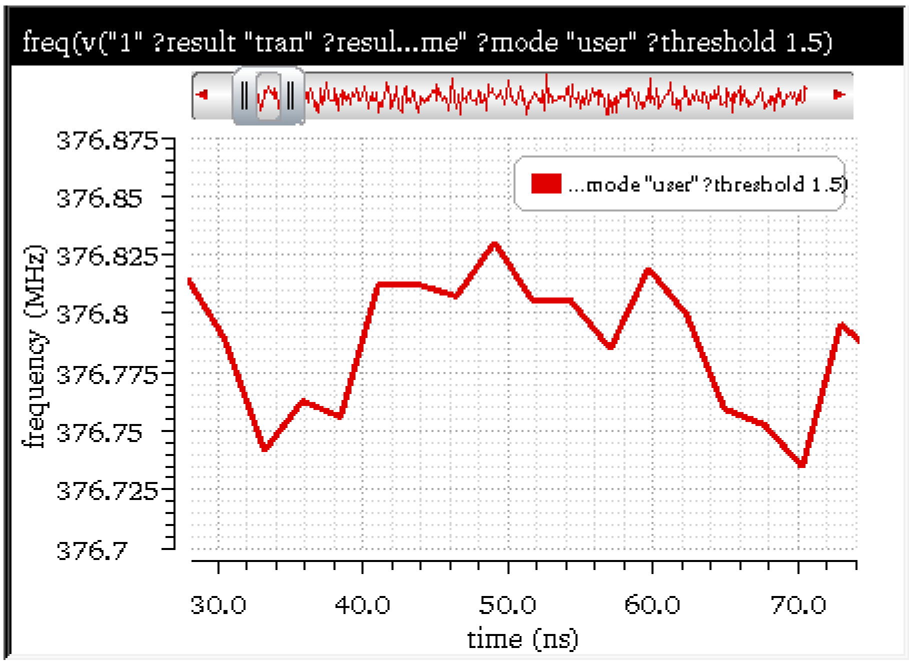
The following arguments are specified in this example:

- *Signal*—`v("1" ?results "tran")`
- *Edge Type*—`rising`
- *Plot Vs.*—`time`
- *Threshold Mode*—`user`
- *User Threshold Value*—`1.5`

The following expression is created in the Buffer:

```
freq("v("1" ?results "tran" ?resultsDir "nand2_ring_tnoise.raw")"
"rising" ?xName "time" ?mode "user" ?threshold 1.5)
```

When you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN function for `freq` is:

```
freq( o_waveform t_crossType
     [?threshold n_threshold] [?mode t_mode]
     [?xName xName][g_histoDisplay][x_noOfHistoBins] )
     => o_outputWave/nil
```

For more information about this OCEAN function, see freq in *OCEAN Reference*.

freq_jitter

Returns a waveform representing the deviation from the average frequency.

This function includes the following arguments:

- *Waveform*—Name of the signal, expression, or family of waveforms.
- *Cross Type*—Points at which the curves of the waveform intersect with the threshold. While intersecting, the curve may be either rising (`rising`) or falling (`falling`).
- *mode*—Specifies the mode used to calculate the threshold value. If you want to specify the threshold value, select `user`. If you want that the Virtuoso Visualization and Analysis XL calculates the threshold value, select `auto`. The auto threshold is calculated as:

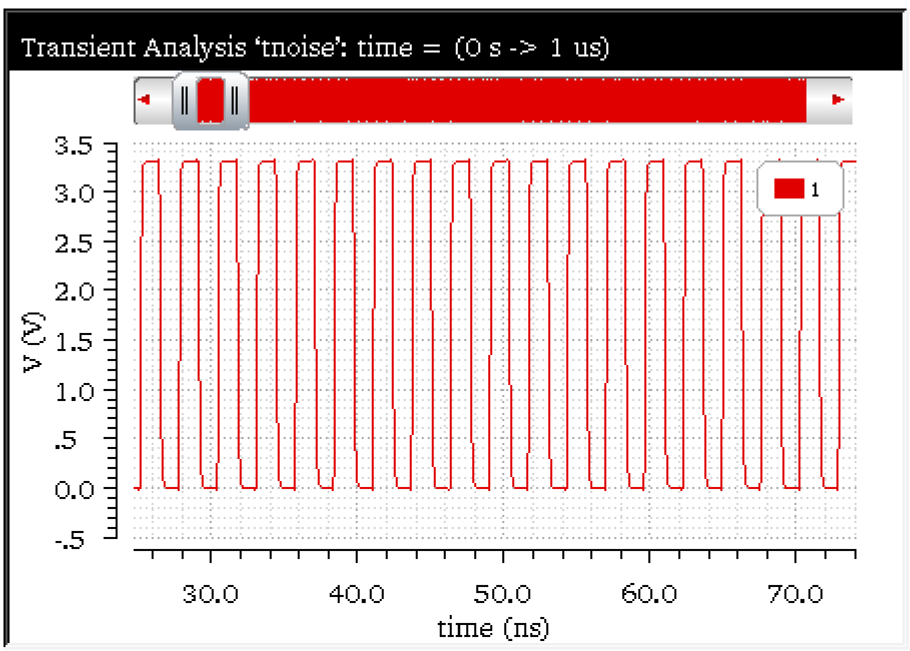
  Auto Threshold Value = integral of the waveform divided by the X range.
- *Threshold*—Threshold value against which the frequency is to be calculated. You need to specify the threshold value only if the *mode* is `user`.
- *Bin Size*—Width of the moving average window, The deviation of value at the particular point from the average of this window is the jitter.
  If *binsize*=0, all frequencies are used to calculate the average.
  If *binsize*=*N*, the last *N* frequencies are used to calculate the average.
- *Plot/print vs.*—Specifies whether you want to retrieve the frequency jitter against *time* (or another X-axis parameter for non-transient data) or *cycle*. Cycle numbers refer to the n'th occurrence of the delay event in the input waveform.
- *Output Type*—Type of output. If set to `sd`, the output is a standard deviation jitter. If set to `plot`, the output is a waveform. The default value is `plot`.

In the MDL mode,

- *sig* is the name of the signal.
- *thresh* is the threshold Y-axis value to be crossed.
- *dir* is the direction of the crossing event.
- *binsize* is the integer used to calculate the average frequency of the signal.
  If *binsize*=0, all frequencies are used to calculate the average.
  If *binsize*=*N*, the last *N* frequencies are used to calculate the average.

Example

This example shows the output waveform generated when you apply the `freq_jitter` function on the following waveform:



The arguments specified in this example are as follows:

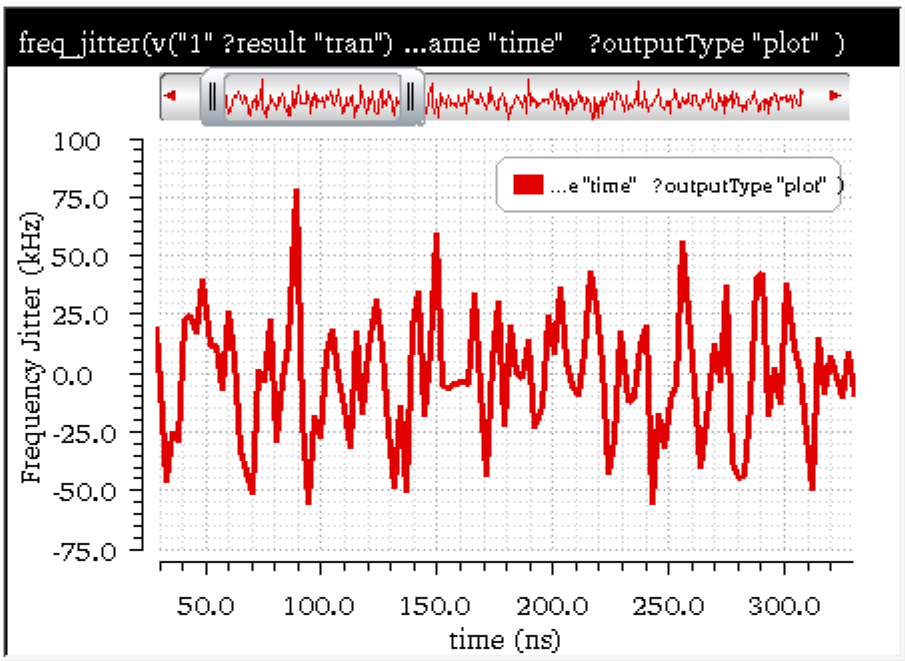
- *Signal*—`v("1" ?results "tran")`
- *Cross Type*—`rising`
- *mode*—`auto`
- *Threshold*—`1.5`
- *Bin Size*—`30`

- *Plot/print vs.*—`time`
- *Output Type*—`plot`

The following expression is created in the Buffer:

```
freq_jitter(v("1"?result "tran" ?resultsDir “nand2_ring_tnoise.raw”)
"rising"  ?mode "auto"   ?binSize 30 ?xName "time"   ?outputType
"plot" )
```

When you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN function for `freq_jitter` is:

```
freq_jitter( o_waveform t_crossType
     [?mode t_mode] [?threshold n_threshold] [binSize n_binSize]
     [?xName t_xName] [?outputType t_outputType] )
     => o_waveform/f_val/nil
```

For more information about this OCEAN function, see freq_jitter in *OCEAN Reference*.

frequency

Returns the average frequency of all cycles plotted. This function is available only in the SKILL mode.

The following equation describes how the average frequency can be calculated:

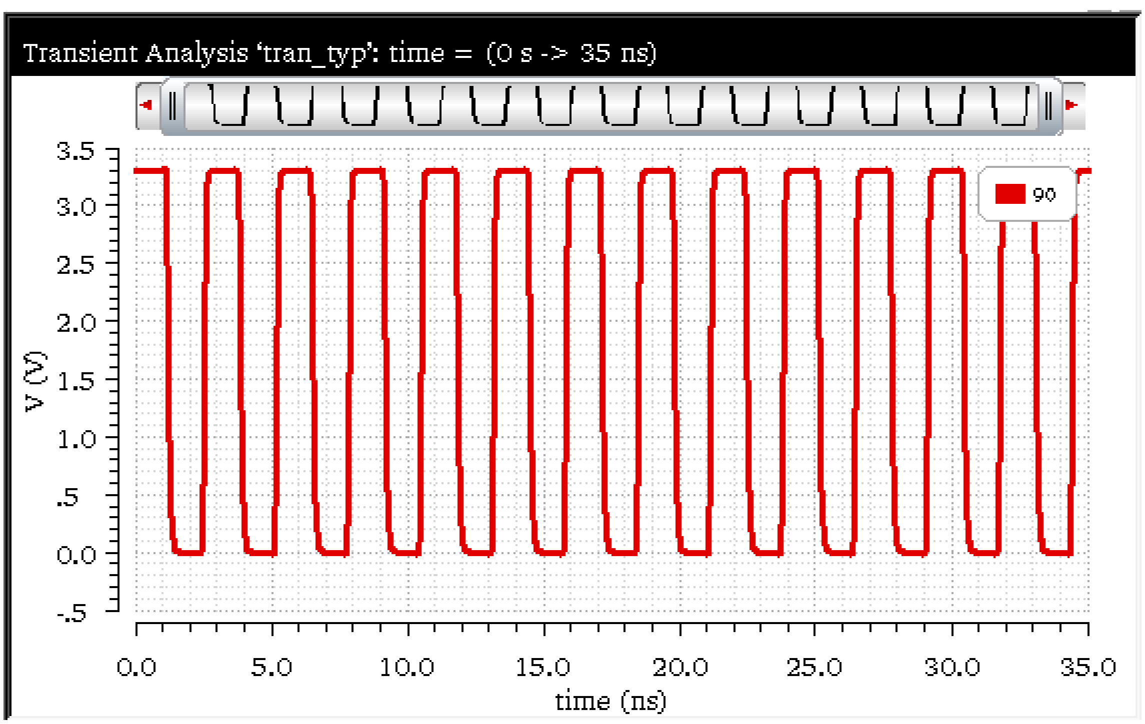
$$avgFreq = \frac{\text{Total number of crosses} - 1}{\text{Total X range between first and last cross}}$$

Here, threshold is defined as:

$$Threshold = \left(\frac{Y_{max} + Y_{min}}{2}\right)\text{of the input waveform}$$

Example

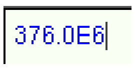
This example shows the output generated when you apply the `frequency` function on the following input waveform (shown in the figure below):



The following expression is created in the Buffer when you apply the `frequency` function:

```
frequency(v("90" ?result "tran" ?resultsDir “nand2_ring.raw))
```

When you evaluate this expression, the following output value is displayed in the Buffer (shown in the figure below):



Related OCEAN Function

The equivalent OCEAN function for `frequency` is:

```
frequency( o_waveform )
     => o_waveform/n_value/nil
```

For more information about this OCEAN function, see frequency in *OCEAN Reference*.

gainBwProd

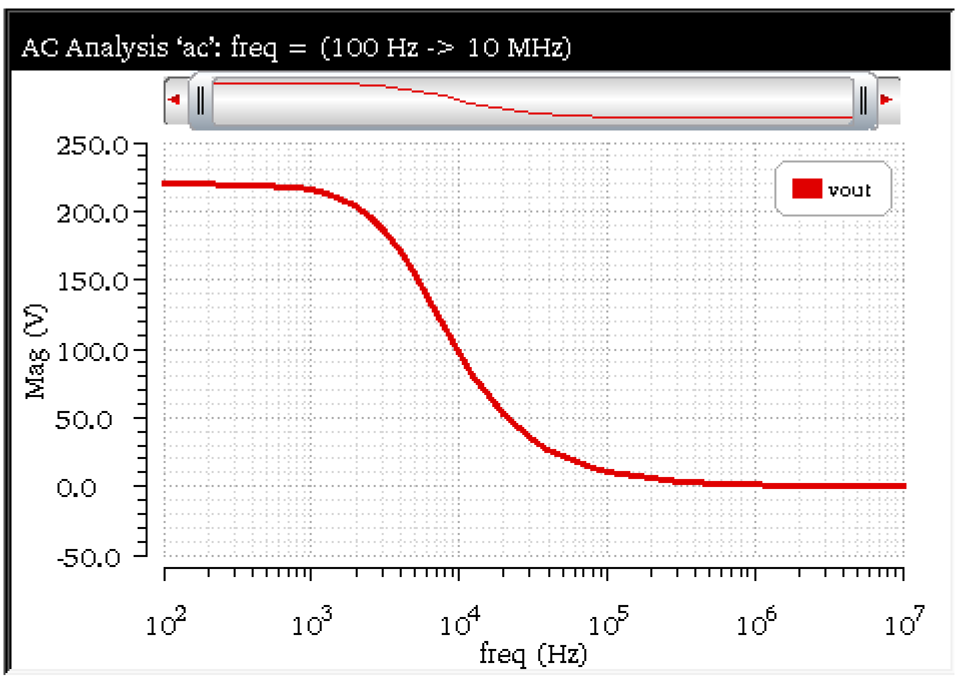
Returns the product of DC gain and upper cutoff frequency for a low-pass type filter or amplifier. (Returns the product of the zero-frequency-gain and 3dB-gain-frequency). Calculates the gain-bandwidth product of a waveform representing the frequency response of interest over a sufficiently large frequency range.

$$gainBwProd(gain) = A_0 \times f_2$$

The gain-bandwidth product is calculated as the product of the DC gain and the critical frequency. The critical frequency is the smallest frequency for which the gain equals 21 times the DC gain 0A.

Example

This example shows the gain bandwidth value generated when you apply the `gainBwProd` function on the following input waveform:



When you apply the function, the following expression is created in the Buffer:

```
gainBwProd(mag(v("vout" ?result "ac" ?resultsDir "./
openloop_ampSim.raw")))
```

Now, when you evaluate this expression, the following result is displayed in the Buffer:

Related OCEAN Function

The equivalent OCEAN command for `gainBwProd` is:

```
gainBwProd( o_waveform )
     => o_waveform/n_value/nil
```

For more information about this OCEAN function, see gainBwProd in *OCEAN Reference*.

gainMargin

Computes the gain margin of the loop gain of an amplifier.

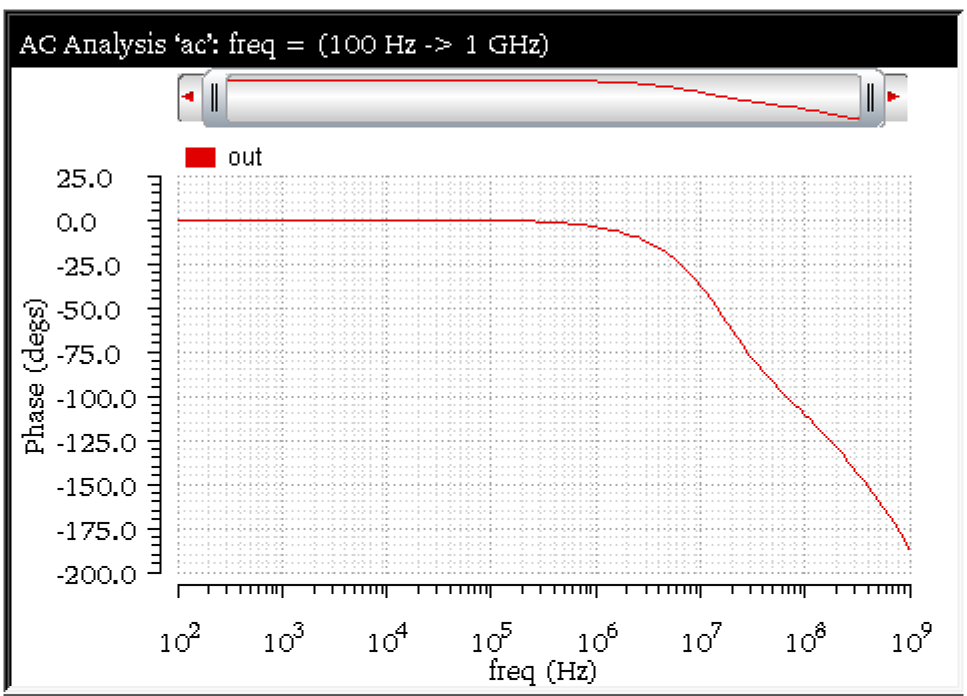
The gain margin is calculated as the magnitude (in dB) of the gain at f0. The frequency f0 is the smallest frequency in which the phase of the gain provided is -180 degrees. For stability, the gain margin must be positive.

$$\text{gainMargin(gain)} = 20\log 10(value(gain\ f_0))$$

The first argument is a waveform representing the loop gain of interest over a sufficiently large frequency range. This command returns the dB value of the waveform when its phase crosses negative pi.

Example

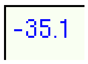
This example shows the gain margin value generated when you apply the `gainMargin` function on the following AC signal:



When you apply this function, the following expression is created in the Buffer:

```
gainMargin(v("out" ?result "ac" ?resultsDir "./ampTest.raw"))
```

Now, when you evaluate this expression, the following output value is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN command for `gainMargin` is:

```
gainMargin( o_waveform [g_stable])
     => o_waveform/n_value/nil
```

For more information about this OCEAN function, see gainMargin in *OCEAN Reference*.

getAsciiWave

Returns a piecewise linear function from a column of x and y values in a file. This function is available only in the SKILL mode.

This function reads an ASCII data file and generates a waveform object from the specified data. The X-axis data must contain real numbers, where as the Y-axis data can be real or complex values. Complex values are represented as (real imag) or complex(real imag). This function skips blank lines and comment lines. Comments are defined as lines beginning with a semicolon.

This function includes the following arguments:

- *Data File Name*—Path of the ASCII data file name.
- *X Column*—Number of columns to be read as X-axis.
- *Y Column*—Number of columns to be read as Y-axis.
- *Skip X values*—Skip X value set skip lines.
- *Skip Y values*—Skip Y value set skip lines.

Example

This example shows the graph plotted when you apply the `getAsciiWave` function on the following data file:

```
./vout_tran_xy.csv
```

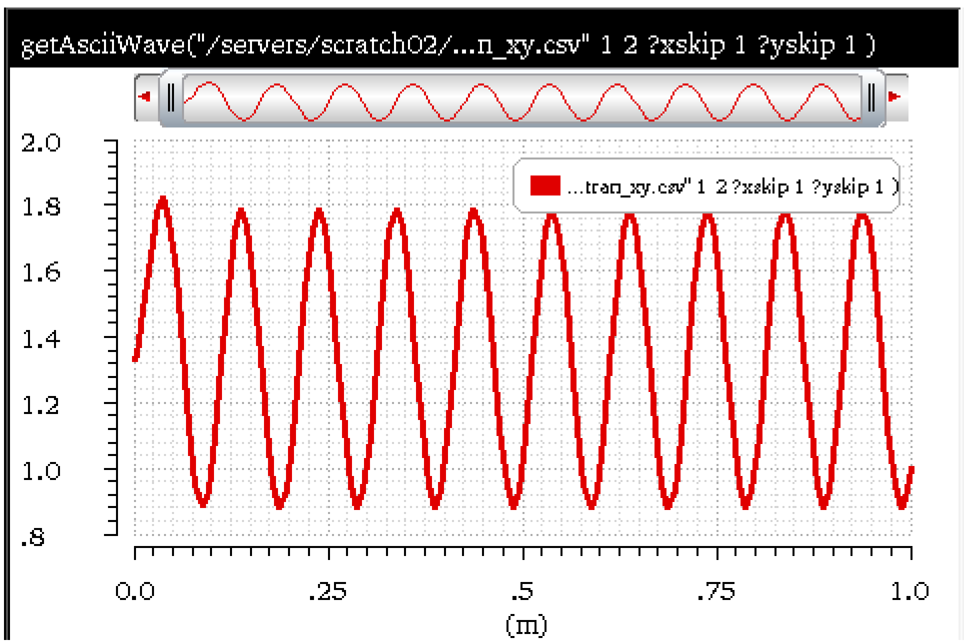
The text file contains the data in the following format:

Time	V(OUT)
0	1.333820380642945
1e-06	1.335676012809686
:	:
:	:

The following expression is created in the Buffer:

```
getAsciiWave("./vout_tran_xy.csv" 1 2 ?xskip 1 ?yskip 1 )
```

When you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `getAsciiWave` is:

```
getAsciiWave( t_filename x_xColumn x_yColumn [x_xskip] [x_yskip])
     => o_wave/nil
```

For more information about this OCEAN function, see getAsciiWave in *OCEAN Reference*.

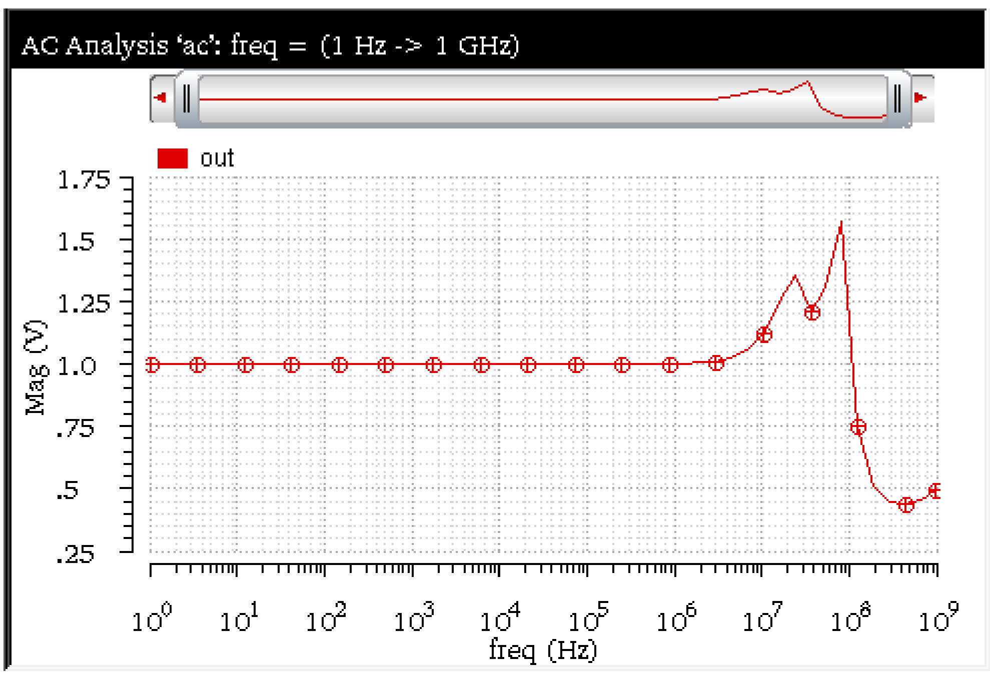
groupDelay

Computes the group delay of the expression in the buffer. Group delay is defined as the derivative of the phase with respect to frequency. Group delay is expressed in seconds. The following equation is used to calculate this function:

$$GroupDelay = \frac{d\phi}{d\omega} = \left(-\frac{d}{df}\right)\left[\frac{phase(/netX)}{360}\right]$$

Example

Consider the following input signal form the AC analysis:



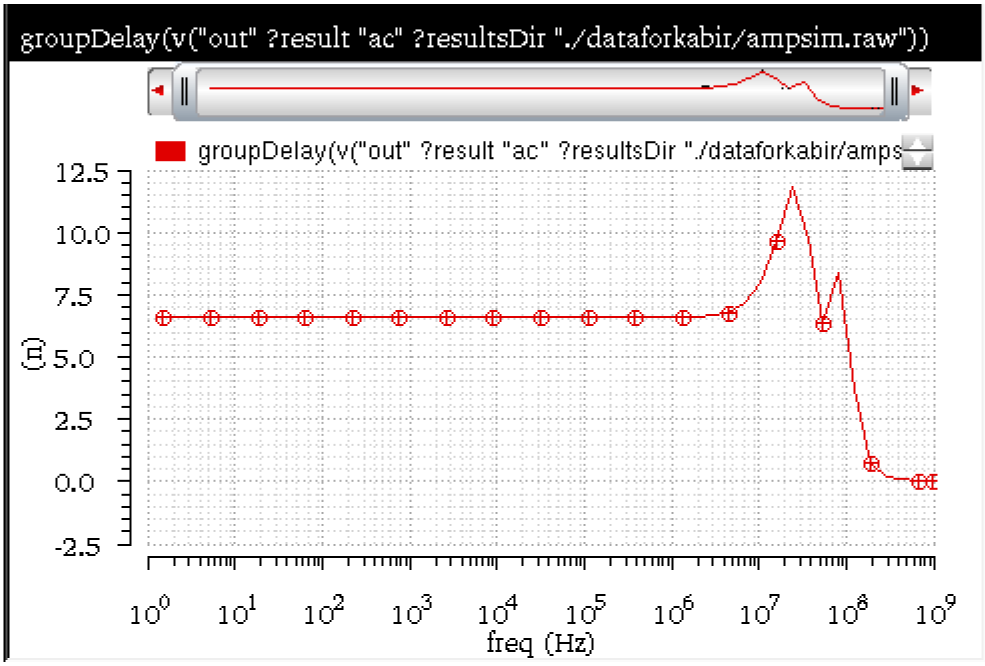
When you send this signal to Calculator, the expression for this signal is displayed in the Buffer. By default, this expression is based on the magnitude of the signal and represented as shown below:

```
mag(v("out" ?result "ac" ?resultsDir "./ampsim.raw"))
```

Edit this expression to remove the magnitude and select the `groupDelay` function in the Function Panel. The function is applied to the signal expression in the Buffer. The corresponding expression created in the Buffer is as follows:

```
groupDelay(v("out" ?result "ac" ?resultsDir "./ampsim.raw"))
```

When you evaluate this expression, the group delay is calculated and the following output waveform is displayed in a new graph window.



Related OCEAN Function

The equivalent OCEAN command for `groupDelay` is:

```
groupDelay( o_waveform )
     => o_waveform/nil
```

For more information, see `groupDelay` in *OCEAN Reference*.

harmonic

Returns the harmonic waveform of the specified harmonic. This function is available only in the SKILL mode.

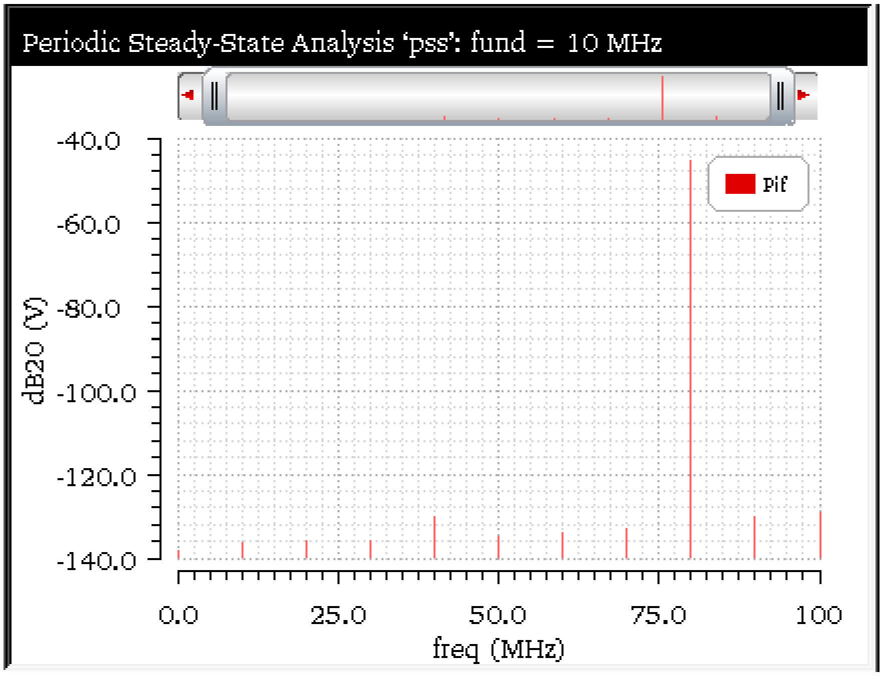
This function includes the following arguments:

- *Signal*—Name of the signal.
- *Harmonic Number* is the index number that designates the harmonic information to be returned. For the pss, pac, and pxf analyses, the index is an integer. For the pdisto analysis, the index is a list of integers that correspond to the frequency names listed in the funds analysis parameter in the netlist. You can specify more than one harmonic number at a time.

Example

This example shows the harmonic value generated when you apply the `harmonic` function on the following input waveform:

Note: Ensure that the dependent modifier displayed on Y-axis of the input waveform is `dB20`. For more details about how to change the dependent modifier, see Plotting Traces Using Different Modifiers on page 125.



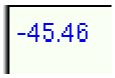
Note that in this waveform, the dB20 of the input signal is plotted against frequency on X-axis.The following arguments are specified in this example:

- *Signal*—`db20(v("Pif" ?results "pss_fd"))`
- *Harmonic Number*—`8`

The following expression is created in the Buffer:

```
harmonic(db20(v("Pif" ?result "pss_fd" ?resultsDir "./
pss_ne600.raw")) 8 )
```

When you evaluate this expression, the following output value is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN function for `harmonic` is:

```
harmonic( o_waveform h_index )
     => o_waveform/g_value/nil
```

For more information about this OCEAN function, see harmonic in *OCEAN Reference*.

harmonicFreq

Returns the frequency(s) of the harmonic waveform for the specified harmonic. This function is available only in the SKILL mode.

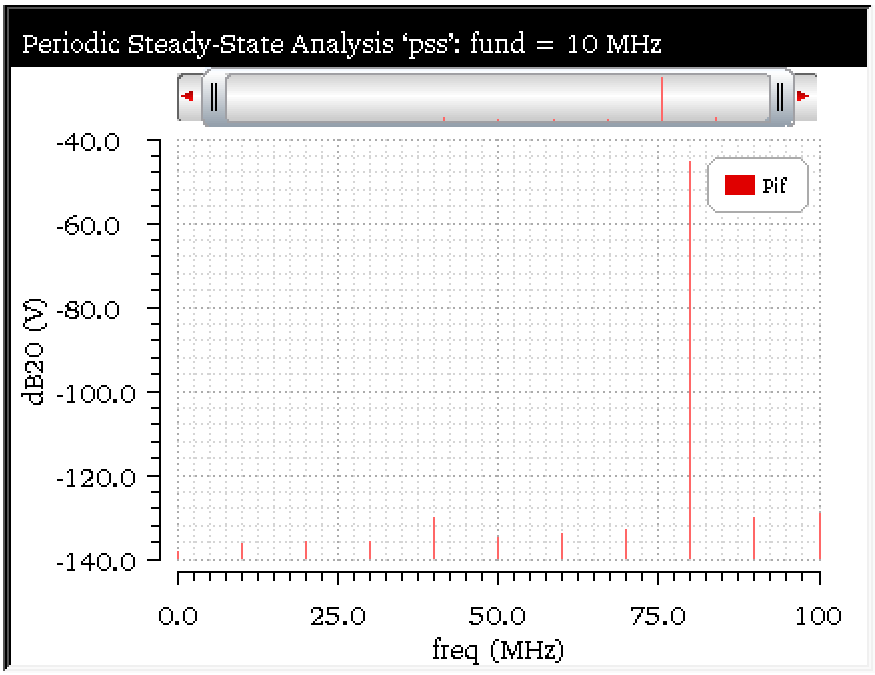
This function includes the following arguments:

- *Signal*—Name of the signal.
- *Harmonic Number*—Index number that designates the harmonic information to be returned. For the pss, pac, and pxf analyses, the index is an integer. For the pdisto analysis, the index is a list of integers that correspond to the frequency names listed in the funds analysis parameter in the netlist. You can specify more than one harmonic number at a time.

Example

This example shows the harmonic frequency calculated when you apply the `harmonicFreq` function on the following input signal.

Note: Ensure that the dependent modifier displayed on Y-axis of the input waveform is `dB20`. For more details about how to change the dependent modifier, see Plotting Traces Using Different Modifiers on page 125.



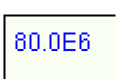
The following arguments are specified in this example:

- *Signal*—`db20(v("Pif" ?result "pss_fd"))`
- *Harmonic Number*—`8`

The following expression is created in the Buffer:

```
harmonic(xval(db20(v("Pif" ?result "pss_fd" ?resultsDir "./
pss_ne600.raw"))), 8)
```

When you evaluate this expression, the following output value is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN command for `harmonicFreq` is:

```
harmonic( xval(o_waveform) h_index )
```

For more information about this OCEAN function, see harmonic in *OCEAN Reference*.

histogram2D

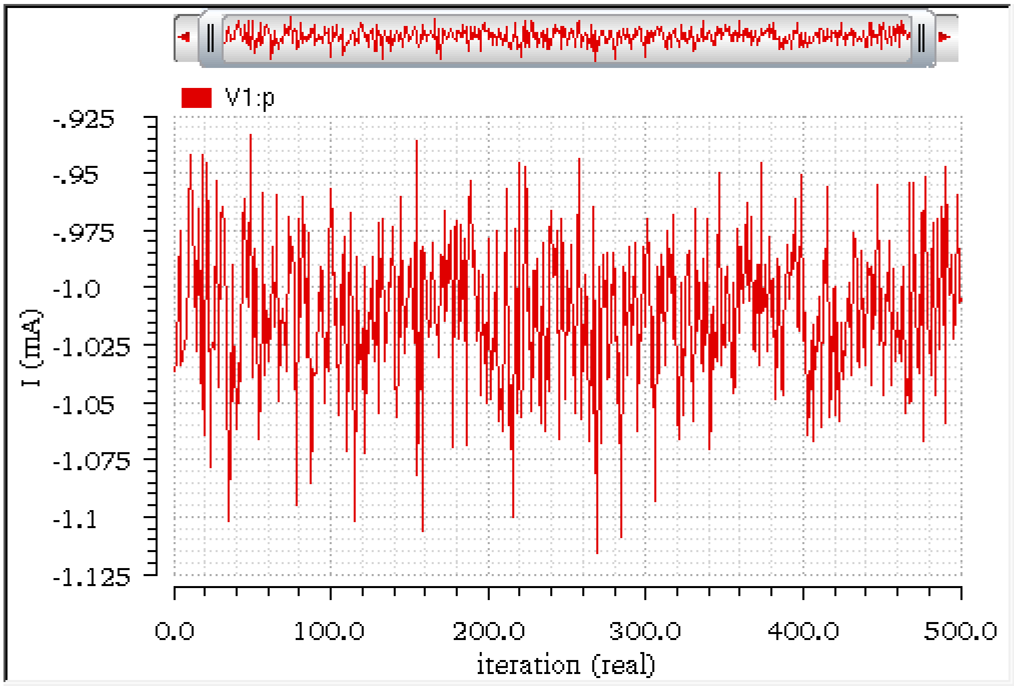
Returns a waveform that represents the statistical distribution of input data in the form of a histogram. The height of the bars (or bins) in the histogram represents the frequency of the occurrence of values within a specific period.

This function includes the following arguments:

- *Signal*—Name of the waveform, expression, or family of waveforms. The expression displayed in the Buffer can be added to this field by selecting `buffer`.
- *Number of bins*—Specifiy how many bars you want to plot in the resulting histogram plot.
  Default value:`10`.
- *Type*—Specify the type of histogram to be plotted.
  Valid values: `Standard`, `Cumulative line`, and `Cumulative box`.
  Default value: `Standard`.
- *Std Dev Lines*—Select `yes` from the drop-down if you want to display the standard deviation lines in the resulting histogram plot, indicating the mean, (mean - standard deviation), and (mean + standard deviation) values.
  Default value: `no`.
- *Density Estimator*—Select `yes` from the drop-down if you want that the resulting histogram plot displays a curve that estimates the distribution concentration.
  Default value: no.

Example

Consider the following input waveform from Monte Carlo analysis:



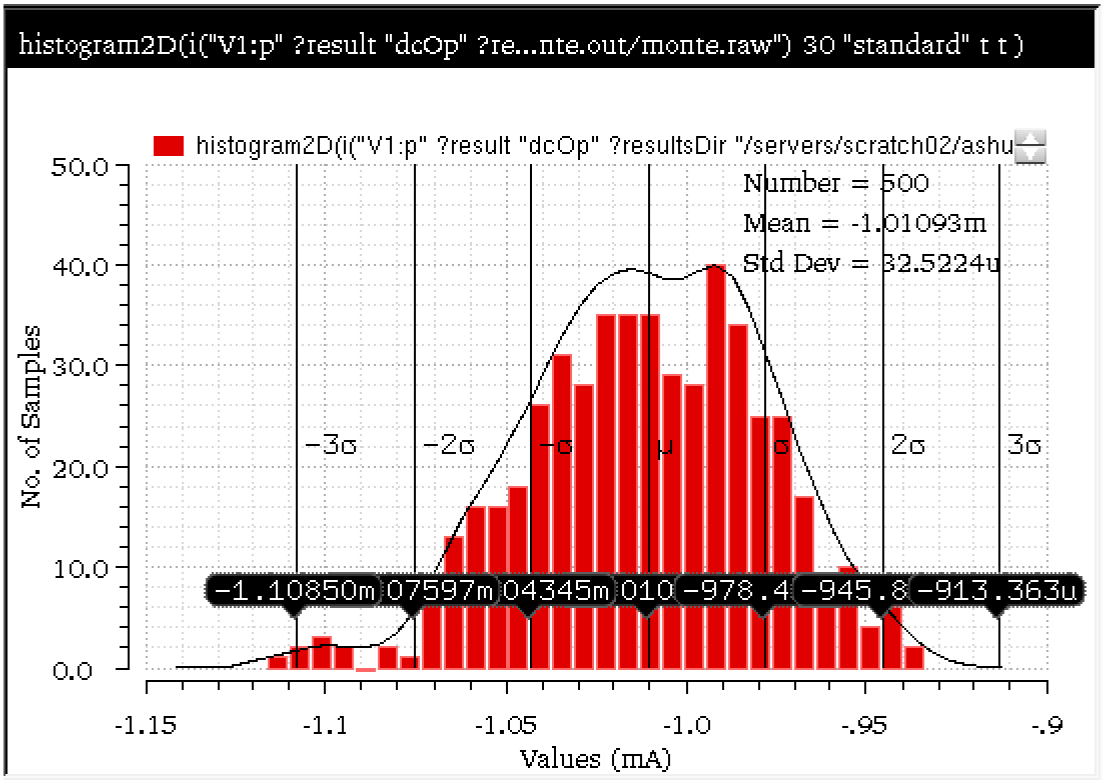
The `histogram2D` function is applied on this input waveform with the following arguments:

- *Signal*—`i("V1:p" ?result "dcOp" ?resultsDir "./monte/monte.out/monte.raw")`
- *Number of bins*—`30`
- *Type*—`standard`
- *Std Dev Lines*—`yes`
- *Density Estimator*—`yes`

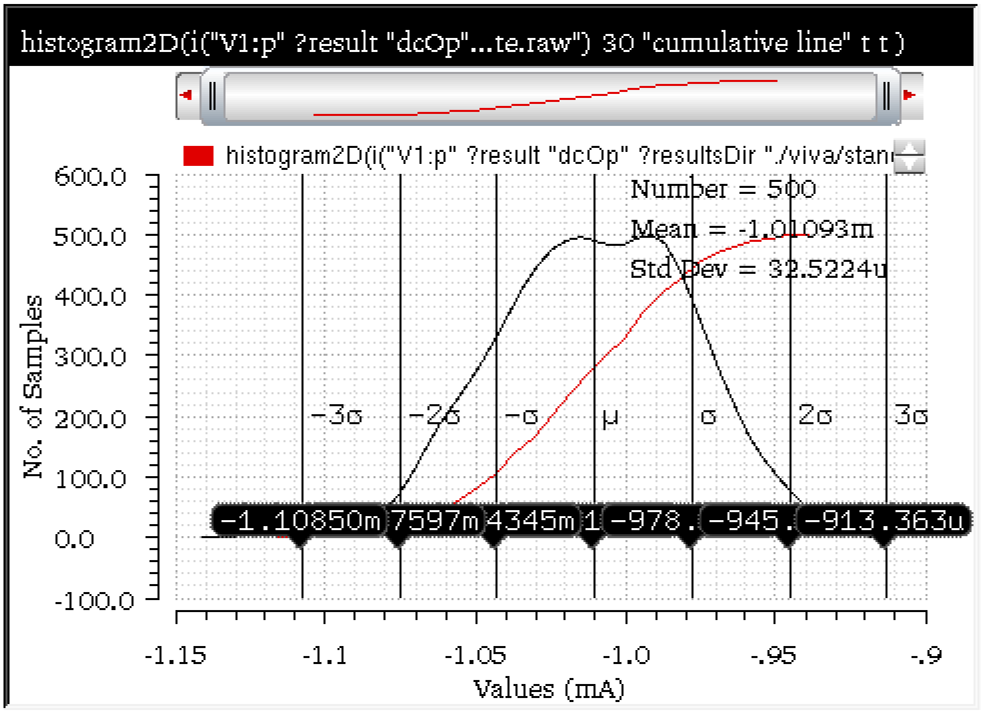
The corresponding expression created in the Buffer is as follows:

```
histogram2D(i("V1:p" ?result "dcOp" ?resultsDir "./monte/monte.out/
monte.raw") 30 "standard" t t )
```

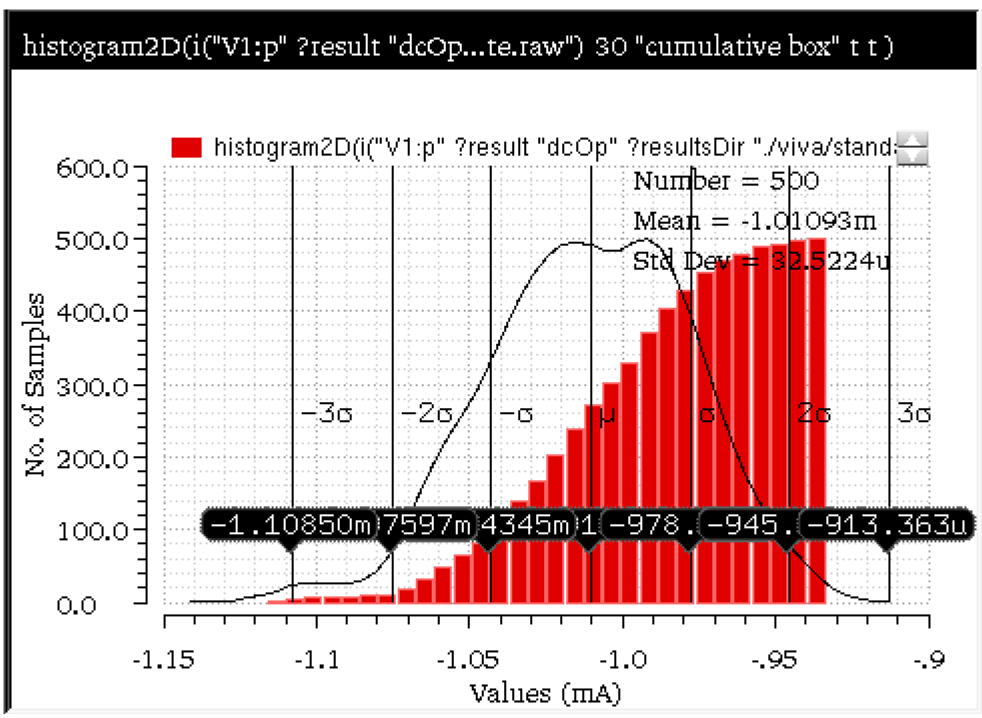
When you evaluate this expression, the following output waveform representing the data in histogram is displayed in a new graph window.



The following figure shows the output generated if you select *Type* as `Comulative line`:



The following figure shows the output generated if you select *Type* as `Comulative box`:



Related OCEAN Function

The equivalent OCEAN command for `histogram2D` is:

```
histogram2D(o_waveform x_nbins s_type g_setAnnotation g_setDensityEstimator)
     => o_waveform/nil
```

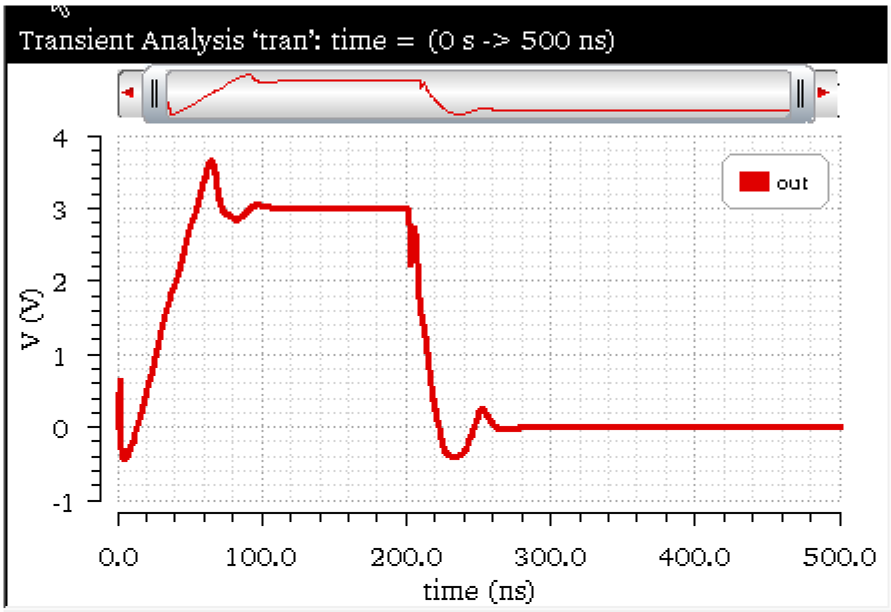
For more information, see `histogram2D` in *OCEAN Reference*.

iinteg

Returns the incremental area under the waveform.

Example

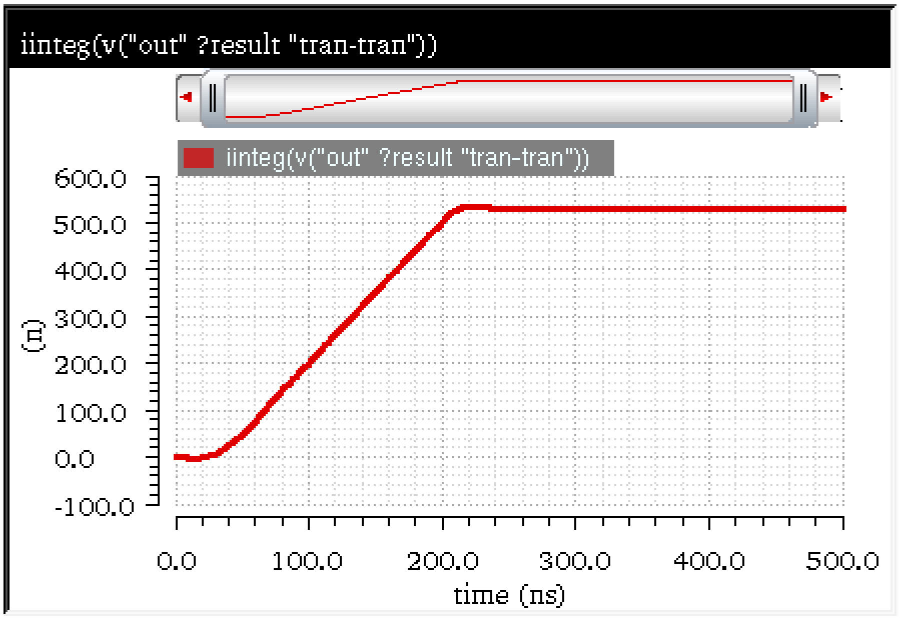
This example calculates the incremental area under the following input waveform:



When you apply the `iinteg` function on this waveform, the following expression is created in the Buffer:

```
iinteg(v("out" ?result "tran-tran" ?resultsDir "ampsim.raw"))
```

When you evaluate this expression, the following output waveform appears:



Related OCEAN Function

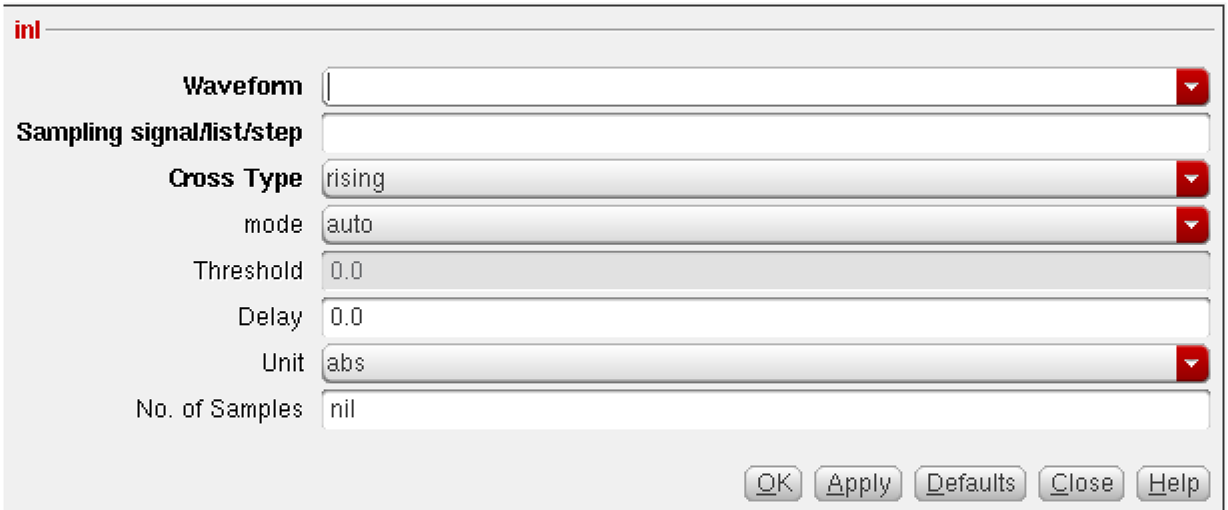
The equivalent OCEAN command for `iinteg` is:

```
iinteg( o_waveform )
     => o_waveform/nil
```

For more information about this OCEAN command, see iinteg in *OCEAN Reference*.

inl

Computes the integral non-linearity of a transient simple or parametric waveform.



This function includes the following arguments:

- *`Waveform`*—Name of the signal.
- *`Samplingsignal/list/step`*—Signal used to obtain the points for sampling the *Waveform (*points at which the waveform crosses the threshold while either `rising` or `falling` as specified in the *cross Type* field with the *delay* added to them), list of domain values at which the sample points are obtained from the *Waveform,* or the sampling interval*.*
- *`Cross Type`*—Specifies the points at which the curves of the waveform intersect with the threshold. While intersecting, the curve may be either rising (`rising`) or falling (`falling`).
- *`mode`*—Specifies whether the threshold value is to be calculated by Virtuoso Visualization and Analysis XL (`auto`) or specified by you (`user`). The auto threshold is calculated as:

  Auto Threshold Value = integral of the waveform divided by the X range
- *`Threshold`*—Threshold value against which the frequency is to be calculated. You need to specify the threshold value only if the *mode* is `auto`
- *`Delay`*—Delay time after which the sampling begins.
- *`Unit`*—Specifies whether the output waveform is to be output as an absolute value (`abs`) or multiples of least significant bit (`lsb`).

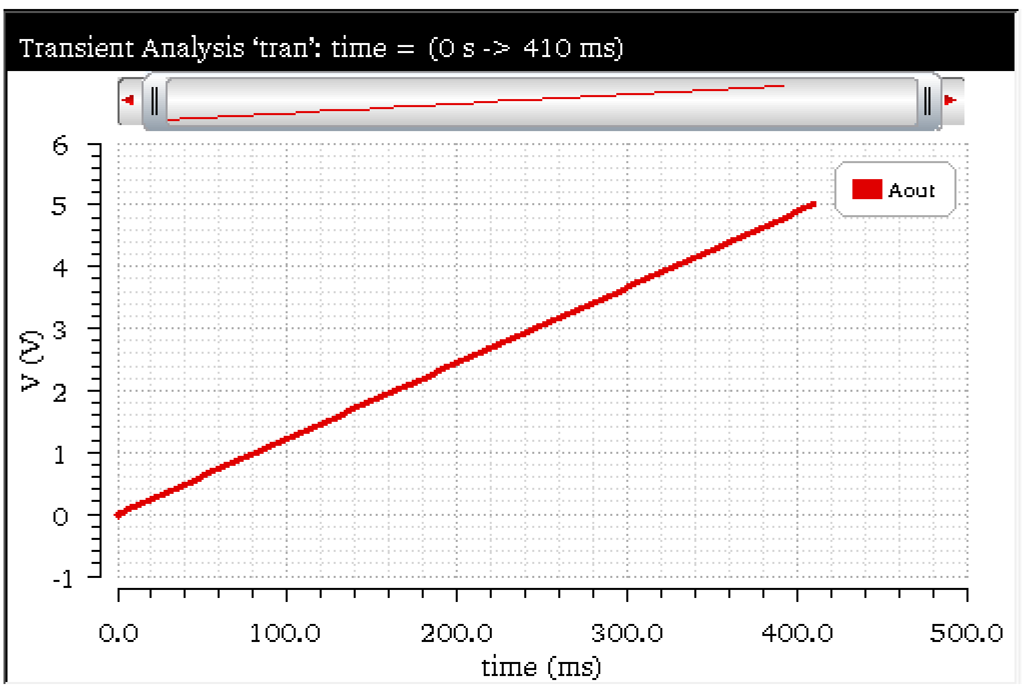
- *No. of Samples*—Specifies the samples used for calculating the non-linearity. If not specified, the samples are taken against the entire data window.

Note: For each of the three ways in which the sample points can be specified, only a few of the other optional arguments are meaningful,

- For *Sampling signal*, the fields *Cross Type*, *mode*, *Threshold*, *Delay*, *Method*, and *Units* are meaningful.
- For *list*, the fields *Method* and *Units* are meaningful.
- For *step*, the fields *Method*, *Units*, and *No. of samples* are meaningful.

Example

This example shows the output waveform generated when you apply the inl function on the following input waveform:



The following arguments are specified in this example:

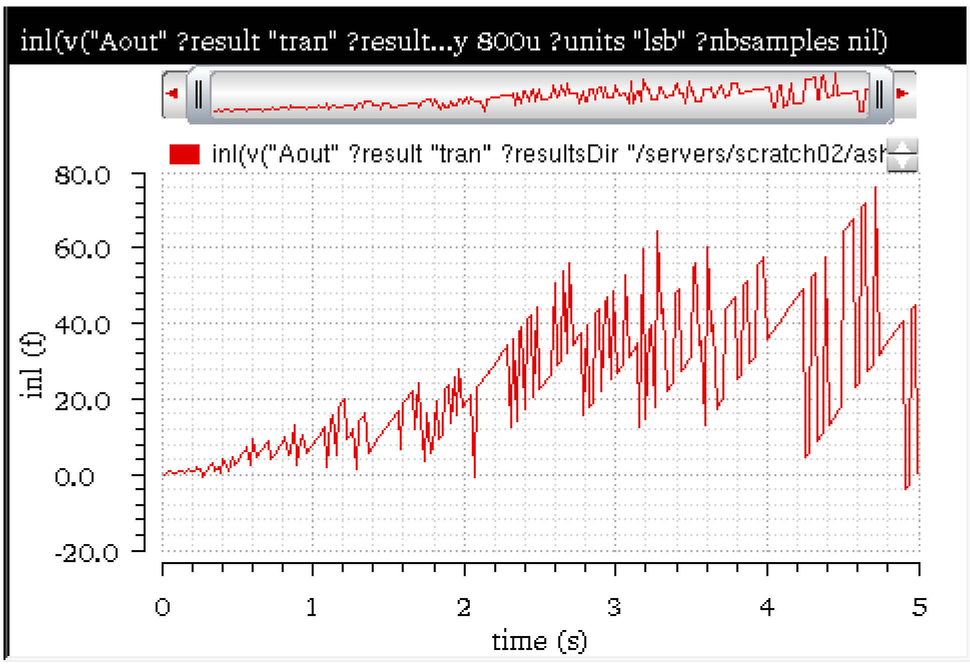
- *Waveform*—`v("Aout" ?result "tran")`
- *Sampling signal/list/step*—`0.0016`
- *Cross Type*—`rising`
- *mode*—`auto`

- *Threshold*—`0.0`
- *Delay*—`800u`
- *Unit*—`lsb`
- *No. of Samples*—`nil`

The following expression is created in the Buffer with the specified argument

```
inl(v("Aout" ?result "tran" ?resultsDir "./psf_forDNL") 0.0016  ?mode
"auto"  ?crossType "rising" ?delay 800u ?units "lsb" ?nbsamples nil)
```

Now, when you evaluated this expression, the followign output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `inl` is:

```
inl( o_dacSignal o_sample|o_pointList|n_interval
     [?mode t_mode] [?threshold n_threshold] [?crossType t_crossType]
     [?delay f_delay] [?units x_units] [?nbsamples n_nbsamples] )
     => n_inl/nil
```

For more information about this OCEAN function, see inl in *OCEAN Reference*.

integ

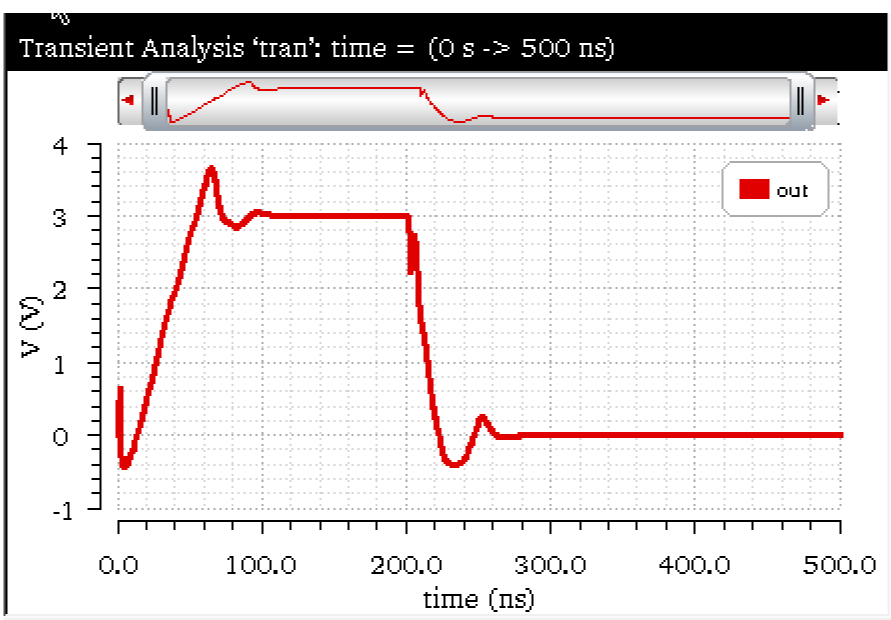
Returns the area bounded under the curve. Computes the definite integral of the waveform with respect to a range specified on the X-axis of the waveform. The result is the value of the area under the curve over the range specified on the X-axis.

This function includes the following arguments:

- *Signal*—Name of the signal.
- *Initial Value*—Initial Value is the X start point of area.
- *Final Value*—Final Value is the X end point of area.

Example

This example shows the definite integral value calculated when you apply the `integ` function on the following input waveform:



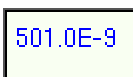
The following arguments are specified in this example:

- *Signal*—`v(“out” ?result “tran”)`
- *Initial Value*—`0`
- *Final Value*—`200n`

The following expression is created in the Buffer:

```
integ(v("out" ?result "tran-tran" ?results "./ampsim.raw") 0 200n )
```

When you evaluate this expression, the following result value is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN command for `integ` is:

```
integ( o_waveform,[n_intial_limit,n_final_limit] )
     => o_waveform/n_value/nil
```

For more information about this OCEAN function, see integ in *OCEAN Reference*.

intersect

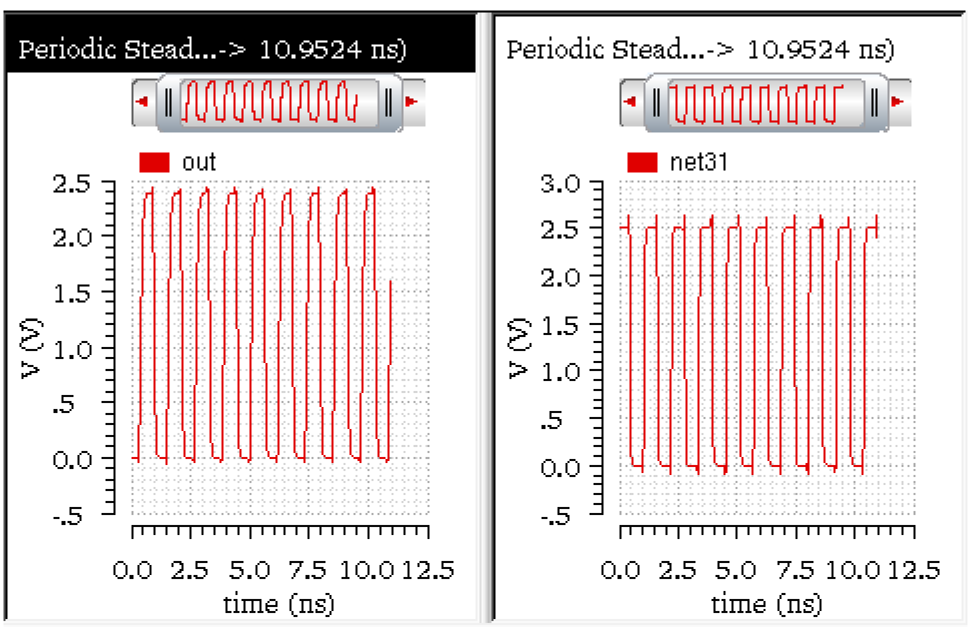
Returns all the points at which two waveforms intersect each other. The intersect function can be used on families of traces swept on the same parameter names and values.

This function includes the following arguments:

- *Signal1*—Name of the first waveform.
- *Signal2*—Name of the second waveform.

Example

This example shows the output waveform generated when you apply the `intersect` function on the following input waveforms:



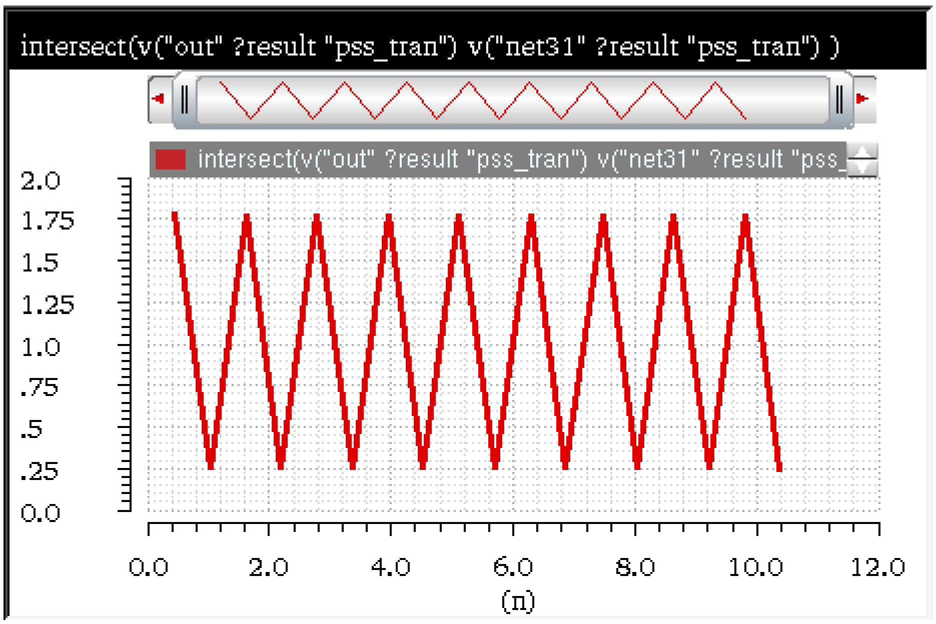
The following arguments are specified in this example:

- *Signal1*—`v("out" ?result "pss_tran")`
- *Signal2*—`v("net31" ?result "pss_tran")`

The following expression is created in the Buffer:

```
intersect(v("out" ?result "pss_tran" ?resultsDir “osc13.raw”)
v("net31" ?result "pss_tran" ?resultsDir “osc13.raw”))
```

Now when you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `intersect` is:

```
intersect( o_waveform1 o_waveform2 )
     => o_wave/nil
```

For more information about this OCEAN function, see intersect in *OCEAN Reference*.

ipn

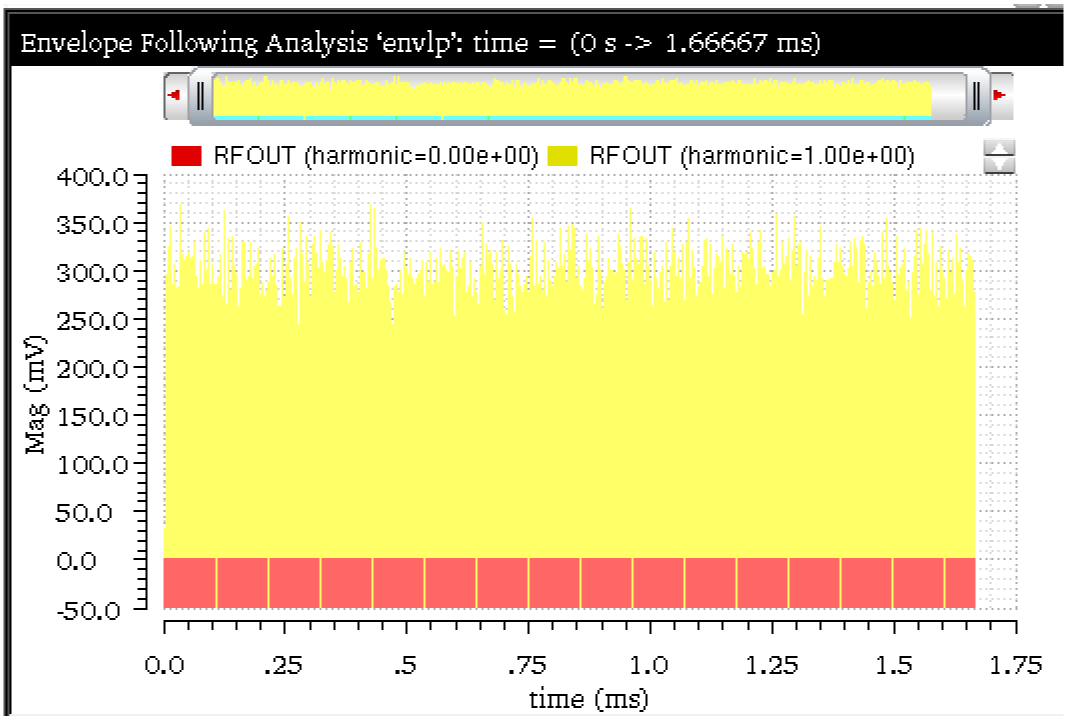
Plots the *N*th order intercept between two harmonics of a waveform that you define. This function is available only in the SKILL mode.

This function includes the following arguments:

- *Signal*—Name of the signal.
- *Spur Order*—Determines what order of interference is calculated for the spurious and reference waves. The default value `3` corresponds to the IP3 function. If you use a value other than 3, that order of interference is calculated between those two waves.
- *Spur Harmonic*—Harmonic number for the spurious waveform.
- *Extrapolation Point*—Extrapolation point for the IPN function. This is the X-axis value.
- *Reference Harmonic*—Harmonic number for the reference waveform.

Example

This example shows the output generated when you apply the `ipn` function on the following input signal:



The following arguments are specified in this example:

- *Signal*—`(v("/RFOUT" ?result "envlp_fd"))`
- *Spur Order*—`3`
- *Spur Harmonic*— `-2`
- *Extrapolation Point*— `-50`
- *Reference Harmonic*—`0`

The following expression is created in the Buffer:

```
ipn(dB20(harmonic(v("/RFOUT" ?result "envlp_fd" ?resultsDir "./
rfworkshop/simulation/EF_example_envlp/spectre/schematic/psf"),-
2)), dB20(harmonic(v("/RFOUT" ?result "envlp_fd" ?resultsDir "./
rfworkshop/simulation/EF_example_envlp/spectre/schematic/psf"),
0)), 3, 1, -50, -50)
```

Now, when you evaluate this expression, the following output value is displayed in the Buffer:

-50.0

Related OCEAN Function

The equivalent OCEAN function for `ipn` is:

```
ipn( o_spurious o_reference
     [ f_ordspur f_ordref f_epspur f_epref g_psweep s_measure ] )
     => o_waveform/f_number/nil
```

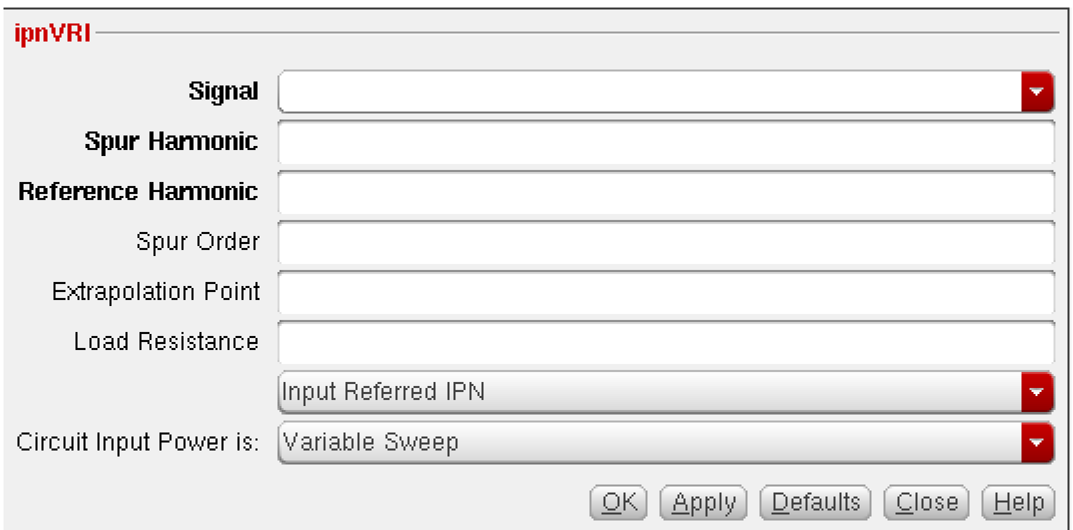
For more information about this OCEAN function, see ipn in *OCEAN Reference*.

ipnVRI

Performs an intermodulation *N*th-order intercept point measurement

You can use this function to simplify the declaration of an IPN measurement. This function extracts the spurious and reference harmonics from the input waveform(s), and uses dBm(spectralPower((i or v/r),v)) to calculate the respective powers. The function then passes these power curves or numbers and the remaining arguments to the IPN function to complete the measurement.

From each of the spurious and reference power waveforms (or points), the IPN function extrapolates a line of constant slope (dB/dB) according to the specified order and input power level. These lines represent constant small-signal power gain (ideal gain). The IPN function calculates the intersection of these two lines and returns the value of either the x coordinate (input referred) or y coordinate (output referred).



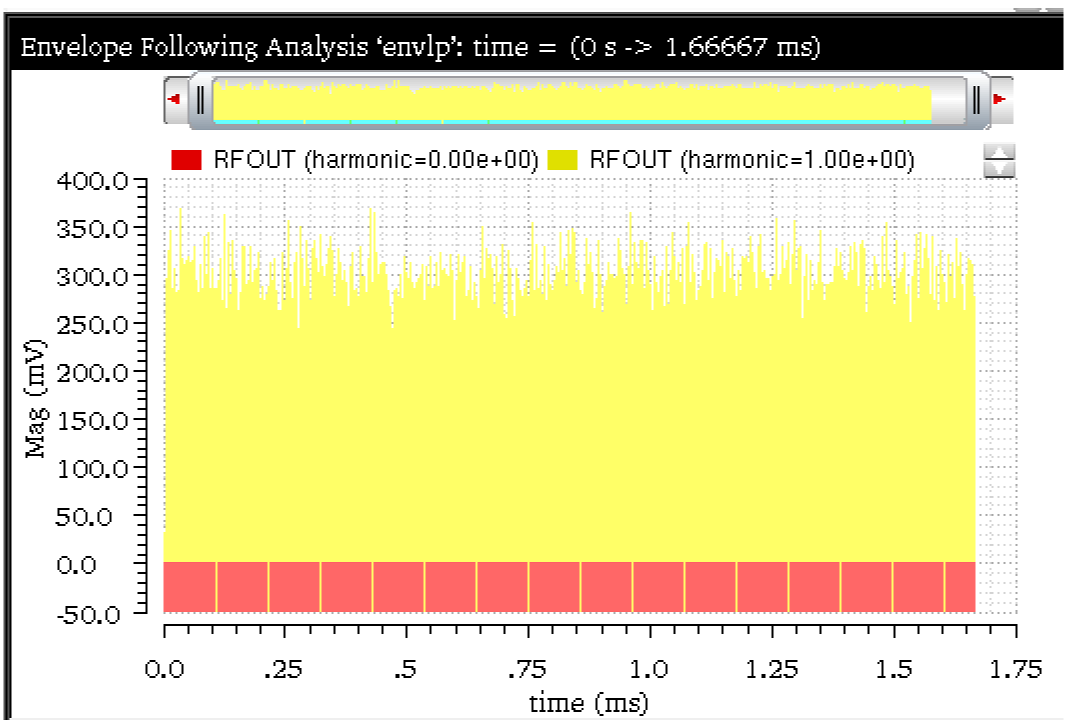
This function includes the following arguments:

- *Signal*—Name of the signal.
- *Spur Harmonic*—Harmonic index for the spurious waveform.
- *Reference Harmonic*—Harmonic index for the reference waveform.
- *Spur Order*—Determines what order of interference is calculated for the spurious and reference waves. The default value `3` corresponds to the IP3 function. If you use a value other than 3, that order of interference is calculated between those two waves.
- *Extrapolation Point*—Extrapolation point for the `ipn` function. This is the X-axis value. The default is the minimum X-axis value of the input voltage waveform.

- *Load Resistance*—Resistance into the output port.
  To get the X-coordinate of the intercept, select *Input Referred IPN*. To get the Y-coordinate of the intercept, specify *Output Referred IPN*.
- *Circuit Input Power*—Specifies whether the input power is a variable sweep or a single point.

Example

This example shows the output generated when you apply the `ipnVRI` function on the following input signal:



The following arguments are specified in this example:

- Signal—`v("/RFOUT" ?result "envlp_fd")`
- *Spur Harmonic*—`-2`
- *Reference Harmonic*—`0`
- *Spur Order*— `3`
- *Extrapolation Point*—`-50`
- *Load Resistance*—`50`

- *Input Referred IPN— Selected*
- *Circuit Input Power is:—*`Variable Sweep`

The following expression is created in the Buffer:

```
ipnVRI(v("/RFOUT" ?result "envlp_fd" ?resultsDir "./
EF_example_envlp/spectre/schematic/psf") -2 0 ?ordspur 3 ?epoint -50
?rport 50 )
```

Now, when you evaluate this expression, the following output value is dispalyed in the Buffer:

```
-50.0
```

Related OCEAN Function

The equivalent OCEAN command for `ipnVRI` is:

```
ipnVRI( o_vport x_harmspur x_harmref
     [?iport o_iport] [?rport f_rport] [?ordspur f_ordspur]
     [?epoint f_epoint] [?psweep g_psweep] [?epref f_epref]
     [?ordref f_ordref] [?measure s_measure] )
     => o_waveform/f_number/nil
```

For more information about this OCEAN function, see ipnVRI in *OCEAN Reference*.

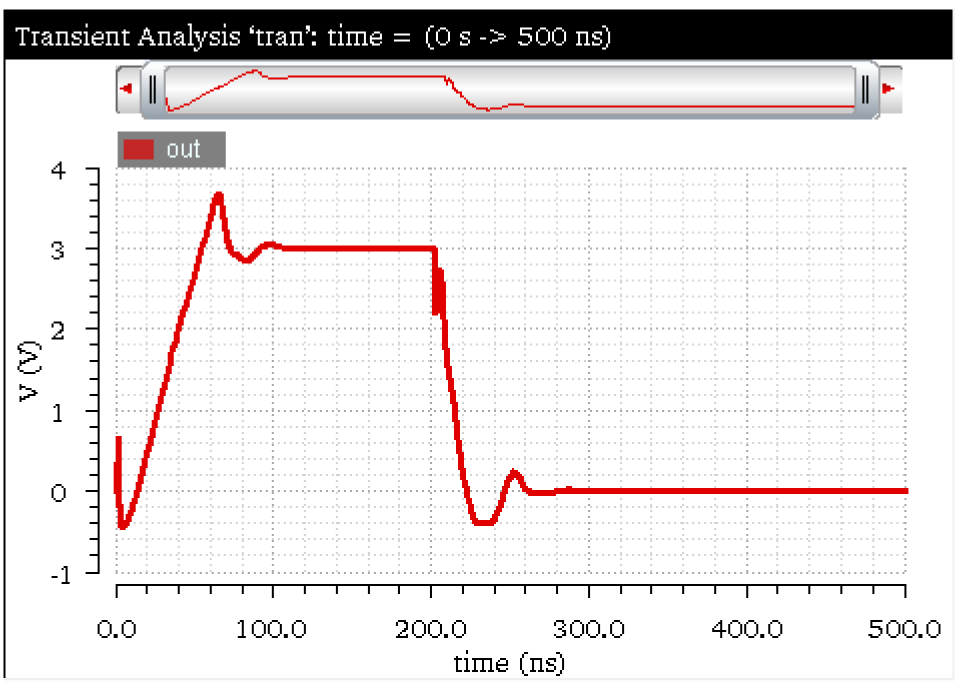
lshift

Shifts the data in the Graph Window to the left by the specified amount. A negative value shifts the data to the right. This function is available only in the SKILL mode.

- *Signal*—Name of the signal.
- *Delta X*—Amount by which you want to shift the data.

Example

This example shows the output waveform left shifted by the specified amount when you apply the `lshift` function on the following input waveform:



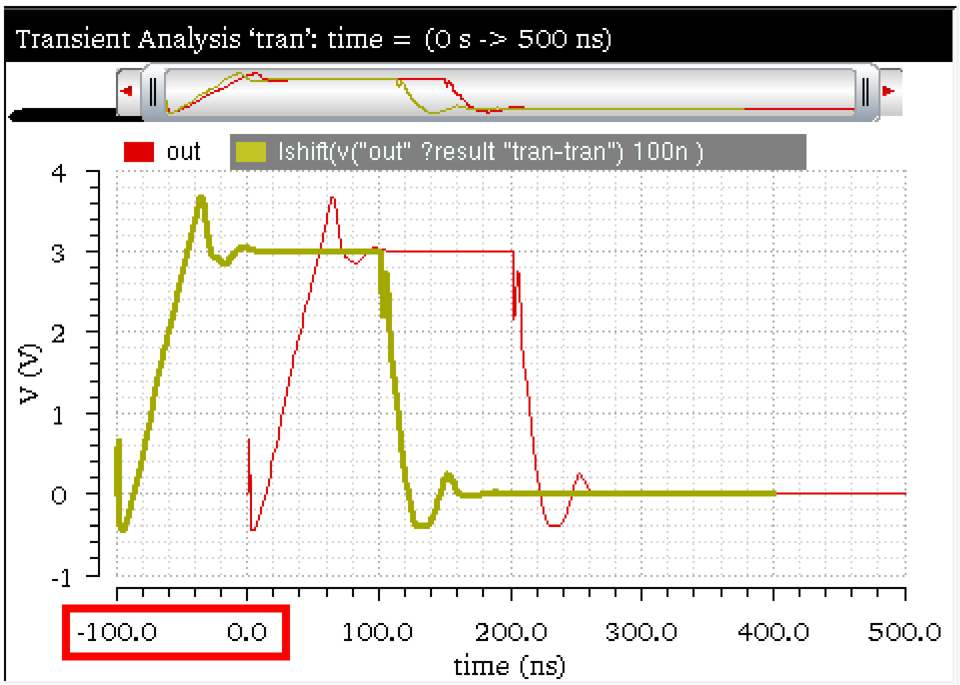
The following arguments are specified in this example:

- *Signal*—`v(out ?result "tran-tran")`
- *Delta X*—`100n`

The following expression is created in the Buffer:

```
lshift(v(out ?result "tran-tran" ?resultsDir "./ampsim.raw") 100n)
```

Now, when you evaluate this expression, the output waveform is shifted by a value of 100n on the X-axis as shown in the figure below.



Related OCEAN Function

The equivalent OCEAN command for `lshift` is:

```
lshift( o_waveform n_delta )
     => o_waveform/nil
```

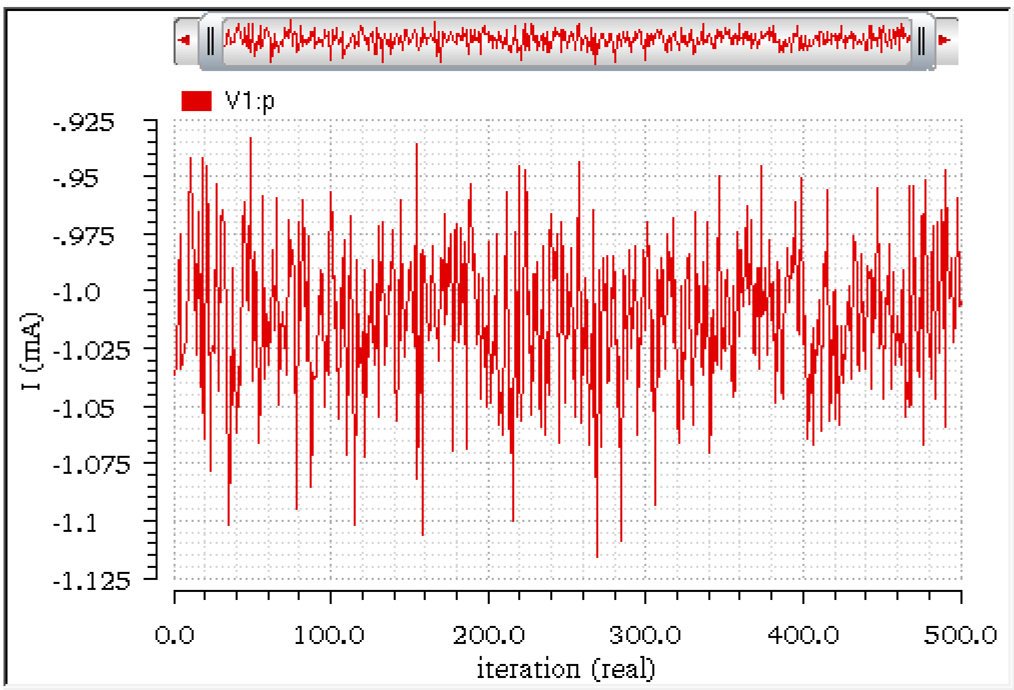
For more information about this OCEAN function, see lshift in *OCEAN Reference*.

normalQQ

Returns a quantile-quantile plot of the sample quantiles versus theoretical quantiles from a normal distribution. If the distribution is normal, the plot is close to a linear waveform.

Example

Consider the following input waveform from the Monte Carlo analysis:



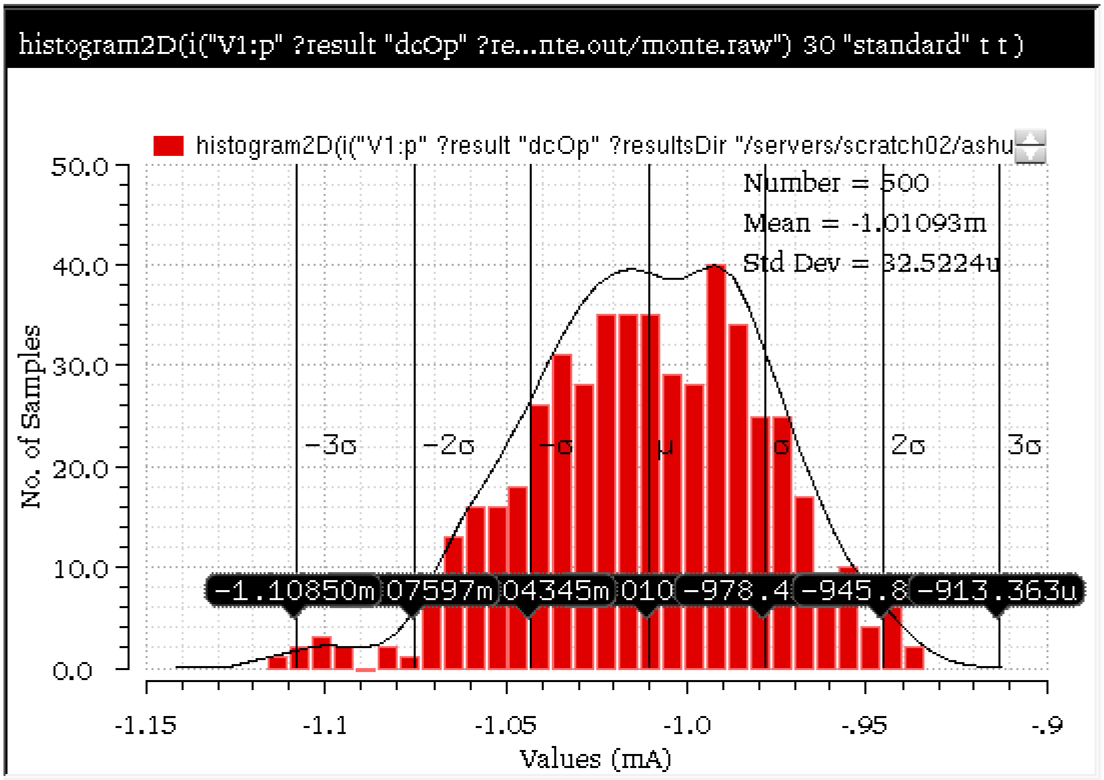
The `histogram2D` function is applied on this input waveform with the following arguments:

- *Signal*— `i("V1:p" ?result "dcOp" ?resultsDir "./monte/monte.out/monte.raw")`
- *Number of bins*—`30`
- *Type*—`standard`
- *Std Dev Lines*—`yes`
- *Density Estimator*—`yes`

The corresponding expression created in the Buffer is as follows:

```
histogram2D(i("V1:p" ?result "dcOp" ?resultsDir "./monte/monte.out/
monte.raw") 30 "standard" t t )
```

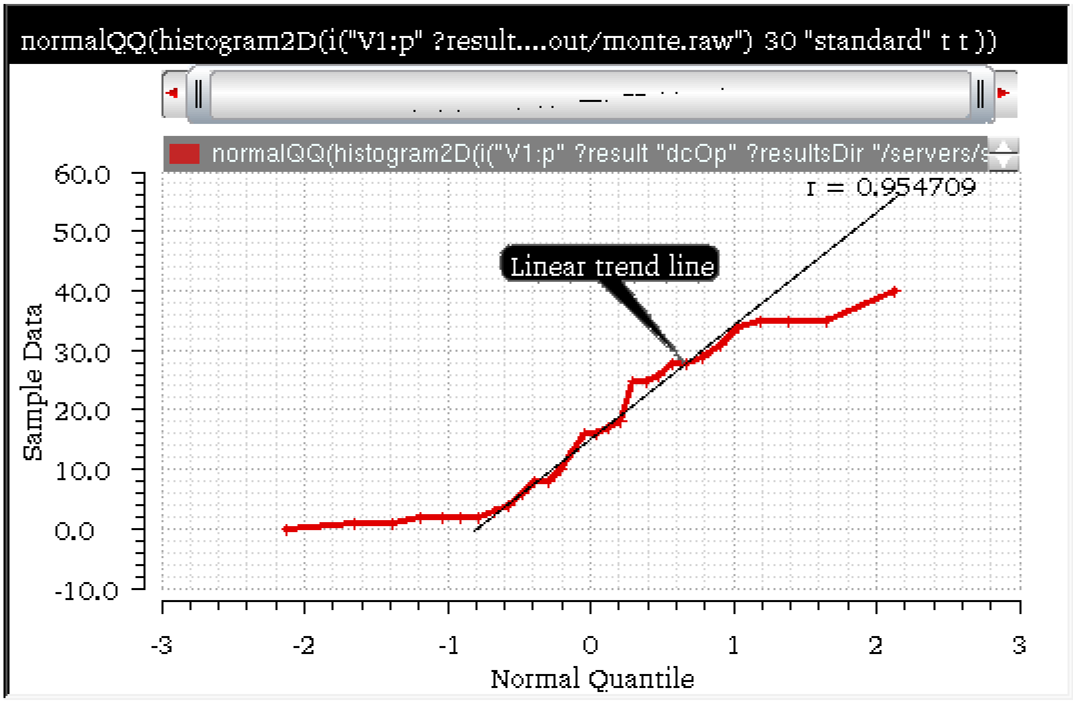
When you evaluate this expression, the following output waveform representing the data in histogram is displayed in a new graph window.



Now, apply the `normalQQ` function on the expression created for the `histogram2D` function. When you select normalQQ function from the Function Panel, the function is applied on the expression in the Buffer. The corresponding expression created for `normalQQ` is as follows:

```
normalQQ(histogram2D(i("V1:p" ?result "dcOp" ?resultsDir "./monte/
monte.out/monte.raw") 30 "standard" t t ))
```

When you evaluate this expression, the quantile-quantile wabeform is dispalyed in a new graph window, as shown in the figure below:



Related OCEAN Function

The equivalent OCEAN command for `normalQQ` is:

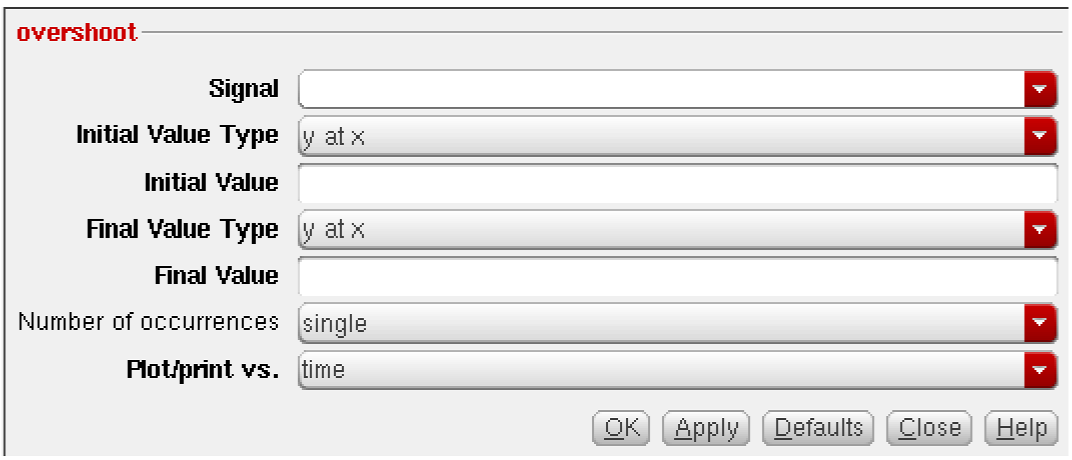
```
normalQQ( o_waveform )
     => o_waveform/nil
```

For more information, see `normalQQ` in *OCEAN Reference*.

overshoot

Returns the overshoot/undershoot of a signal as a percentage of the difference between initial and final values. Overshoot is calculated as:

$$OvershootOut = \frac{MaximumValue - FinalValue}{FinalValue - InitialValue}$$



This function includes the following arguments in the SKILL mode:

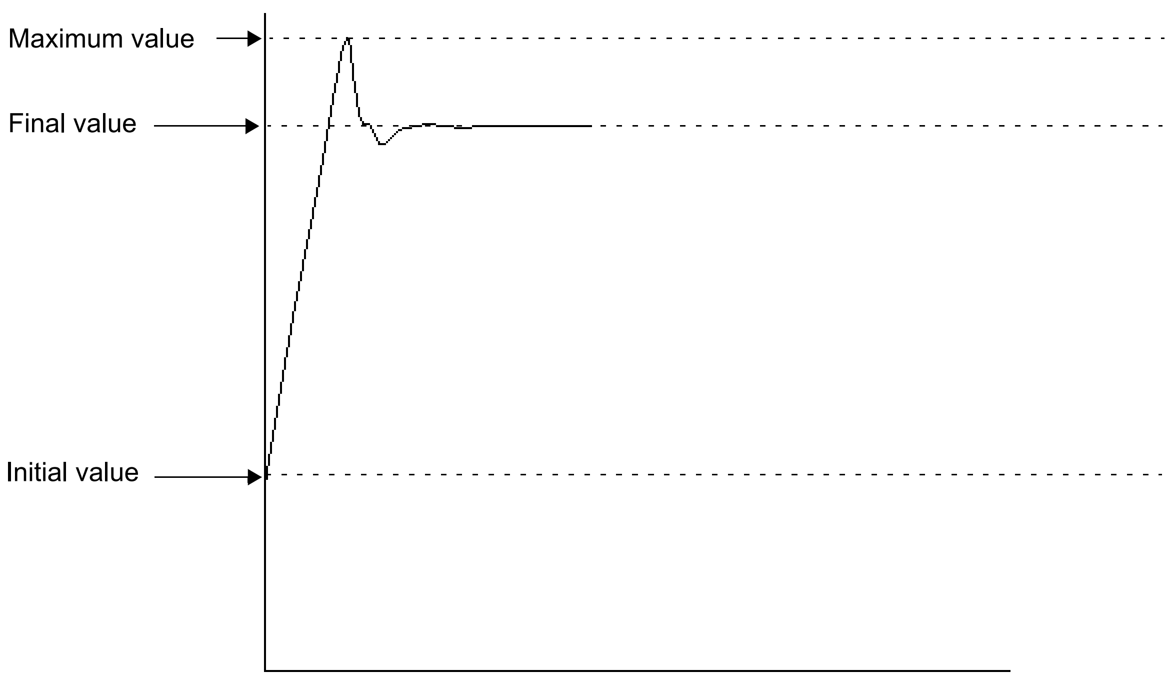
- *Signal*—Name of the signal.
- *Initial Value Type*—Specifies whether the initial value is the Y-axis value at the specified X-axis value (`y at x`) or Y-axis value (`y`).
- *Initial Value*—Initial value. To calculate the undershoot of a signal, the Initial Value should be higher than Final Value.
- *Final Value Type*—Specifies whether the final value is the Y-axis value at the specified X-axis value (`y at x`) or Y-axis value (`y`).
- *Final Value* is the final value.
- *Number of Occurences*—Specifies whether you want to retrieve only one occurrence of an overshoot event for the given waveform (*single*), or all occurrences of overshoot for the given waveform which you can later plot or print (*multiple*).
- *Plot/print vs*—Specifies whether you want to retrieve overshoot data against *time* (or another X-axis parameter for non-transient data) or *cycle*. Cycle numbers refer to the n'th occurrence of the delay event in the input waveform.
  The value in this field is ignored when you specify *Number of Occurences* as *single*.

In the MDL mode,

- `sig` is the name of the signal.
- `initval` is the initial value. To calculate the undershoot of a signal, the initval should be higher than finalval.
- `finalval` is the final value.
- `inittype` specifies whether the initial value is a time (`'x)` or voltage value (`'y)`.
- `finaltype` specifies whether the final value is a time (`'x)` or voltage value (`'y)`.

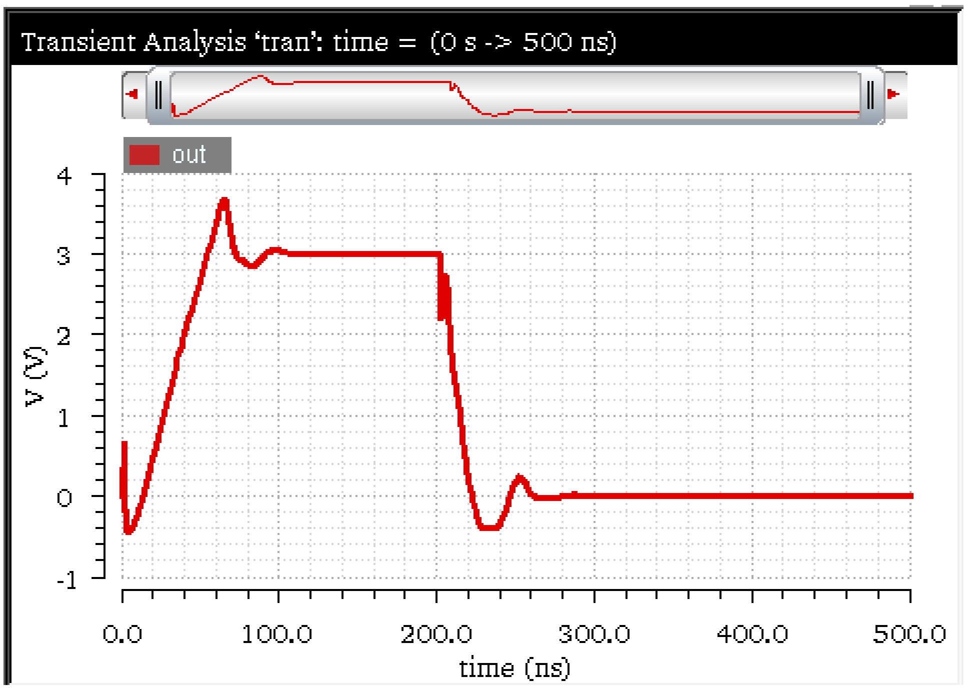
Example

The following diagram illustrates how the result is obtained with the values *signal*=`V(out)`, *Initial Value Type*=`y`, *Final Value Type*=`y`, *Initial Value*=`1`, and *Final Value*=`3`.



Example

This example shows the overshoot value generated when you apply the `overshoot` function on the following input waveform:



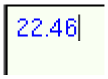
The following arguments are specified in this example:

- *Signal*—`v(out ?result "tran-tran")`
- *Initial Value Type*—`y`
- *Initial Value*—`0`
- *Final Value Type*—y
- *Final Value*—3.0
- *Number of occurrences*—single
- *Plot/print vs.*—time

The following expression is created in the Buffer:

```
overshoot(v(out ?result "tran-tran" ?resultsDir "ampsim.raw") 0 nil
3.0 nil nil "time")
```

When you evaluate this expression, the following output is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN command for `overshoot` is:

```
overshoot( o_waveform n_initVal g_initType n_finalVal g_finalType
     [g_multiple [s_Xname]][g_histoDisplay][x_noOfHistoBins] )
     => o_waveform/n_value/nil
```

For more information about this OCEAN function, see overshoot in *OCEAN Reference*.

pavg

Returns the periodic average of a family of signals for each time point.

This function includes the following arguments:

- *Signal*—Name of the signal for which you want to calculate the periodic average.
- *From*—Start time from where you want to calculate the periodic average.
- *To*—End time till you want to calculate the periodic average.
- *Period*—Difference between the end time and start time. This is an optional argument.
- *Sampling Factor*—Sampling factor, which can be increased to increase the accuracy of the output. Default value is `1`.

Additional Information

The periodic average of an input waveform is calculated by using the following equations:

```
pavg (signal from to period (sfactor 1) )
```

```
pavg(tk) = average(sample( o_waveform(t), from tk, to N*T, linear, by
T))
```

where,

`o_waveform(t)` is the input waveform

`N` = `floor((to-from) / T)`

`T` = period

If the input waveform is a multi-dimensional waveform or a family of waveforms, the output is calculated as:

$$pavg(tk) = 1/M \sum_{j=0}^{M-1} pavg(tk)[o\_waveform\_family[j]]$$

where,

`o_waveform_family` is the input waveform family

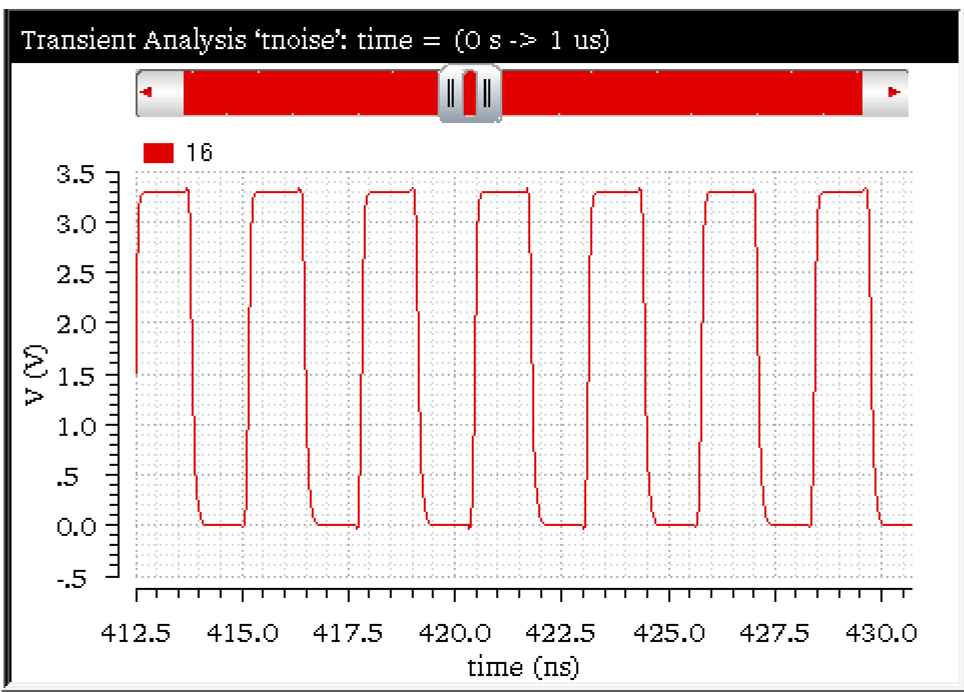
`M` is the number of leaf waveforms in the family.

`o_waveform_family[j]` is the `j`th leaf waveform of the family of waveform represented by the input waveform.

It calculates the `pavg` on each leaf, and then averages over the number of leaves in the family of waveform. In this case the resultant is not a family of waveforms, but a normal waveform of dimension 1.

Example

This example shows the output plot generated when you apply the `pavg` function on the following input waveform:



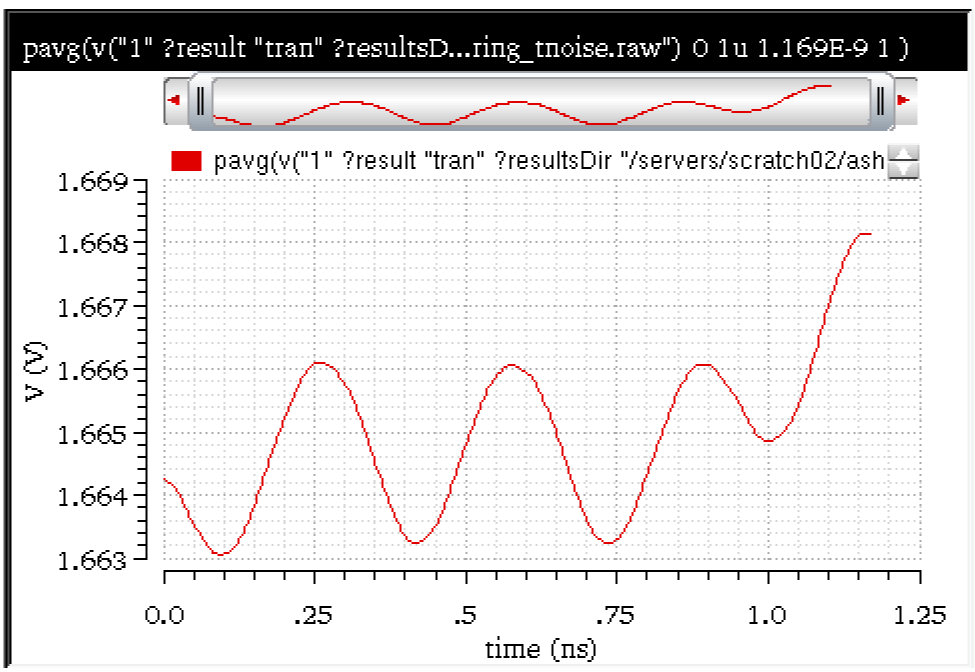
The following arguments are specified in this example:

- *Signal*—`v("16" ?result "tran")`
- *From*—`0`
- *To*—`1u`
- *Period*—`1.169E-9`
- *Sampling Factor*—`1`

The following expression is created in the Buffer:

```
pavg(v("16" ?result "tran" ?resultsDir "./nand2_ring_tnoise.raw") 0
1u 1.169E-9 1 )
```

Now, when you evaluate this expression, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `pavg` is:

```
pavg( o_waveform n_from n_to [n_period [n_sfactor]])
     => o_waveform/nil
```

For more information about this OCEAN function, see pavg in *OCEAN Reference*.

peak

Detects the peaks in the input waveform and returns the X and Y coordinates of these peak points in the form of a waveform. The peak function will not work for waveforms that comprise of complex numbers.

This function is available only in the SKILL mode.

- *Signal*—Name of the signal.
- *From*—Initial point on the specified waveform to start determining the peaks. By default, the first point of the waveform is the starting point.
- *To*—Final point on the specified waveform up to which the peaks are to be determined. By default, the last point of the waveform is the end point.
- *X-Tolerance*—Distance on the X-axis within which all peaks are to be filtered. The default value is `0.0`.
- *Y-tolerance*—Distance on the Y-axis within which all peaks are to be filtered. The default value is `0.0`.

If only the *X-Tolerance* is specified, the peaks are filtered in the X-direction.

If only the *Y-Tolerance* is specified, the peaks are filtered in the Y-axis direction.

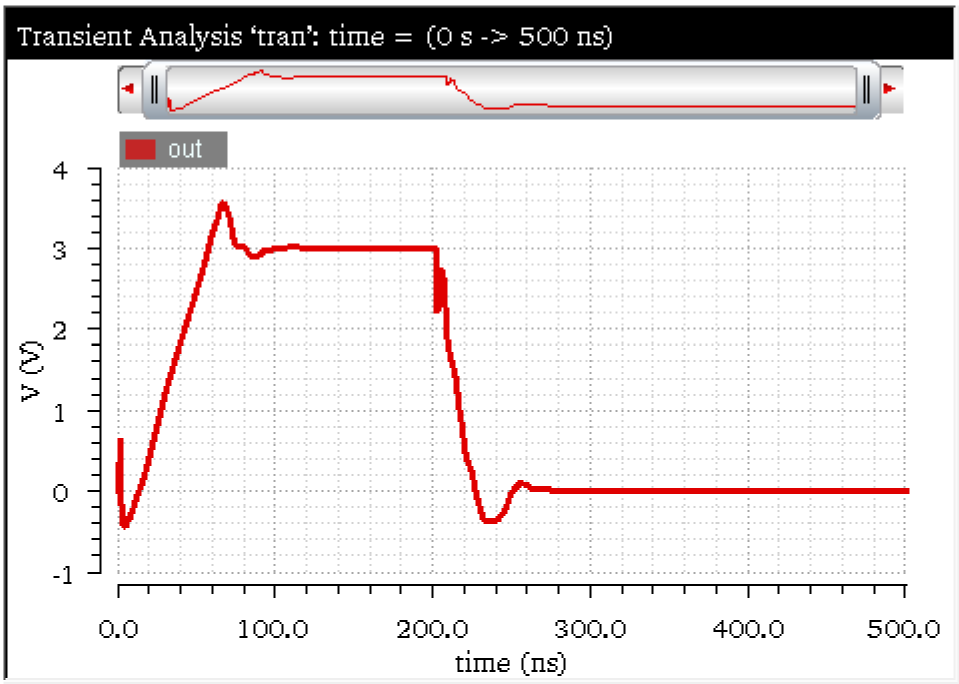
If both *X-Tolerance* and *Y-Tolerance* are specified, the filtering mechanism operates as follows:

1. The maximum peak is selected first.
2. All adjacent peaks in the neighborhood of both *X-Tolerance* in the X-axis direction and *Y-Tolerance* in the Y-axis direction are then filtered.

3. All the peaks in the rectangular window thus formed are filtered based on both *X-Tolerance* and *Y-Tolerance*.

Example

This example shows the output waveform generated when you apply the `peak` function on the following input signal:



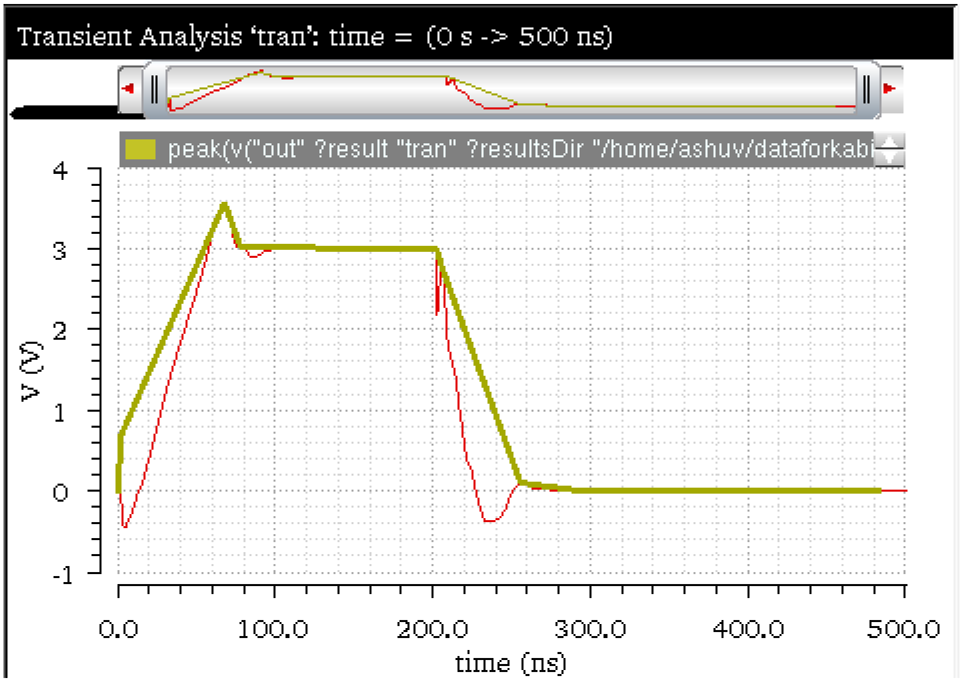
The following arguments are specified in this example:

- *Signal*—`v("out" ?result "tran")`
- *From*—`0`
- *To*—`500n`
- *X-Tolerance*—`0`
- *Y-Tolerance*—`0`

The following expression is created in the Buffer:

```
peak(v("out" ?result "tran" ?resultsDir "./ampsim.raw") ?from 0 ?to
500n ?xtol 0 ?ytol 0)
```

Now, when you evaluate this expression and plot the results in append mode, the following waveform is appended to the existing input waveform. This waveform shows the peak points of the input waveform.



Related OCEAN Function

The equivalent OCEAN command for `peak` is:

```
peak( o_waveform ?from f_from ?to f_to ?xtol f_xtol ?ytol f_ytol )
     => o_waveform/nil
```

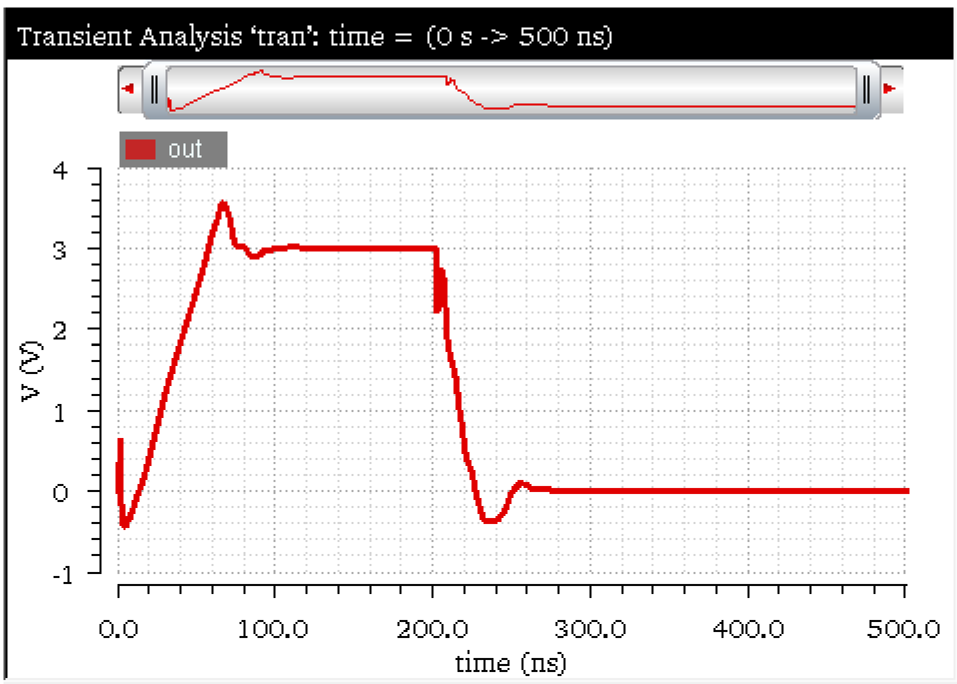
For more information about this OCEAN function, see peak in *OCEAN Reference*.

peakToPeak

Returns the difference between the highest and lowest values of a signal.

Example

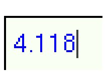
This example calculates the peak to peak value when you apply the `peakToPeak` function on the following input signal:



The following expression is created in the Buffer when you apply this function:

```
peakToPeak(v("out" ?result "tran" ?resultsDir "./ampsim.raw"))
```

Now, when you evaluate this expression, the following output value is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN command for `peakToPeak` is:

```
peakToPeak( o_waveform )
     => o_waveform/n_value/nil
```

For more information about this OCEAN function, see peakToPeak in *OCEAN Reference*.

period_jitter

Returns a waveform or a value representing the deviation from the average period.

This function includes the following arguments in the SKILL mode,

- *Waveform* is the name of the signal, expression, or family of waveforms.
- *Cross Type* is the points at which the curves of the waveform intersect with the threshold. While intersecting, the curve may be either rising (`rising`) or falling (`falling`).
- *mode* specifies whether the threshold value is to be calculated by the Virtuoso Visualization and Analysis XL (`auto`) or specified by you (`user`). The auto threshold is calculated as:

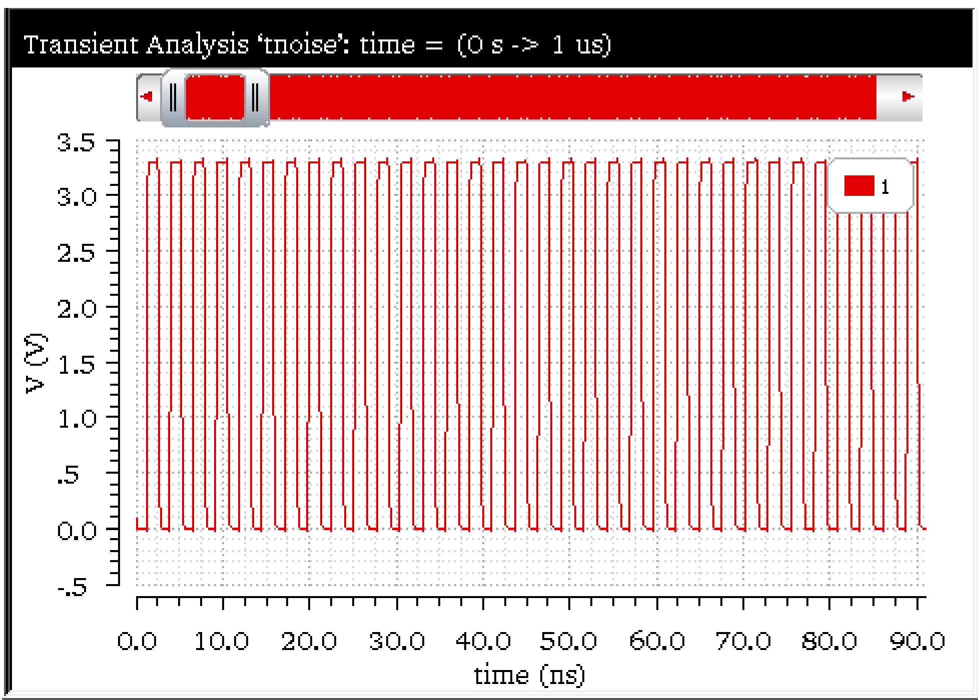
  Auto Threshold Value = integral of the waveform divided by the X range.
- *Threshold* is the threshold value against which the period is to be calculated. You need to specify the threshold value only if the *mode* is `user`.
- *Bin Size* is the width of the moving average window. The deviation of value at the particular point from the average of this window is the jitter.
  If *binsize*=0, all periods are used to calculate the average.
  If *binsize*=*N*, the pervious *N* periods are used to calculate the average.
- *Plot/print vs.* specifies whether you want to retrieve the period jitter against *time* (or another X-axis parameter for non-transient data) or *cycle*. Cycle numbers refer to the n'th occurrence of the delay event in the input waveform.
- *Output Type* is the type of output. If set to `plot`, the output is a jitter waveform. If set to `sd`, the output is a standard deviation of the jitter waveform. The default value is `plot`.

In the MDL mode,

- *sig* is the name of the signal.
- *thresh* is the threshold Y-axis value defining the period of the signal.
- *dir* is the direction of the crossing event.
- *binsize* is the integer used to calculate the average period of the signal.
  If *binsize*=0, all periods are used to calculate the average.
  If *binsize*=*N*, the previous *N* periods are used to calculate the average.

Example

This example displays the period jitter waveform generated when you apply the `period_jitter` function on the following input waveform:



The following arguments are specified in this example:

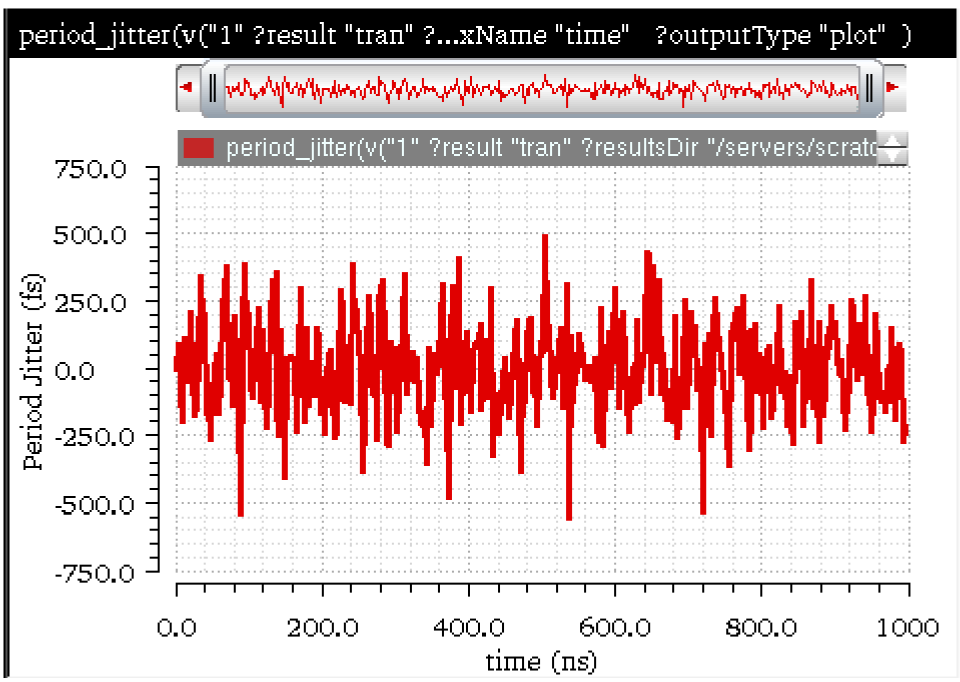
- *Waveform*—`v("1" ?result "tran”)`
- *Cross Type*—`rising`
- *mode*—`user`
- *Threshold*—`1.5`

- *Bin Size*—`30`
- *Plot/print vs.*—`time`
- *Output Type*—`plot`

The following expression is created in the Buffer:

```
period_jitter(v("1" ?result "tran" ?resultsDir "./
nand2_ring_tnoise.raw") "rising"  ?mode "user" ?threshold 1.5
?binSize 30 ?xName "time"   ?outputType "plot" )
```

Now, when you evaluate this expression and plot the results, the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `period_jitter` is:

```
period_jitter( o_waveform t_crossType
     [?mode t_mode] [?threshold n_threshold] [?binSize n_binSize]
     [?xName t_xName] [?outputType t_outputType] )
     => o_waveform/f_val/nil
```

For more information about this OCEAN function, see period_jitter in *OCEAN Reference*.

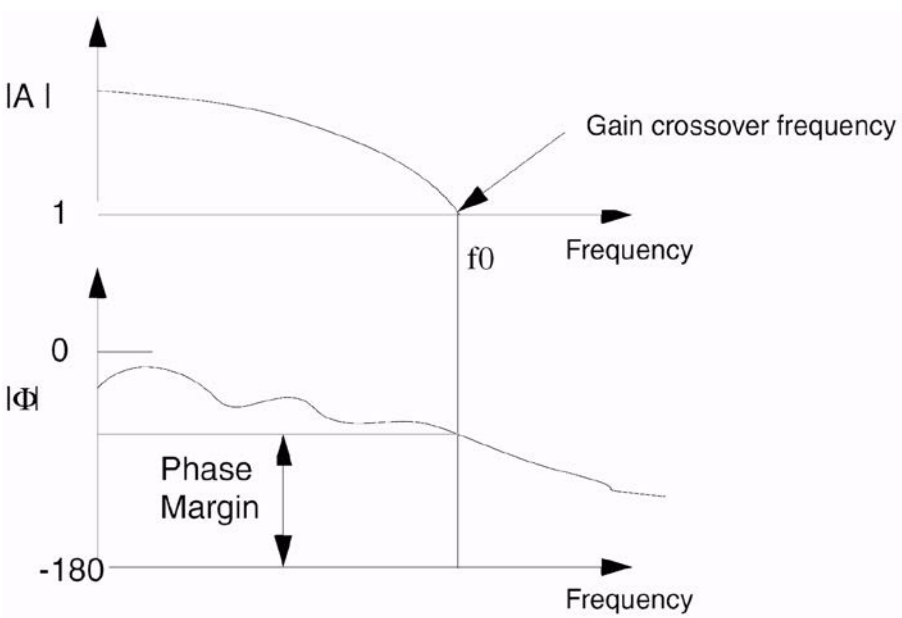
phaseMargin

Computes the phase margin of the loop gain of an amplifier. You supply a waveform representing the loop gain of interest over a sufficiently large frequency range.

$$phaseMargin(gain) = 180 + phase(value(gain\ f_0))$$

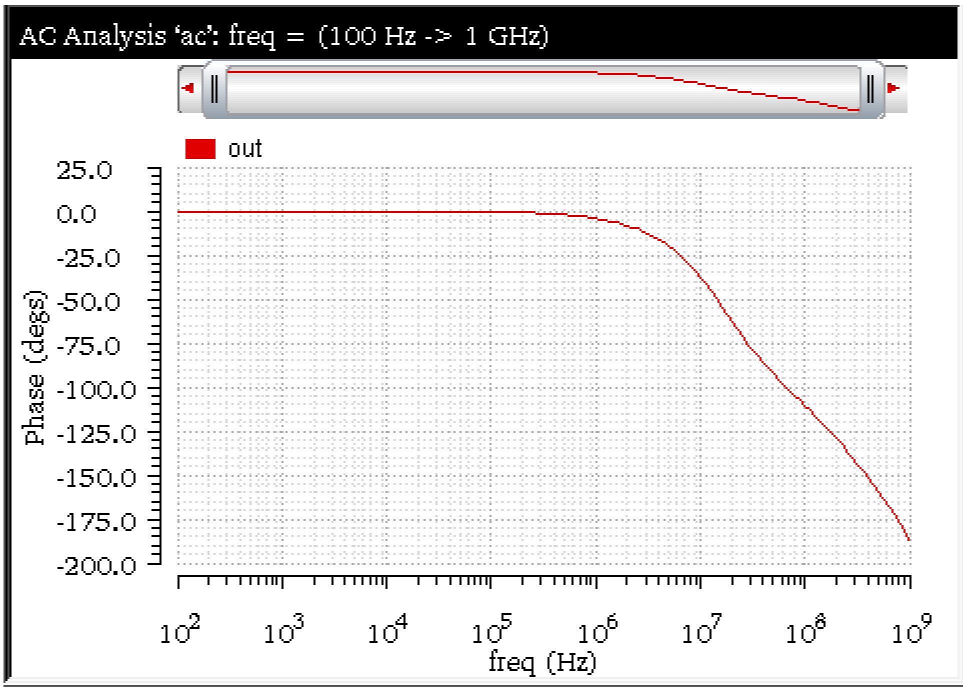
The phase margin is calculated as the difference between the phase of the gain in degrees at f0 and at -180 degrees. The frequency f0 is the smallest frequency where the gain is 1. For stability, the phase margin must be positive.

small-sig is the loop gain of interest over a sufficiently large frequency range.



Example

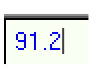
This example shows the phase margin value calculated when you apply the `phaseMargin` function on the following AC signal:



When you apply this function, the following expression is created in the Buffer:

```
phaseMargin(v("out" ?result "ac" ?resultsDir "/.ampTest.raw"))
```

Now, when you evaluate this expression, the following output value is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN command for `phaseMargin` is:

```
phaseMargin( o_waveform )
     => o_waveform/n_value/nil
```

For more information about this OCEAN function, see phaseMargin in *OCEAN Reference*.

phaseNoise

Plots the phase noise waveform for noise analysis results.

This function is available only in the SKILL mode.

This function includes the following arguments:

- *Harmonic*—Specify a harmonic number.
- *Signal dataset*—Select the signal data set from the drop-down list.
  Available options: `pss_fd`, `hb_fd`, `hb_mt_fi`, and `qpss_fi`
- *Noise dataset*—Select the noise dataset from the drop-down list.
  Available options: `pnoise`, `pnoise_corr`, `pnoise_src`, `pnoise_xfersrc`, `hbnoise`, `hbnoise_xf`, `hbnoise_mt_xf`, `pnoise_hbnoise`, `qpnoise`, and `qpnoise_hbnoise`

Important

You need to set the database context to the `pss` results directory for which you want to plot the phase noise. For more information about how to set a results directory as in-context results directory, see Changing In-Context Results Directory on page 42.

Example

This example shows the output generated when you apply the `phaseNoise` function on the `./osc13.raw/pss_fd` results database.

Firstly, ensure that the database context is set to the `pss_fd` results directory.

The following arguments are specified in this example:

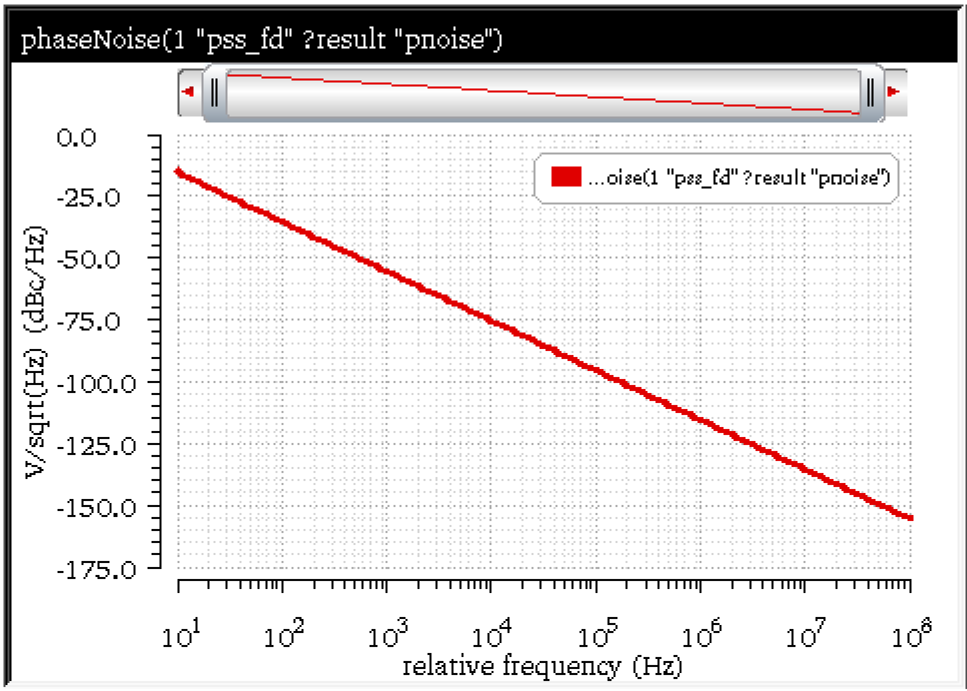
- *Harmonic number*—`1`
- *Signal dataset*—`pss_fd`

- *Noise dataset*—`pnoise`

The following expression is created in the Buffer:

```
phaseNoise(1 "pss_fd" ?result "pnoise")
```

Now, when you evaluate this expression, the following waveform is displayed in the graph window.



Related OCEAN Function

The equivalent OCEAN command for `phaseNoise` is:

```
phaseNoise( g_harmonic S_signalResultName
     [?result s_noiseResultName [?resultsDir t_resultsDir]] )
     => o_waveform/nil
```

For more information, see `phaseNoise` in *OCEAN Reference*.

PN

Returns a waveform for the transient phase noise of the input waveforms in decibels (`dBc/Hz`). Phase noise is defined as the power spectral density of the absolute jitter of an input waveform. This function is available only in the SKILL mode.

This function includes the following arguments:

- *Signal*—Name of the waveform, expression, or family of waveforms. The expression displayed in the Buffer can be added to this field by selecting *buffer*.
- *Cross Type*—Points at which the curves of the waveform intersect with the threshold. While intersecting, the curve may be either rising (`rising`) or falling (`falling`). The default value is `rising`.
- *Threshold*—Threshold value against which the phase noise is to be calculated.
- *Tnom*—Nominal time period of the input waveform.
  Default value: `nil`.
- *Window Type*—Window you want to use. The following window types are supported—*Blakman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hanning, Kaiser, Parzen*, *Rectangular,* and *Triangular.* The default window type is *Rectangular*.

- *Smoothing Factor*—Smoothing factor applicable to only Kaiser window. The *Smoothing Factor* field accepts values from `0` to `15`. The value `0` implies no smoothing and is equivalent to a rectangular window. The default value is `1`.
- *Window Size*—Number of frequency domain points to be used while calculating the power spectral density. A larger window size results in an expected operation over fewer samples, which can result in larger variations in the phase noise. A small window size can smear out sharp steps in the phase noise that might really be present. Default value is `256`. For more information, see psd on page 768.
- *Detrending Mode*—Determines the expected trend for the underlying data while calculating the power spectral density. The default value is `None`. For more information, see psd on page 768.
- *Coherent Gain*—The coherent gain of a window is the zero frequency gain (or the DC gain) of the window. It is calculated by normalizing the maximum amplitude of the window to one and then summing the values of the window amplitudes over the duration of the window. The result is then divided by the length of the window (that is, the number of samples).
  When you do a dft, the applied Window Type (aside from the rectangular type) changes the signal amplitude. Applying a coherent gain factor is a way to get consistent results regardless of the window type.
  Valid Values: `none`, `default`, `magnitude`, `dB20`, or `dB10`
  Default value: `default`
- *Coherent Gain Factor*—This is the scaling factor. A non-zero value that scales the power spectral density by 1/(Coherent Gain).
  Valid values : 0 < coherent_gain_factor < 1. You can use 1 if you do not want this scaling factor to be used.
  Default value : `0`.
- *Method Type*—Determines the algorithm you want to use to calculate the phase noise, which can be `Absolute Jitter Method` or `Direct Power Spectral Density Method`.
  Default value: `Absolute Jitter Method`.

Note: The *Window Type, Smoothing Factor, Window Size, Detrending Mode*, and *Coherent Gain Factor*, *Coherent Gain* arguments are used to calculate the power spectral density of the absolute jitter to obtain the phase noise.

Defining the Phase Noise

For a given waveform $v(t), t_{start} \leq t \leq t_{end}$, that has the following properties:

- Oscillating with expected nominal period, `T`

- Between minimum and maximum values, $v_{min} \le v(t) \le v_{max}$,
- Rising and falling through a given threshold, v_{th}
- In time intervals, $t_k ; 0 \le k \le N$

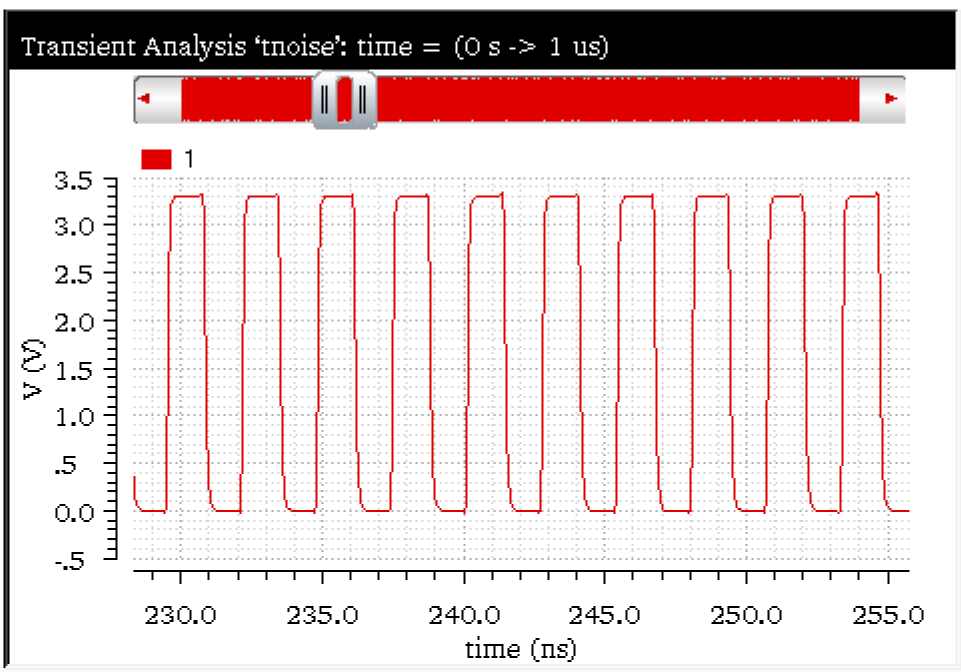
The phase noise, `P(f)`, can be represented as the spectral density (in decibels) of absolute jitter in phase units as a function of reference clock time:

$$P(f) = PSD\left\{\frac{J_a(k \cdot T)}{T}\right\}$$

where, `PSD` is the power spectral density.

Example

Consider the following input signal from the transient analysis:



The PN function is applied on this input waveform with the following arguments:

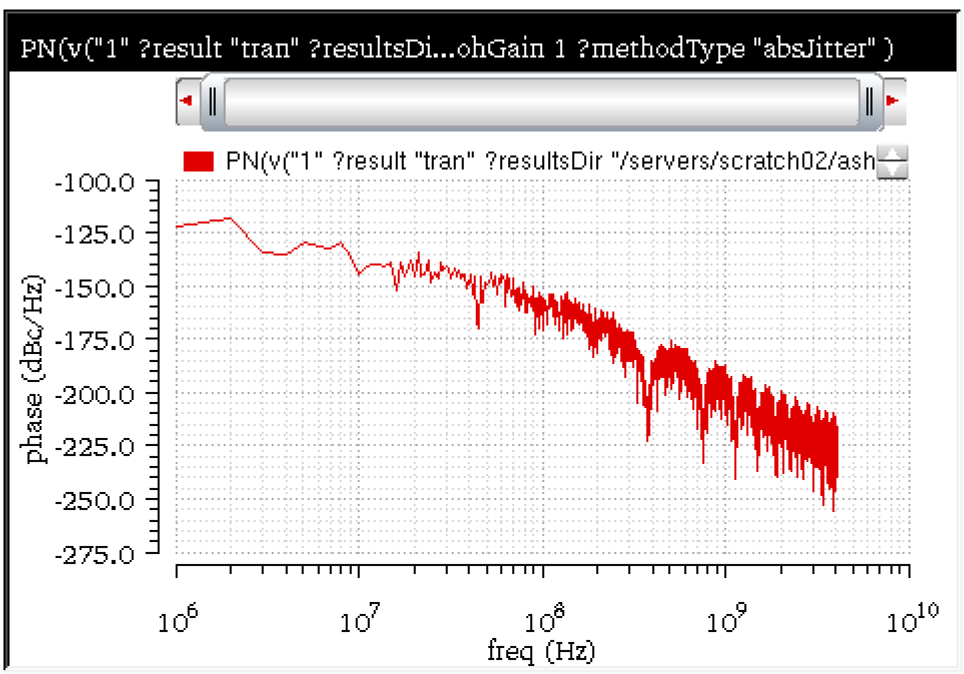
- *Signal*=`v("1" ?result "tran" ?resultsDir "nand2_ring_tnoise.raw")`
- *Cross Type*=`rising`
- *Threshold*=`1.0`

- *Tnom*=`nil`
- *Window Type*=`Rectangular`
- *Smoothing Factor*=`1`
- *Window Size*=`8192`
- *Detrending Mode*=`None`
- *Coherent Gain*=`(magnitde)`
- *Coherent Gain Factor*=`1`
- *Method Type*=`Absolute Jitter Method`

The corresponding expression created in the Buffer is as follows:

```
PN(v("out" ?result "tran" ?resultsDir “nand2_ring_tnoise.raw”)
"rising" 1.0 ?Tnom nil ?windowName "Rectangular" ?smooth 1
?windowSize 8192 ?detrending "None" ?cohGain 1 ?methodType
"absJitter" )
```

When you evaluate this expression, the following output waveform showing the transient phase noise is displayed in a new graph window:



Related OCEAN Function

The equivalent OCEAN command for `PN` is:

```
PN( o_waveform t_crossType n_threshold ?Tnom n_tnom ?windowName t_windowName
     ?smooth x_smooth ?windowsize x_windowsize ?detrending t_detrending)
     ?cohGain f_cohGain ?methodType t_methodType)
     => o_waveform/nil
```

For more information, see `PN` in *OCEAN Reference*.

pow

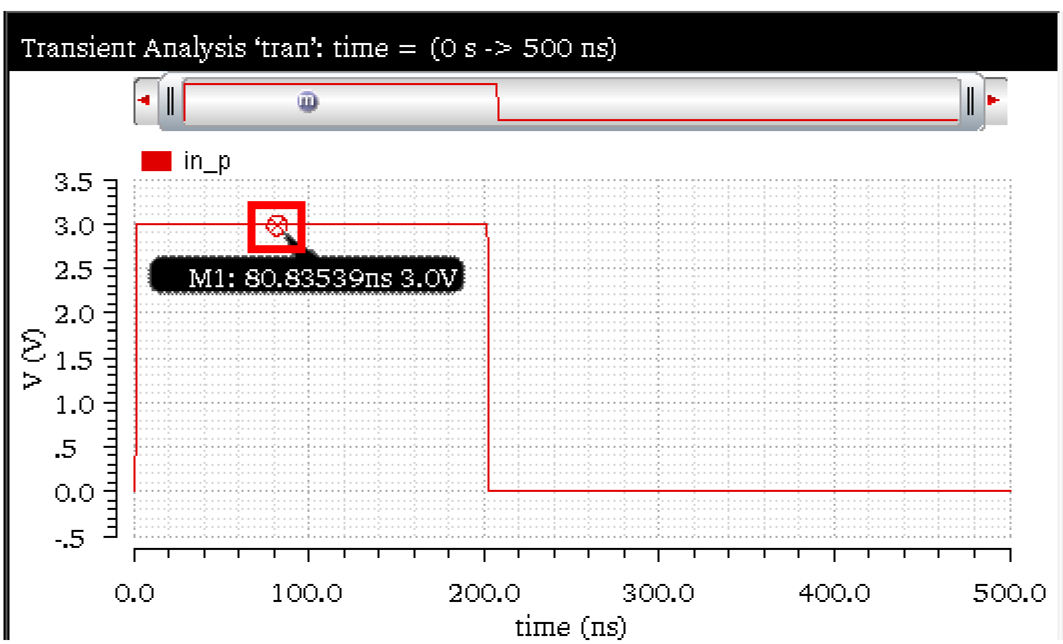
Returns the value of base raised to the power of exponent ($base^{exponent}$).

This function includes the following arguments:

- *Base*—Name of the signal
- *Exponent*—Power value, which indicates how many times you want to multiply the signal. The default value is `1`.

Example

Consider the following input signal from the transient analysis. Note that a point marker is placed on the input waveform at the voltage value `3.0V` on Y-axis.



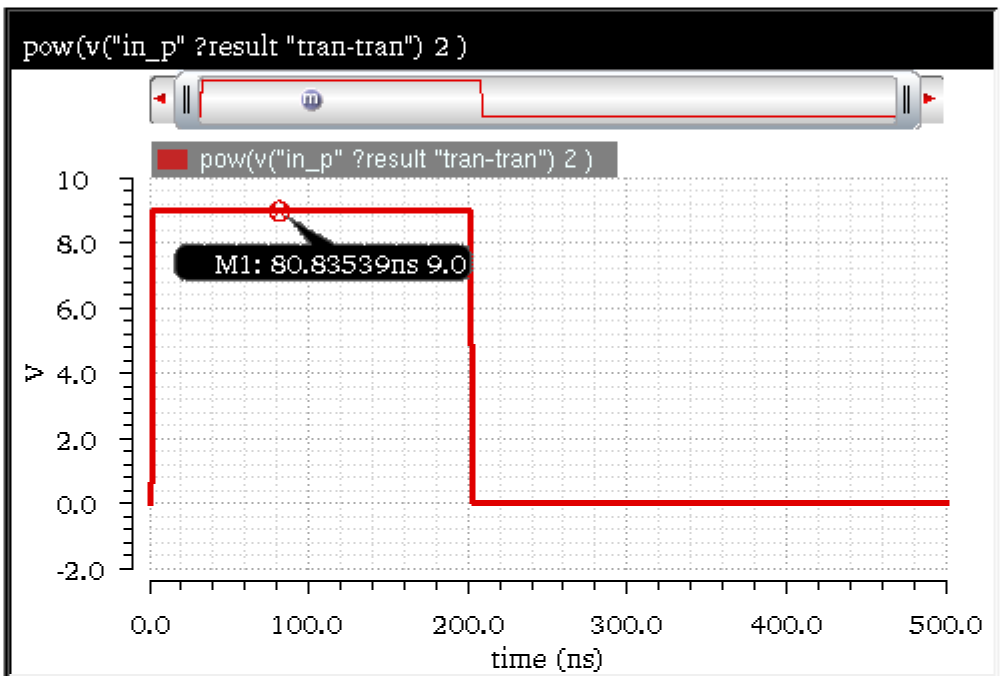
The `pow` function is applied to this input signal with the following arguments:

- *Base*—`v("in_p" ?result "tran-tran" ?resultsDir "./ampsim.raw")`
- *Exponent*—`2`

The corresponding expression created in the Buffer is as follows:

```
pow(v("in_p" ?result "tran-tran" ?resultsDir "./ampsim.raw")2)
```

When you evaluate this expression, the power is calculated and the following output waveform is displayed in a new graph window. Note that the voltage value for the point marker has been changed to `9.0V` in the output waveform, which represents the `pow(3,2)` calculation.



Below are additional examples to indicate the power calculation:

- `pow( average( v( "/net9" ) ) 0.5 )`

  Returns the square root of the average value of the voltage at `"/net9"`.

- `pow( 2 3 )=> 8`

  Returns the value of `2` to the third power, or `8`.

- `pow( -2 2 )=> 4`

  Returns the value of `-2` to the second power.

- `pow( 2.5 -1.2 )=> 0.3330213`

Returns the value of `2.5` to the power of `-1.2`

Related OCEAN Function

The equivalent OCEAN command for `pow` is:

```
pow( {o_waveformBase | n_numberBas} {o_waveformExpn | n_numberExpn} )
     => o_waveform/n_result/nil
```

For more information, see `pow` in *OCEAN Reference*.

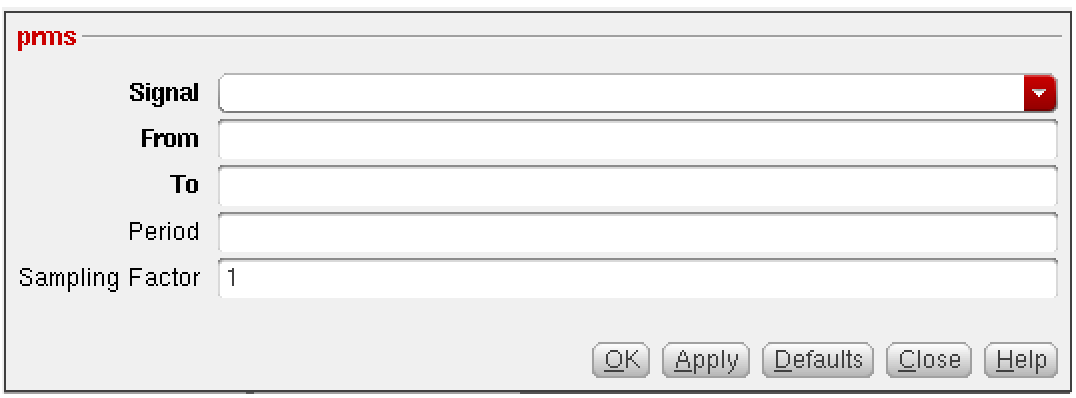
prms

(Periodic Root Mean Square)

Returns the periodic root mean square of a family of signals for each time point, which is the square root of the periodic average of the square of input waveform and is represented as follows:

```
prms(o_waveform n_from n_to n_period n_sfactor) =
sqrt(pavg(o_waveform*o_waveform) from to period sfactor)
```

For more information about the `pavg` function, see pavg on page 741.



This function includes the following arguments:

- *Signal*—Name of the signal for which you want to calculate the root mean square.
- *From*—Start time from where you want to calculate the periodic root mean square.
- *To*—End time till you want to calculate the periodic root mean square.
- *Period*—Difference between the end time and start time.
- *Sampling Factor*—Controls the output accuracy, You can increase the sampling factor to increase the accuracy in the output.
  Default value is `1`.

Additional Information

The periodic root mean square is represented by the following equation:

```
prms(t)= sqrt( pavg( fam(t)**2  start_time end_time period sfactor))
```

Here,

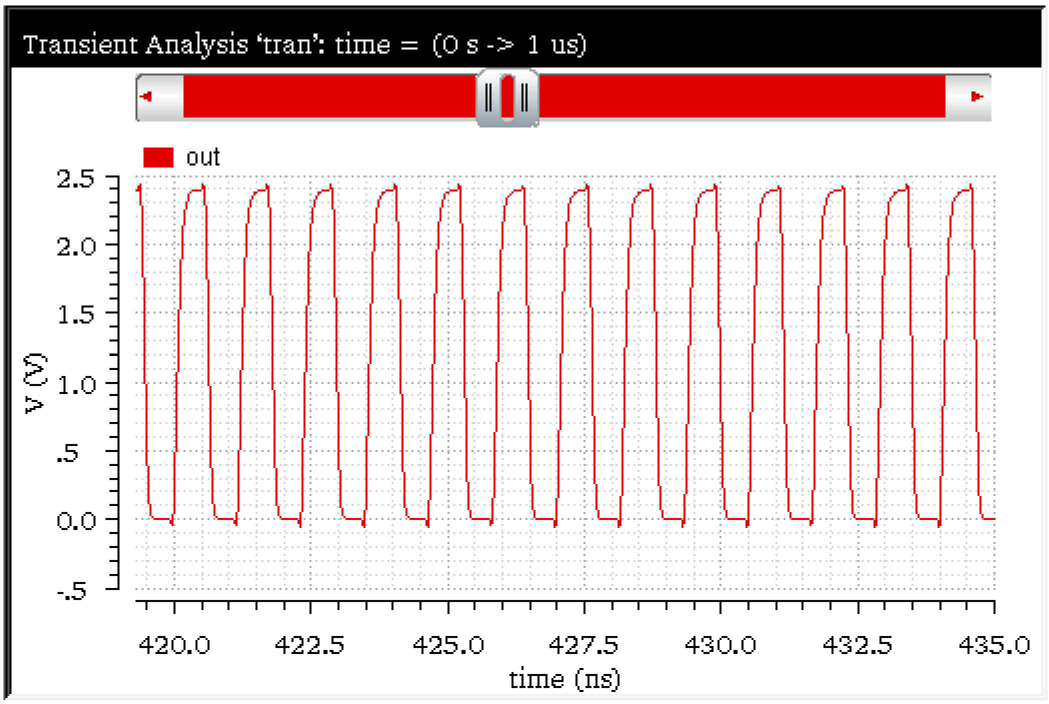
`pavg` is the periodic average

`fam(t)` is the family of `t` waveforms

`period=end_time-start_time`

Example

Consider the following input waveform from the transient analysis:



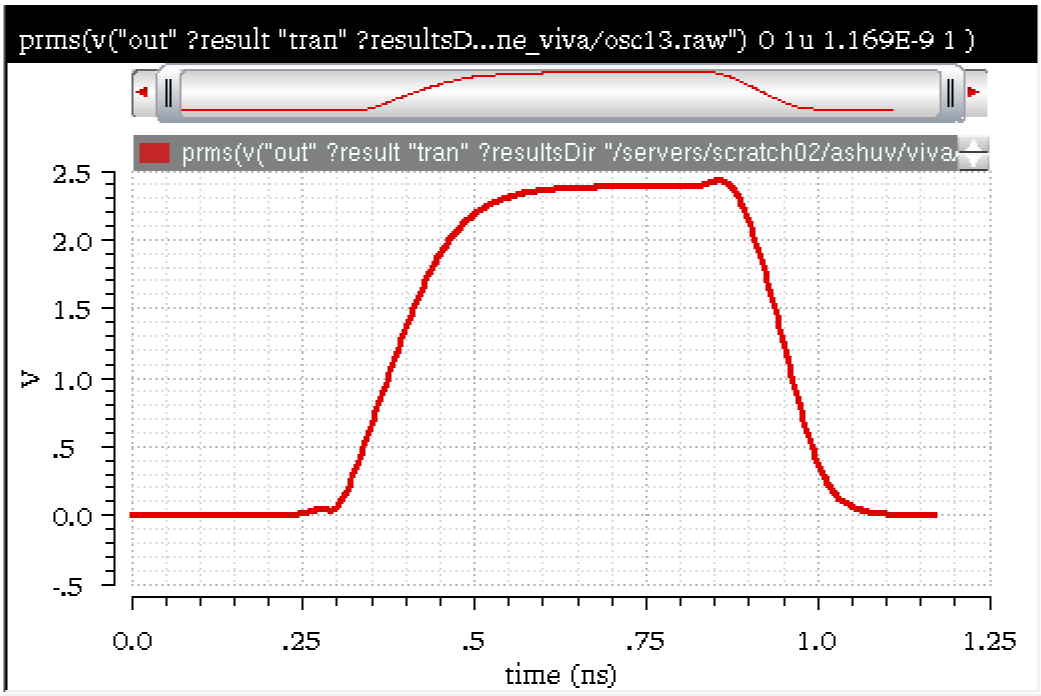
The `prms` function is applied on this input waveform with the following arguments:

- *Signal*—`v("out" ?result "tran" ?resultsDir "./osc13.raw")`
- *From*—`0`
- *To*—`1u`
- *Period*—`1.169E-9`
- *Sampling Factor*—`1`

The corresponding expression created in the Buffer is as follows:

```
prms(v("out" ?result "tran" ?resultsDir "./osc13.raw") 0 1u 1.169E-9
1 )
```

When you evaluate this expression, the following output waveform is displayed in a new graph window to represent the periodic root mean square values:



Related OCEAN Function

The equivalent OCEAN command for `prms` is:

```
prms( o_waveform n_from n_to [n_period [n_sfactor]])
     => o_waveform/nil
```

For more information, see `prms` in *OCEAN Reference*.

psd

Power Spectral Density

Describes how the power (or variance) of a time series (signal) is distributed with frequency. Mathematically, it is defined as the Fourier Transform of the auto correlation sequence of the time series (signal). The waveform is first interpolated to generate evenly spaced data points in time. The spacing of the data points is the inverse of the *dft* sampling frequency. The *psd* is computed by first breaking up the time interval into overlapping segments. Each segment is multiplied, time point by time point, by the specified windowing function. The *dft* is performed on each windowed segment of the baseband waveform. At each frequency, the *dft*s from all segments are averaged together and the squared modulus of these averages gives the *psd*.

This function is available only in the SKILL mode and includes the following arguments:

- *Signal*—Name of the signal.
- *From*—Starting time for the spectral analysis interval.
- *To*—Ending time for the spectral analysis interval.
- *Number* of Samples—Number of time domain points to be used.
  Default value: `512`
- *Window Type*—Window you want to use.
  Default value: `Hanning`
  Valid values: `Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine,`

`HalfCycleSine3, HalfCycleSine6, Hanning, Kaiser, Parzen,Rectangular, and Triangular`
If you select *Kaiser*, type in a value for the Kaiser smoothing factor. The smoothing factor must be in the range 0 <= factor <= 15, where 0 is the same as using a rectangular window.

- *Smoothing Factor*—Controls the output accuracy. This argument applies only to the *Kaiser* window type.
  Default value: `1`
  It is applicable to the Kaiser window only. The Smoothing Factor accepts values from `0` to `15`. The value `0` implies no smoothing and is equivalent to a rectangular window.

- *Window Size*—Number of frequency domain points to use in the Fourier analysis.
  A larger window size results in an expectation operation over fewer samples, which leads to larger variations in the power spectral density. A small window size can smear out sharp steps in the power spectral density that might really be present.
  Default value: `256`

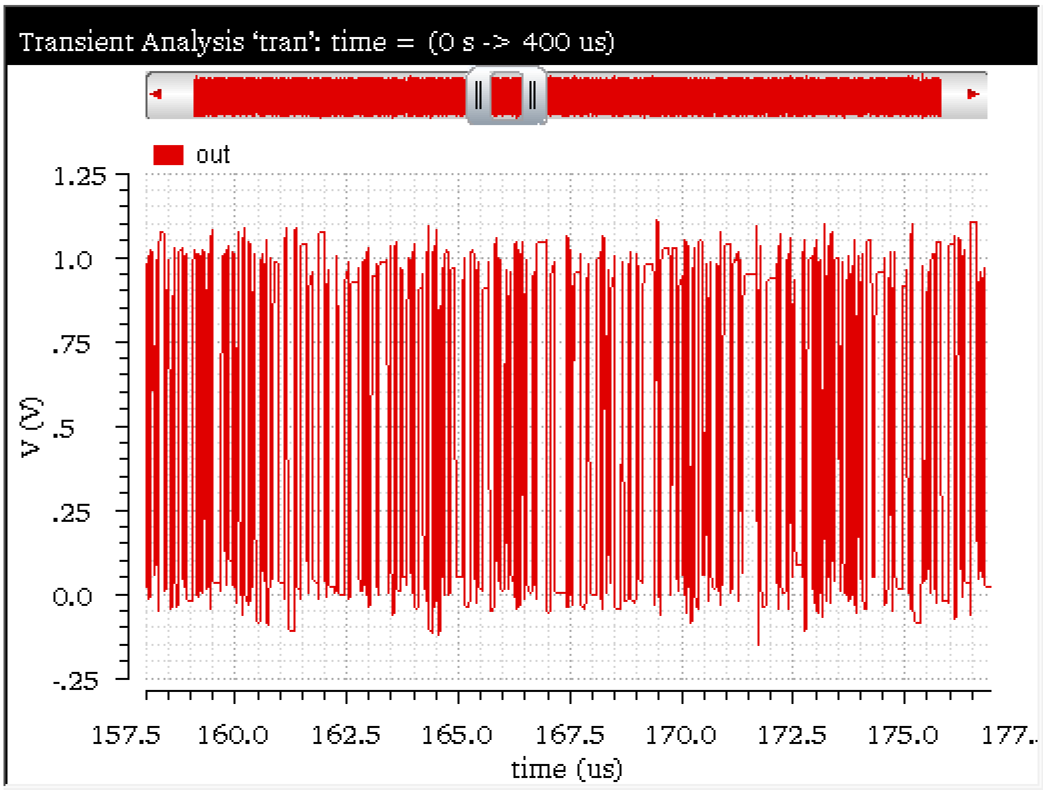
- *Detrending Mode*—Specifies the trend mode.
  Valid values: `Linear`, `Mean`, `None`
  Default value: `None`
  The `psd` function works by applying a moving windowed FFT to time-series data. If there is a deterministic trend to the underlying data, you may want to remove the trend before performing the spectral analysis. For example, consider analyzing phase noise in a VCO model. Without the noise, the phase increases more or less linearly with time, so it is appropriate to set the detrending mode to `Linear`. To subtract an average value, set the detrending mode to `Mean`. Where the spectrum of raw data is desired, set the detrending mode to `None`.

- *Coherent Gain*—The coherent gain of a window is the zero frequency gain (or the DC gain) of the window. It is calculated by normalizing the maximum amplitude of the window to one and then summing the values of the window amplitudes over the duration of the window. The result is then divided by the length of the window (that is, the number of samples).
  When you do a dft, the applied Window Type (aside from the rectangular type) changes the signal amplitude. Applying a coherent gain factor is a way to get consistent results regardless of the window type.
  Valid Values: `none`, `default`, `magnitude`, `dB20`, or `dB10`
  Default value: `default`

- *Coherent Gain Factor*—This is the scaling factor. A non-zero value that scales the power spectral density by 1/(Coherent Gain).
  Valid values : 0 < coherent_gain_factor < 1. You can use 1 if you do not want this scaling factor to be used.
  Default value : `0`

Example

Consider the following input signal from the transient analysis:

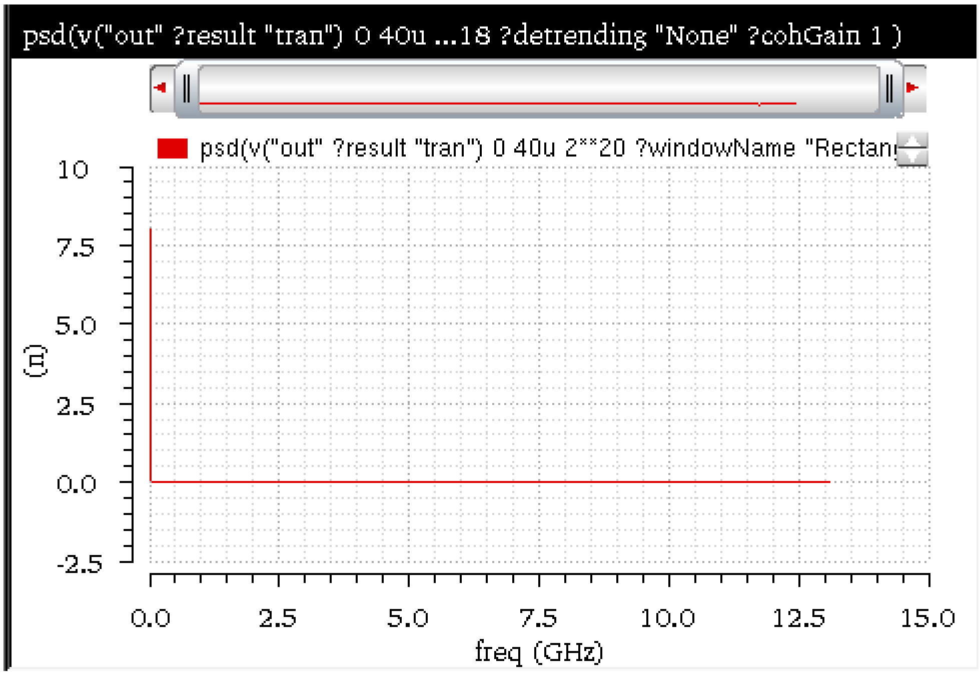


The `psd` function is applied on this input waveform with the following arguments:

- *Signal*—`v("out" ?result "tran" ?resultsDir "./prbs_sim.raw")`
- *From*—`0`
- *To*—`40u`
- *Number of Samples*—`2**20`
- *Window Type*—`Rectangular`
- *Smoothing Factor*—`1`
- *Window Size*—`2**18`
- *Detrending Mode*—`None`
- *Coherent Gain*—`(none)`
- *Coherent Gain Factor*—`1`

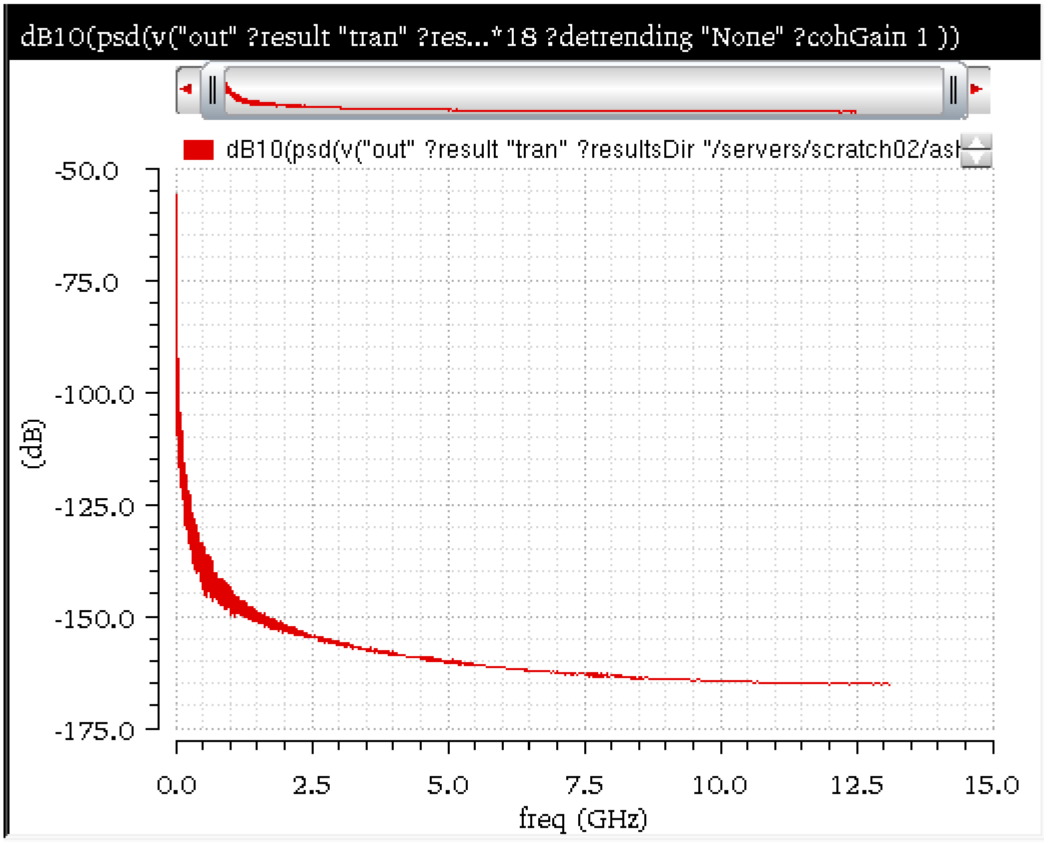
The corresponding expression created in the Buffer is as follows:

```
psd(v("out" ?result "tran" ?resultsDir "./prbs_sim.raw") 0 40u 2**20
?windowName "Rectangular" ?smooth 1 ?windowSize 2**18 ?detrending
"None" ?cohGain 1 )
```



Then, the `dB20` function is applied on this expression to modify the power spectral density calculation.

When you evaluate this expression, the power spectral density is calculated and the spectrum is displayed, as shown in the following output waveform:



Related OCEAN Function

The equivalent OCEAN command for `psd` is:

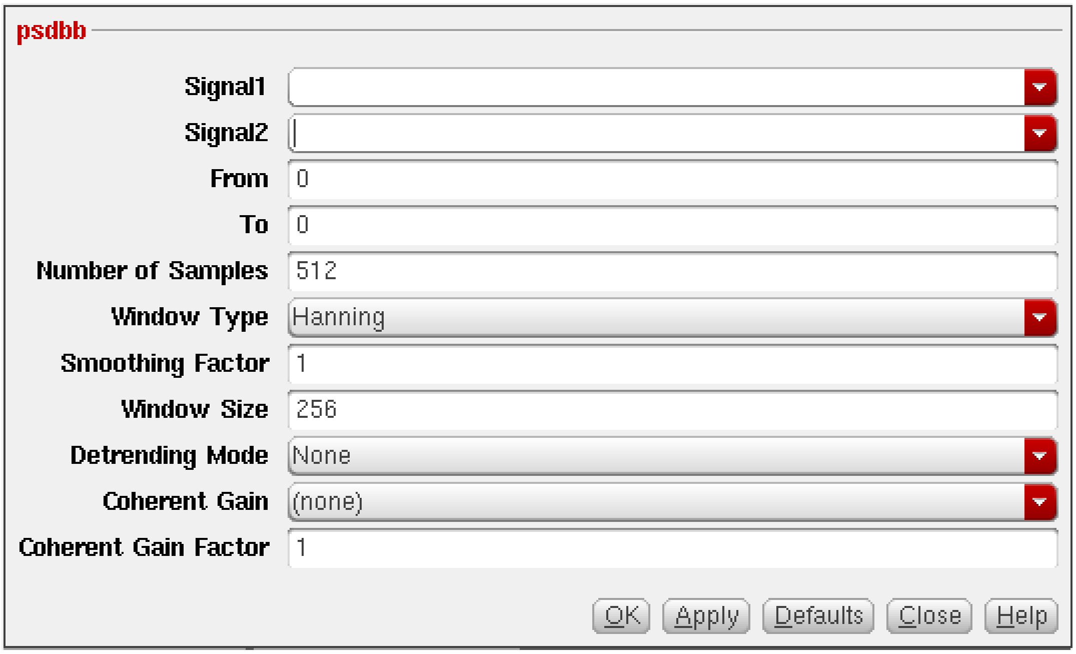
```
psd( o_waveform f_timeStart f_timeEnd x_num
     ?windowName t_windowName ?smooth x_smooth ?cohGain f_cohGain
     ?windowsize x_windowsize ?detrending t_detrending)
     => o_waveformReal/nil
```

For more information, see `psd` in *OCEAN Reference*.

psdbb

(Power Spectral Density Baseband)

Returns an estimate for the power spectral density of a `waveform1+j * waveform2`. This function is available only in the SKILL mode.



This function includes the following arguments:

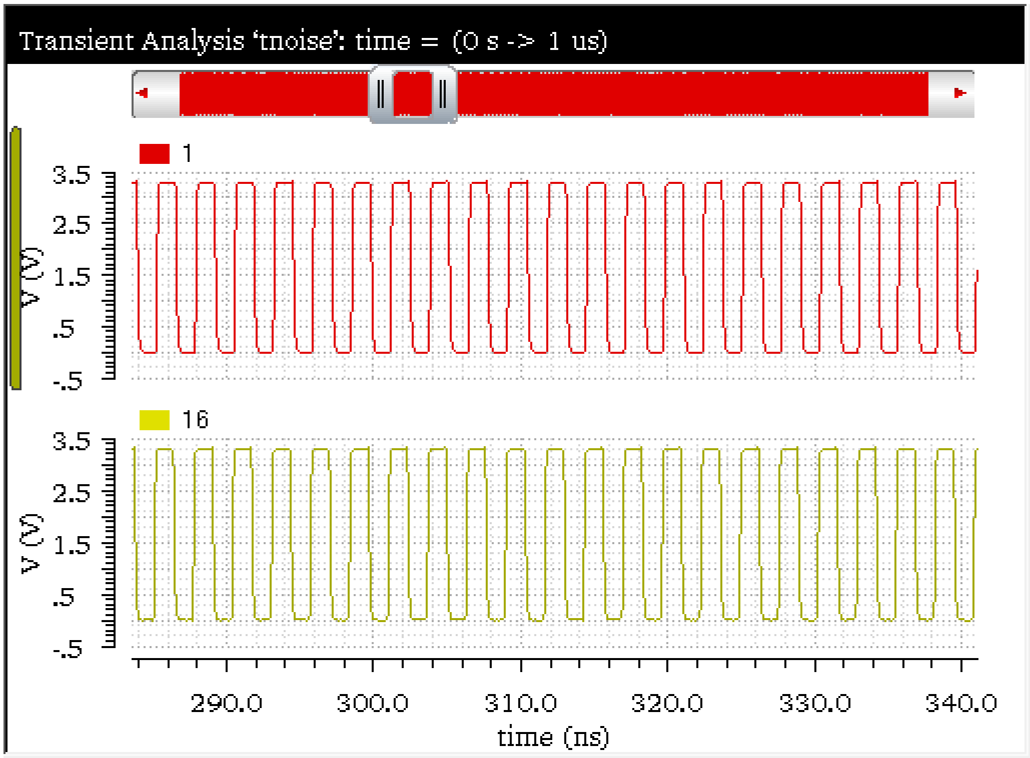
- *Signal1*—First waveform.
- *Signal2*—Second waveform.
- *From*—Starting time for the spectral analysis interval.
- *To*—Ending time for the spectral analysis interval.
- *Number of Samples*—Specifies how many times the domain points are to be used. The maximum frequency in the Fourier analysis is proportional to the *Number of Samples* parameter and inversely proportional to the difference between the starting time and the ending time. The default value is `512`.
- *Window Type*—Window you want to use.
  Default value: `Hanning`
  Valid values: `Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hanning, Kaiser, Parzen,Rectangular, and Triangular`

If you select *Kaiser*, type in a value for the Kaiser smoothing factor. The smoothing factor must be in the range 0 <= factor <= 15, where 0 is the same as using a rectangular window.

- *Smoothing Factor*—Specify the smoothing factor only if you have selected the *Window Type* as `Kaiser`.
- *Window Size*—Number of frequency domain points to use in the Fourier analysis.
  A larger window size results in an expectation operation over fewer samples, which leads to larger variations in the power spectral density. A small window size can smear out sharp steps in the power spectral density that might really be present. The default value is `256`.
- *Detrending Mode*—Specifies the trend mode.
  Valid values: `Linear`, `Mean`, `None`
  Default value: `None`
  The `psd` function works by applying a moving windowed FFT to time-series data. If there is a deterministic trend to the underlying data, you may want to remove the trend before performing the spectral analysis. For example, consider analyzing phase noise in a VCO model. Without the noise, the phase increases more or less linearly with time, so it is appropriate to set the detrending mode to `Linear`. To subtract an average value, set the detrending mode to `Mean`. Where the spectrum of raw data is desired, set the detrending mode to `None`.
- *Coherent Gain*—The coherent gain of a window is the zero frequency gain (or the DC gain) of the window. It is calculated by normalizing the maximum amplitude of the window to one and then summing the values of the window amplitudes over the duration of the window. The result is then divided by the length of the window (that is, the number of samples).
  When you do a dft, the applied Window Type (aside from the rectangular type) changes the signal amplitude. Applying a coherent gain factor is a way to get consistent results regardless of the window type.
  Valid Values: `none`, `default`, `magnitude`, `dB20`, or `dB10`
  Default value: `default`
- *Coherent Gain Factor*—This is the scaling factor. A non-zero value that scales the power spectral density by 1/(Coherent Gain).
  Valid values : 0 < coherent_gain_factor < 1. You can use 1 if you do not want this scaling factor to be used.
  Default value : `1`

Example

Consider the following input waveform from the transient analysis:



The `psdbb` function is applied on this input waveform with the following arguments:

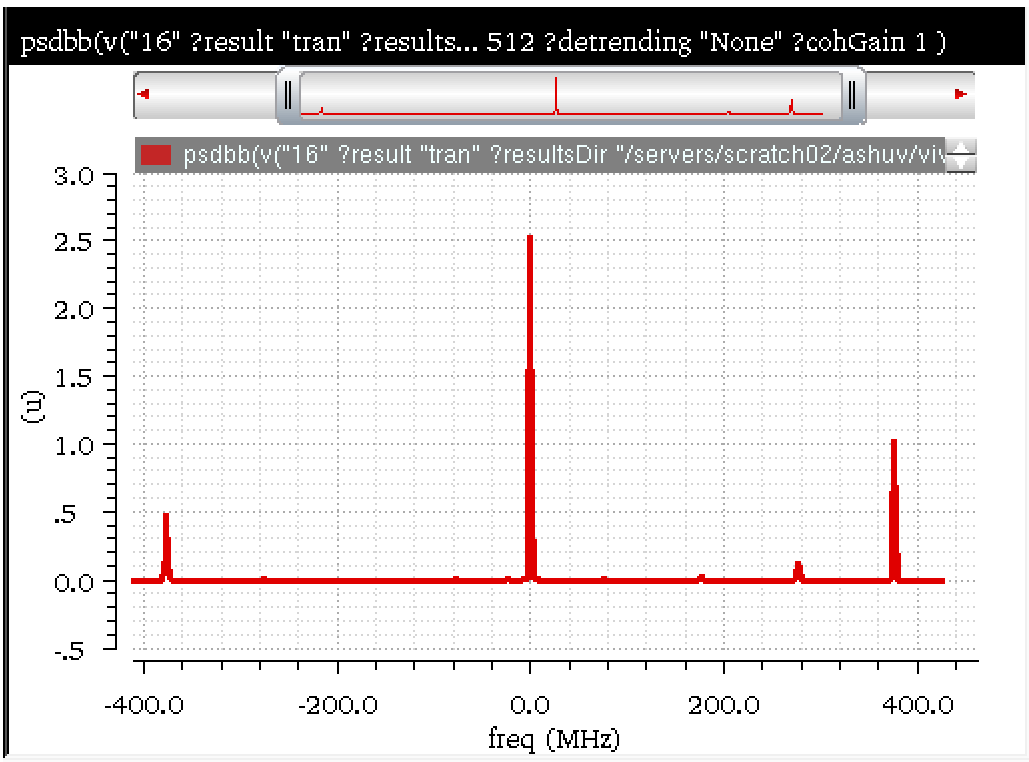
- *Signal1*—`v("16" ?result "tran" ?resultsDir "./nand2_ring_tnoise.raw")`
- *Signal2*—`v("1" ?result "tran" ?resultsDir "./nand2_ring_tnoise.raw")`
- *From*—`0`
- *To*—`1.2u`
- *Number of Samples*—`1024`
- *Window Type*—`Hanning`
- *Smoothing Factor*—`1`
- *Window Size*—`512`
- *Detrending Mode*—`None`

- *Coherent Gain*—`(none)`
- *Coherent Gain Factor*—`1`

The corresponding expression created in the Buffer is as follows:

```
psdbb(v("16" ?result "tran" ?resultsDir "./nand2_ring_tnoise.raw")
v("1" ?result "tran" ?resultsDir "./nand2_ring_tnoise.raw") 0 1.2u
1024 ?windowName "Hanning" ?smooth 1 ?windowSize 512 ?detrending
"None" ?cohGain 1 )
```

When you evaluate this expression, the power spectral density is calculated and a spectrum is displayed, as shown in the following output waveform:



Related OCEAN Function

The equivalent OCEAN command for `psdbb` is:

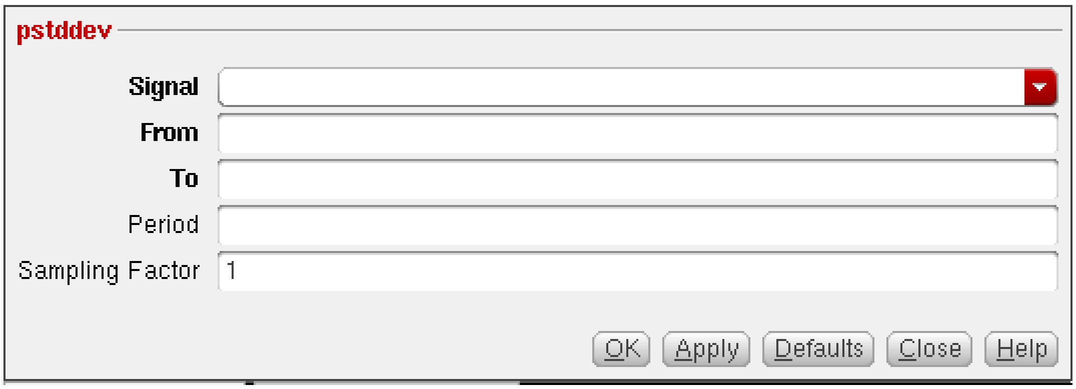
```
psdbb( o_waveform1 o_waveform2 f_timeStart f_timeEnd x_num
     ?windowName t_windowName ?smooth x_smooth ?cohGain f_cohGain
     ?windowsize x_windowsize ?detrending t_detrending)
     => o_waveformReal/nil
```

For more information, see `psdbb` in *OCEAN Reference*.

pstddev

(Periodic Standard Deviation)

Returns the periodic standard deviation of a family of signals for each time point.



This function includes the following arguments:

- *Signal*—Name of the signal for which you want to calculate the periodic standard deviation.
- *From*—Start time from where you want to calculate the periodic standard deviation.
- *To*—End time till you want to calculate the periodic standard deviation.
- *Period*—Difference between the end time and start time. This is an optional argument.
- *Sampling Factor*—Determines the output accuracy. Increase its value in order to increase the accuracy of the outputs.
  Default value is `1`.

Additional Information

The periodic standard deviation is represented by the following equation:

```
pstddev(tk) = stddev(sample( o_waveform(t), from tk, to N*T, linear,
by T))
```

where,

`o_waveform(t)` is the input waveform

`N` = `floor((to-from) / T)`

`T` = period

If the input waveform is a multi-dimensional waveform or a family of waveforms, the output is calculated as:

$$\text{pstddev(tk)} = 1/M \sum_{j=0}^{M-1} \text{pstddev(tk)[o\_waveform\_family[j]]}$$

where,

`o_waveform_family` is the input waveform family

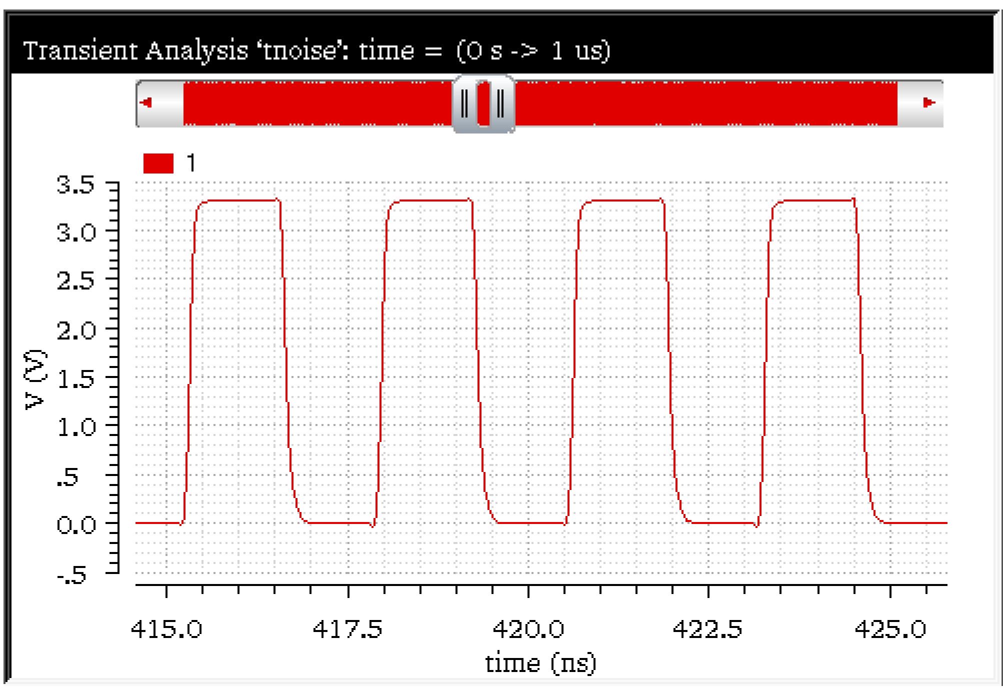
`M` is the number of leaf waveforms in the family.

`o_waveform_family[j]` is the `j`th leaf waveform of the family of waveform represented by the input waveform.

It calculates the `pstddev` on each leaf, and then averages over the number of leaves in the family of waveform. In this case the resultant is not a family of waveforms, but a normal waveform of dimension 1.

Example

Consider the following input waveform from transient analysis:



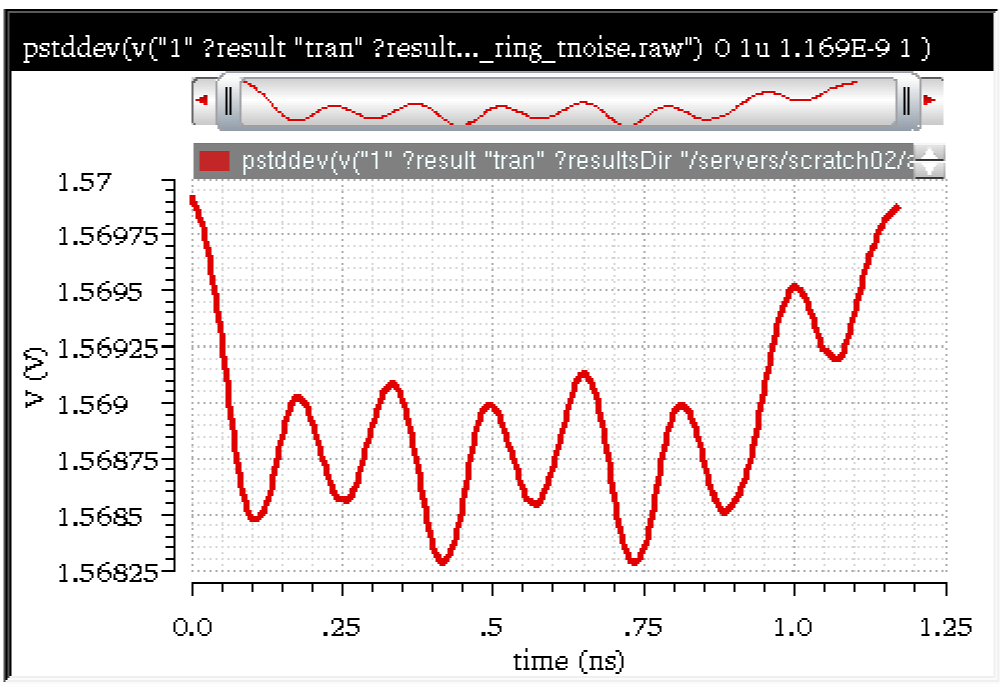
The `pstddev` function is applied on this input waveform with the following arguments:

- *Signal*—`v("1" ?result "tran" ?resultsDir "./nand2_ring_tnoise.raw")`
- *From*—`0`
- *To*—`1u`
- *Period*—`1.169E-9`
- *Sampling Factor*—`1`

The corresponding expression created in the Buffer is as follows:

```
pstddev(v("1" ?result "tran" ?resultsDir "./nand2_ring_tnoise.raw")
0 1u 1.169E-9 1)
```

When you evaluate this expression, the following output waveform is displayed in a new graph window, which indicates the periodic standard deviation at each time point.



Related OCEAN Function

The equivalent OCEAN command for `pstddev` is:

```
pstddev( o_waveform n_from n_to [n_period [n_sfactor]])
     => o_waveform/nil
```

For more information, see `pstddev` in *OCEAN Reference*.

pzbode

Calculates and plots the transfer function for a circuit from pole zero simulation data.This function also works on the parametric or sweep data.

This function includes the following arguments:

- *DC Gain*—Transfer gain constant.
- *Min. Frequency*—Minimum frequency for the bode plot.
- *Max. Frequency*—Maximum frequency for the bode plot.
- *No. of Points*—Frequency interval for the bode plot, in points per decade.
- *Poles*—Poles from the simulation data.
- *Zeroes*—Zeroes from the simulation data.

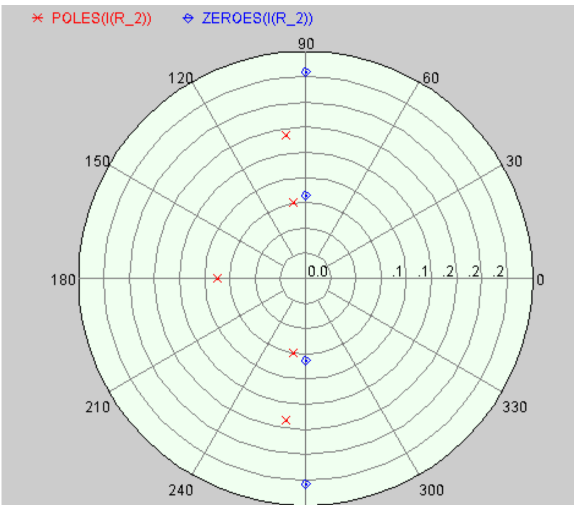
Important

You need to set the database context to the `pz` results directory for which you want to plot the pzbode plot. For more information about how to set a results directory as in-context results directory, see Changing In-Context Results Directory on page 42.

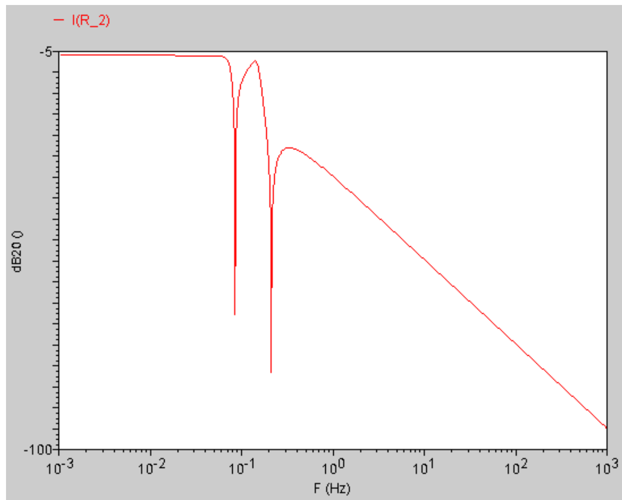
Example 1

The following diagram illustrates how the result with the values *poles*=`POLES<I<R_1>>`, *zeroes*=`ZEROES<I<R_1>>`, *c*=`I<R_1>\[K\]`, *minfreq*=`1e-3`, *maxfreq*=`1e3`, and *npoints*=`1000` is determined.

Polar Plot

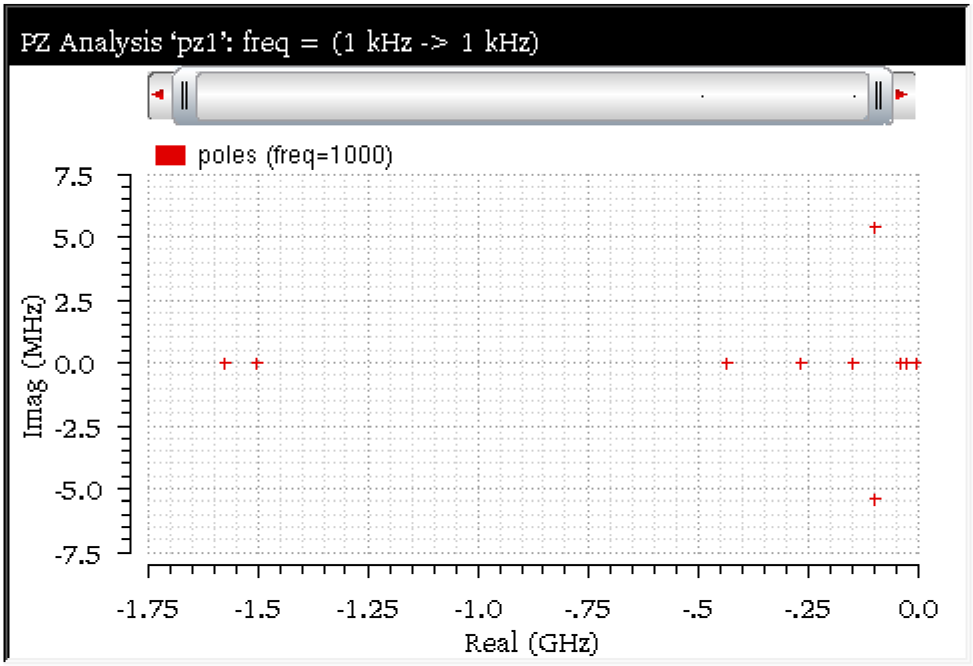


Corresponding bode plot



Example 2

This example shows the bode plot generated when you apply the `pzbode` function on the following pole-zero results from the `openloop_ampSim.raw` database:



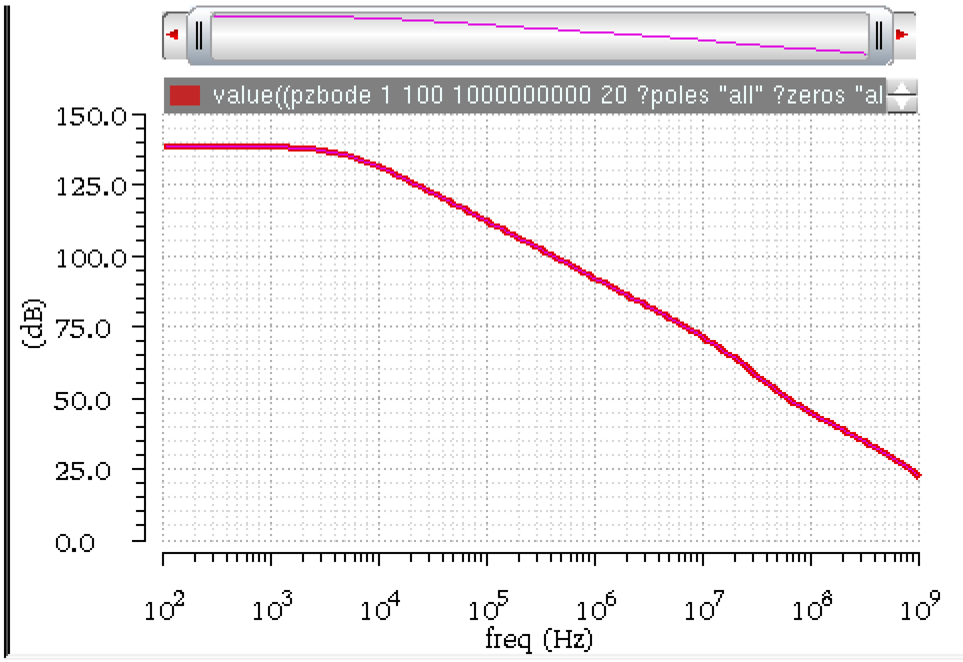
The following arguments are specified in this example:

- *DC Gain*—`1`
- *Min. Frequency*—`100`
- *Max. Frequency*—`1G`
- *No. of Points*—`20`
- *Poles*—`all`
- *Zeros*—`all`

The following expression is created in the Buffer:

```
pzbode(1 100 1G 20 ?poles "all" ?zeros "all")
```

Now, when you evaluate this expression, the following output waveform showing the bode plot is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `pzbode` is:

```
pzbode( f_c f_minf f_maxf x_nponits ?poles o_waveform1 ?zeros o_wavefoem2)
     => o_waveform/nil
```

For more information about this OCEAN function, see pzbode in *OCEAN Reference*.

pzfilter

Filters the poles and zeroes according to the specified criteria. The `pzfilter` function works only on pole zero simulation data. In addition, note that this function also works on the parametric or sweep data.

- *maxfreq*—Frequency upto which the poles and zeroes are plotted.
- *reldist*—Relative distance between the pole and zero. Pole-zero pairs with a relative distance lower than the specified value are not plotted.
- *absdist*—Absolute distance between the pole and zero. Pole-zero pairs with an absolute distance lower than the specified value are not plotted.
- *minq*—Minimum Q-factor. Pole-zero pairs with a Q-factor less than the specified value are not cancelled. The equations that define the Q-factor of a complex pole or zero are described in the section below.

Note: If you do not specify *maxfreq*, *reldist*, *absdist*, or *minq*, `pzfilter` filters out the poles and zeroes with a frequency higher than 10 GHz (default value of *maxfreq*).

Important

You need to set the database context to the `pz` results directory for which you want to plot the pzfilter plot. For more information about how to set a results directory as in-context results directory, see Changing In-Context Results Directory on page 42.

Additional Information

Equations Defining the Q-Factor of a Complex Pole or Zero

$$Re(X) < 0.0 \qquad Q = 0.5 \times \sqrt{[Im(X)/\ Re(X)]^2 + 1}$$

$$Re(X) = 0 \qquad UNDEFINED$$

$$Re(X) > 0.0 \qquad Q = -0.5 \times \sqrt{[lm(X)/ \ Re(X)]^2 + 1}$$

Filtration Rules

- Real poles can be cancelled only by real zeroes. A real pole *P* is cancelled by a real zero *Z* if the following equation is satisfied:

$$|P - Z| < absdist + \frac{|P + Z|}{2} \times reldist$$

- Complex poles and zeroes always occur in conjugated pairs. A pair of conjugated poles can only be canceled by a pair of conjugated zeroes. A pole pair `P1=a+jb`, `P2=a-jb` is cancelled by a zero pair `Z1=c+jd`, `Z2=c-jd`, if the following equation is satisfied:

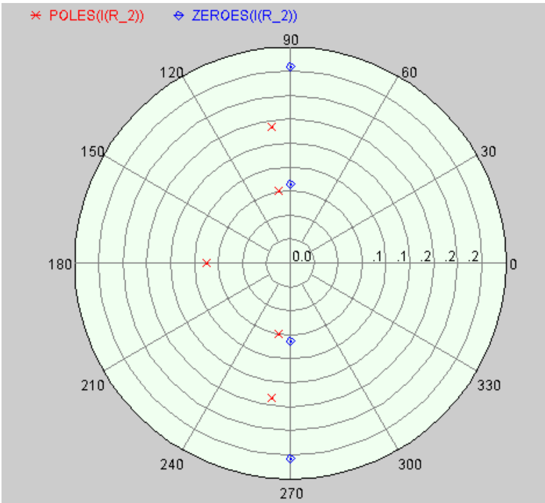
$$|P1 - Z1| = |P2 - Z2| = \sqrt{(a-c)^2 + (b-d)^2} < absdist + \frac{|a+c|}{2} \times reldist$$

- Poles in the right-half plane are never cancelled because they show the instability of the circuit.

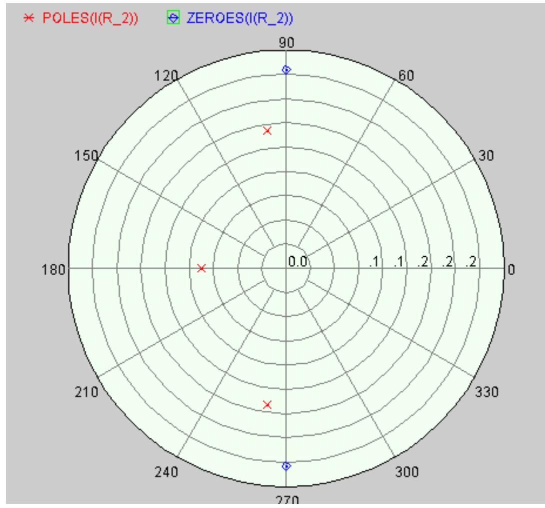
Example

The values *poles*=`POLES<I<R_2>>`, *zeroes*=`ZEROES<I<R_2>>`, *absdist*=`0.05`, and *minq*=`10000` filters pole-zero pairs with a relative distance of less than 0.05 Hz from the plot on the left side. In the filtered plot shown on the right side, two pole-zero pairs have been filtered out.

Original polar Plot

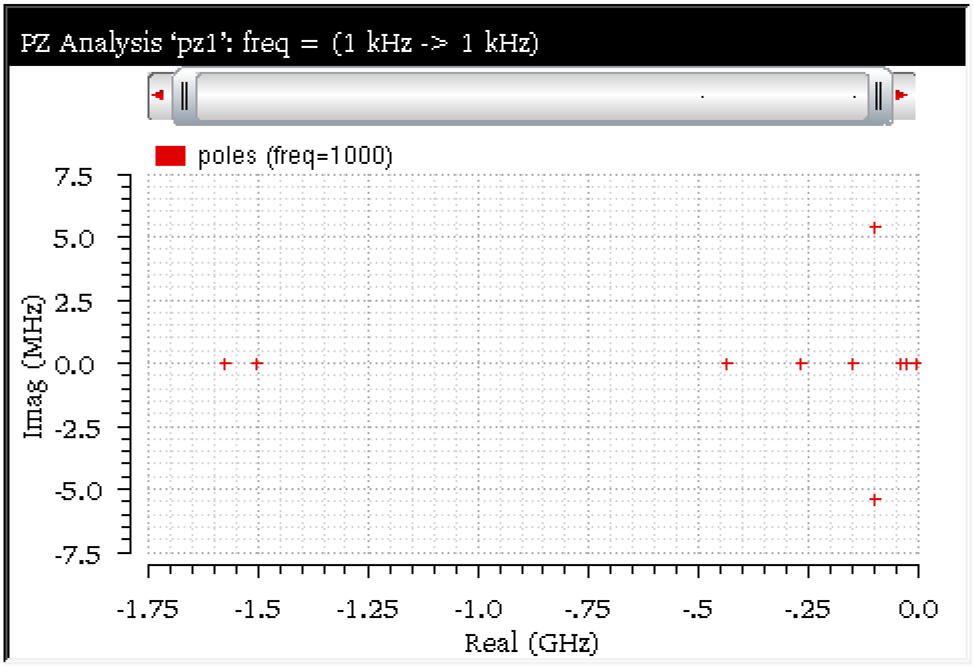


Filtered polar plot



Example

This example shows the output plot generated with the filtered pole-zero results when you apply the `pzfilter` function on the following pole-zero data:



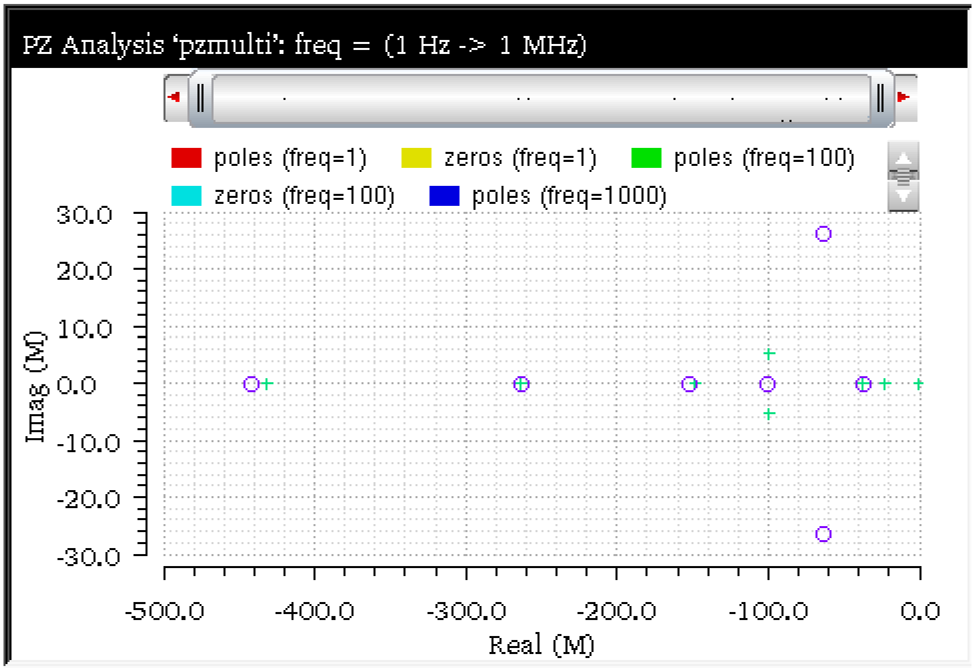
Only the following argument is specified in this example:

- *maxfreq*—`1G`

The following expression is created in the Buffer:

```
pzfilter(?maxfreq 1G )
```

Now, when you evaluate this expression, the following output plot is displayed in the graph window:



Related OCEAN Function

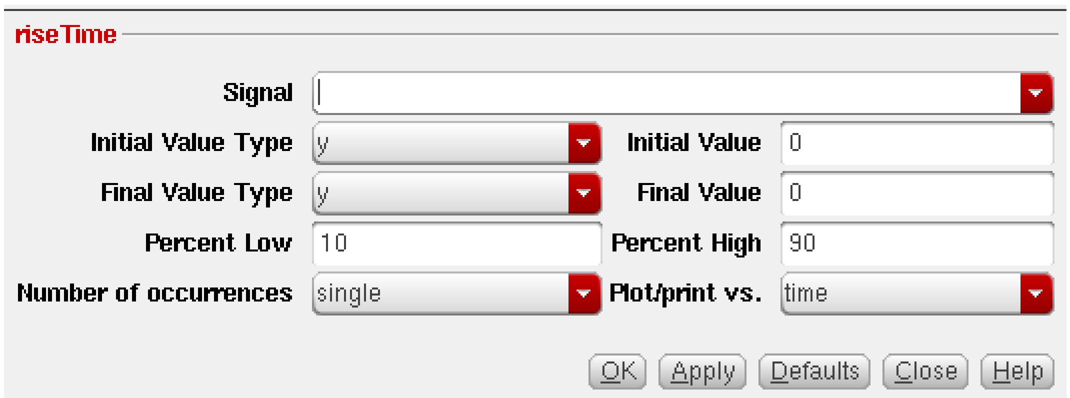
The equivalent OCEAN command for `pzfilter` is:

```
pzfilter( [o_PoleWaveform] [o_ZeroWaveform]
     [?maxfreq t_maxfreq] [?reldist n_relDist] [?absdist n_absdist]
     [?minq n_minq] [?output_type o_output] )
     => o_waveform/nil
```

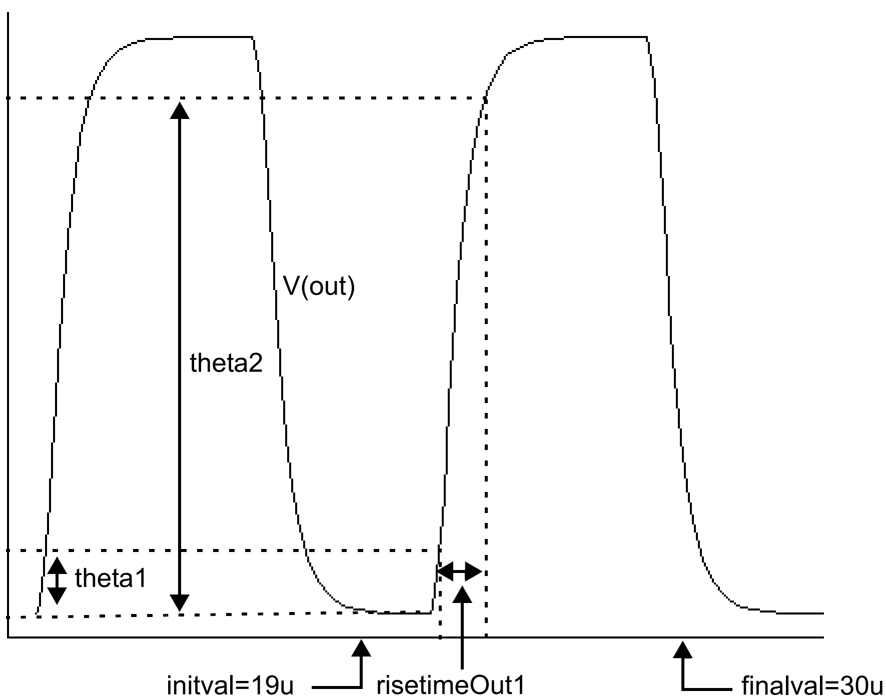
For more information about this OCEAN function, see pzfilter in *OCEAN Reference*.

risetime

Returns the rise time for a signal, which is the time taken by a signal to change from a specified low value to a specified high value. Typically, these values are 10% and 90% of the step height.



The following illustration shows how the rise time value is calculated:



This function includes the following arguments in the SKILL mode:

- *Signal*—Name of the signal.

- *Initial Value Type—S*pecify whether the initial value is the Y-axis value at the specified X-axis value (`y at x`) or Y-axis value (`y`). The default value is `y`.
- *Initial Value*—Value at which the rise time interval is started.
- *Final Value Type*—Specify whether the final value is the Y-axis value at the specified X-axis value (`y at x`) or Y-axis value (`y`). The default value is `y`.
- *Final Value*—Value at which the rise time interval ends.
- *Percent Low*—Percentage low. The default value is `10`.
- *Percent High*—Percentage high. The default value is `90`.
- *Number of Occurences*—Specify whether you want to retrieve only one occurrence of a risetime event for the given waveform (*single*), or all occurrences of risetime for the given waveform which you can later plot or print (*multiple*). The default value is `single`.
- *Plot/print vs*—Specify whether you want to retrieve risetime data against *time* (or another X-axis parameter for non-transient data) or *cycle*. Cycle numbers refer to the n'th occurrence of the delay event in the input waveform. The default value is `time`. The value in this field is ignored when you specify *Number of Occurrences* as *single*.

In the MDL mode,

- *sig*—Name of the signal.
- *initval*—X-axis (if inittype is `'x`) or Y-axis value (if inittype is `'y`) that starts the rise time interval. The measurement is always done in ordinate values.
- *finalval*—X-axis (if inittype is `'x`) or Y-axis (if inittype is `'y`) that ends the rise time interval. The measurement is always done in ordinate values.
- *inittype*—Specify whether the initial value is an X-axis (`'x`) or Y-axis value (`'y).`
- *finaltype*—Specify whether the final value is an X-axis (`'x`) or Y-axis value (`'y)`
- *theta1*—Percentage low.
- *theta2*—Percentage high.
- *xtol*—Absolute tolerance in the X direction.
- *ytol*—Absolute tolerance in the Y direction.
- *accuracy*—Specify that the function uses interpolation in the SKILL mode.

  In the MDL mode, *accuracy* specifies whether the function should use interpolation, or use iteration controlled by the absolute tolerances to calculate the value. `'interp` directs the function to use interpolation, and `'exact` directs the function to consider the xtol and yval values.

Additional Information

Consider the following equations:

```
Val1 = theta1 / 100.0 * diff + initVal

Val2 = theta2 / 100.0 * diff + initVal
```

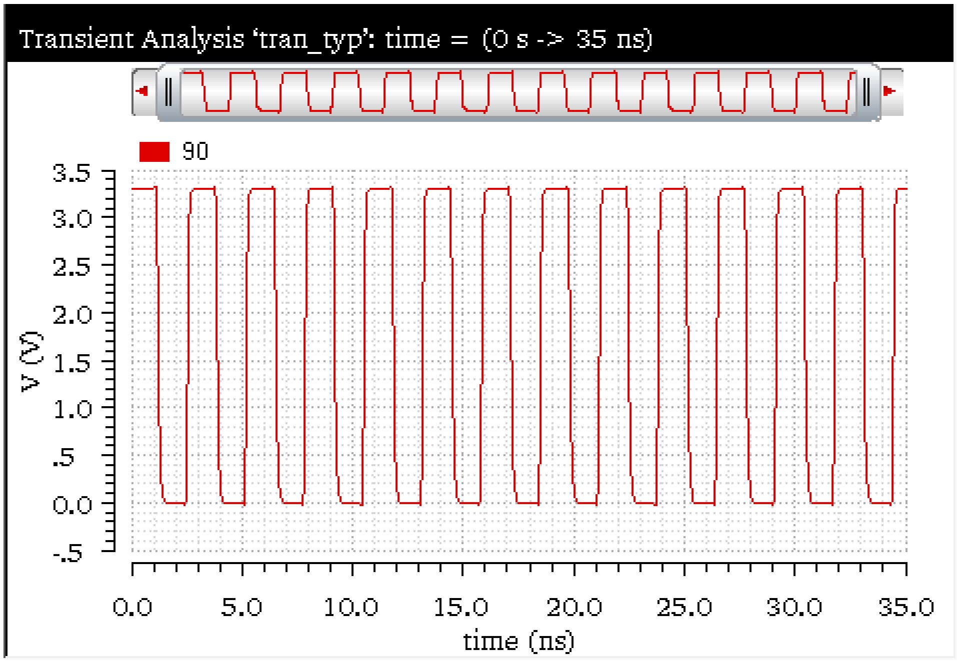
The following table shows how the *riseTime* function works when you apply the above equations:

Function *riseTime(w initVal nil finalVal nil theta1 theta2)*	**initVal**	**finalVal**	**theta1**	**theta2**	**Val1**	**Val2**	**Output**
Case 1	0	1	10	80	0.1	0.8	1.4n (time taken to rise from 200.0p ns to 1.6ns) See Figure for Case 1 on page 679.
Case 2	0	1	80	10	0.1	0.8	0.7n (time taken to fall from 4.2ns to 4.9ns) See Figure for Case 2 on page 680.

Function *riseTime(w initVal nil finalVal nil theta1 theta2)*	**initVal**	**finalVal**	**theta1**	**theta2**	**Val1**	**Val2**	**Output**
Case 3	1	0	10	80	0.9	0.2	0.7n (time taken to fall from 4.1ns to 4.8ns) See Figure for Case 3 on page 681.
Case 4	1	0	80	10	0.2	0.9	1.4n (time taken to rise from 400.0p s to 1.8ns) See Figure for Case 4 on page 682.

Example 3

Consider the following input waveform from transient analysis:



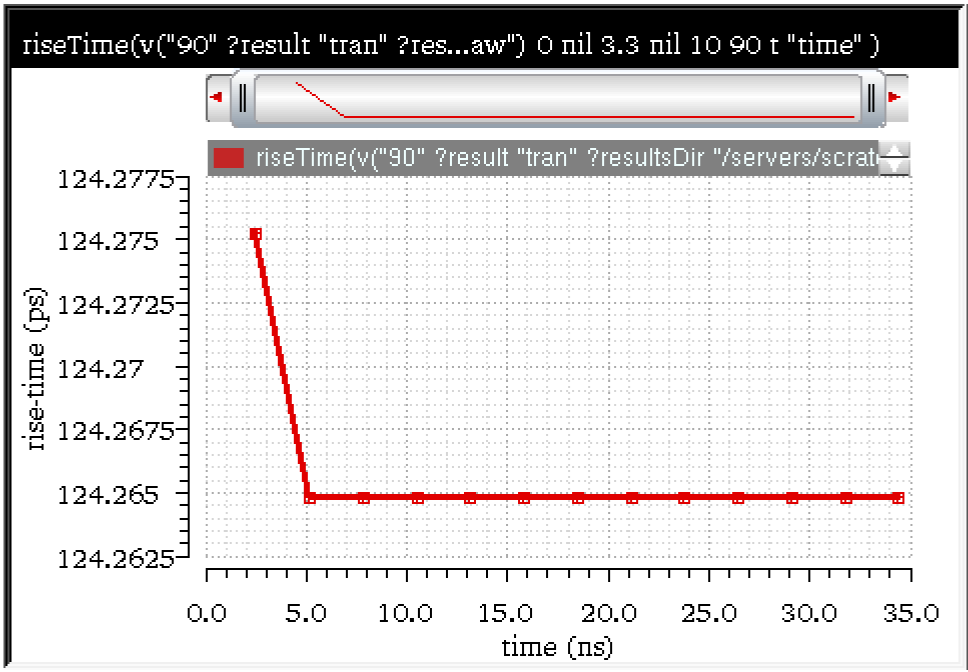
The `riseTime` function is applied on this input waveform with the following arguments:

- *Signal*—`v("90" ?result "tran" ?resultsDir "./nand2_ring.raw")`
- *Initial Value Type*—`y`
- *Initial Value*—`0`
- *Final Value Type*—`y`
- *Final Value*—`3.3`
- *Percent Low*—`10`
- *Percent High*—`90`
- *Number of occurrences*—`multiple`
- *Print/plot vs.*—`time`

The corresponding expression created in the Buffer is as follows:

```
riseTime(v("90" ?result "tran" ?resultsDir "./nand2_ring.raw") 0 nil
3.3 nil 10 90 t "time" )
```

When you evaluate this expression, the rise time is calculated and the following output waveform is displayed in a new graph window:



Related OCEAN Function

The equivalent OCEAN command for `riseTime` is:

```
riseTime( o_waveform n_initVal g_initType n_finalVal g_finalType
     n_theta1 n_theta2
     [g_multiple [s_Xname][g_histoDisplay][x_noOfHistoBins] ] )
     => o_waveform/n_value/nil
```

For more information, see `riseTime` in *OCEAN Reference*.

rms

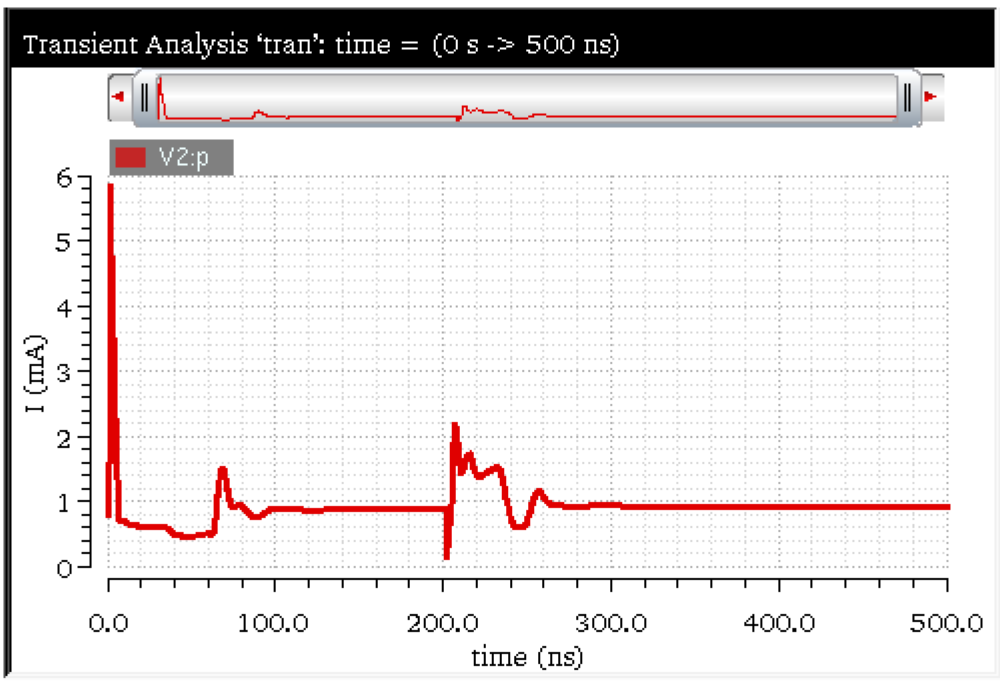
(Root Mean Square)

Returns the root mean square of a signal. The equation for `rms` is:

```
rms = sqrt(average(integ(f(x)**2)))
```

Example

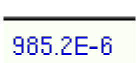
Consider the following input waveform from the transient analysis:



When you send this input signal to Calculator and apply the `rms` function, the function is applied on the input waveform expression displayed in the Buffer. The corresponding expression created in the Buffer is as follows:

```
rms(i("V2:p" ?result "tran-tran" ?resultsDir "ampsim.raw")))
```

When you evaluate this expression, the following scalar output is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN command for `rms` is:

```
rms( o_waveform )
     => o_waveform/n_value/nil
```

For more information, see `rms` in *OCEAN Reference*.

rmsNoise

Computes the integrated root-mean-square of the total output noise over the bandwidth specified in Hertz in the *From* and *To* fields.This function is available only in the SKILL mode.

The used to calculate `rmsNoise` is as follows:

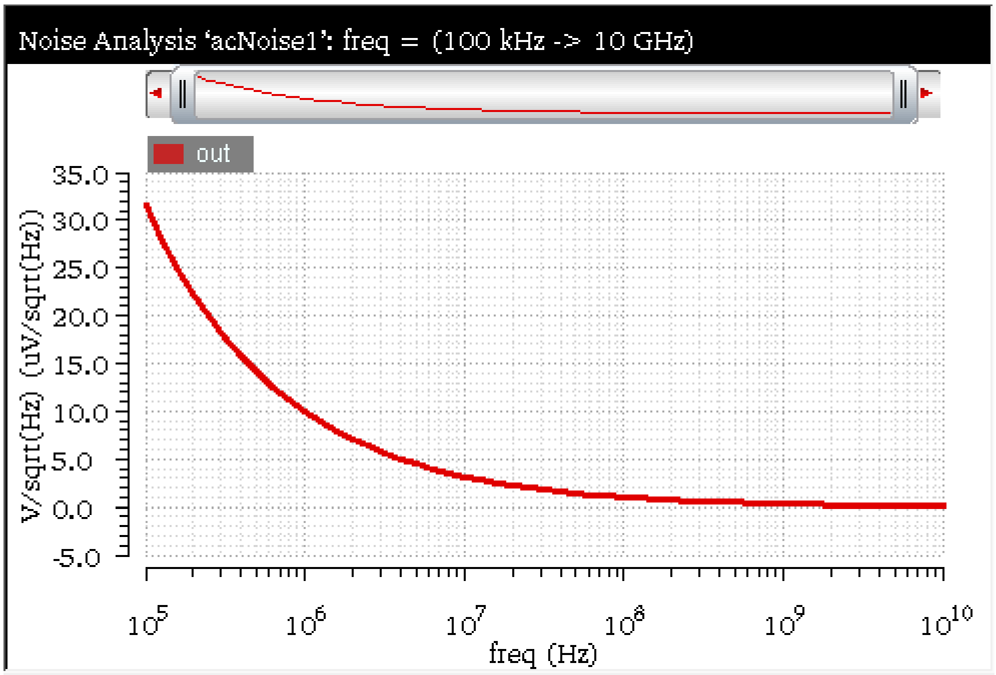
```
rmsNoise = sqrt(integ(f(x)**2))
```

This function includes the following arguments:

- *From*—Starting time for the measurement.
- *To*—Ending time for the measurement.

Example

Consider the following input waveform from the noise analysis: `v("out" ?result "noise" ?resultsDir "./linear.raw")`



Note: Before using this function, you need to set the results database context to the noise analysis results database that contains the input signal, such as `linear.raw` used in this example. For more information about how to change the in-context results directory, see Changing In-Context Results Directory on page 42.

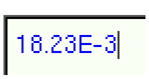
The `rmsNoise` function is applied on this input waveform with the following arguments:

- *From*—`10`
- *To*—`1M`

The corresponding expression created in the Buffer is as follows:

```
rmsNoise(10 1M)
```

When you evaluate this expression, the following scalar output indicating the root mean square is displayed in the Buffer:



Related OCEAN Function

The equivalent command for `rmsNoise` is:

```
rmsNoise( n_from n_to )
     => o_waveform/n_value/nil
```

For more information, see `rmsNoise` in *OCEAN Reference*.

root

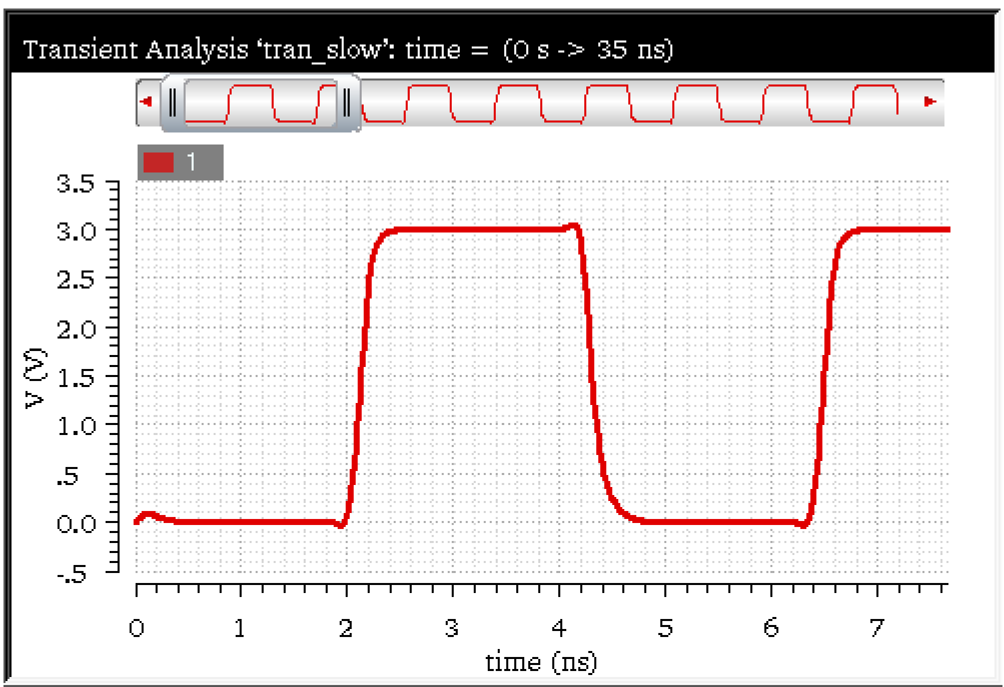
Computes the value of x at which f(x) equals the specified threshold. This function is available only in the SKILL mode.

This function includes the following arguments:

- *Signal*—Name of the signal.
- *Threshold*—Waveform value at which the root value is to be computed.
- *Nth Root*—Root value you want to calculate.

Example

Consider the following input waveform from transient analysis:



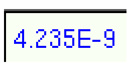
The `root` function is applied on this input waveform with the following arguments:

- *Signal*—`v("1" ?result "tran_slow-tran" ?resultsDir "./nand2_ring.raw")`
- *Threshold*—`2.5`
- *Nth Root*—`2`

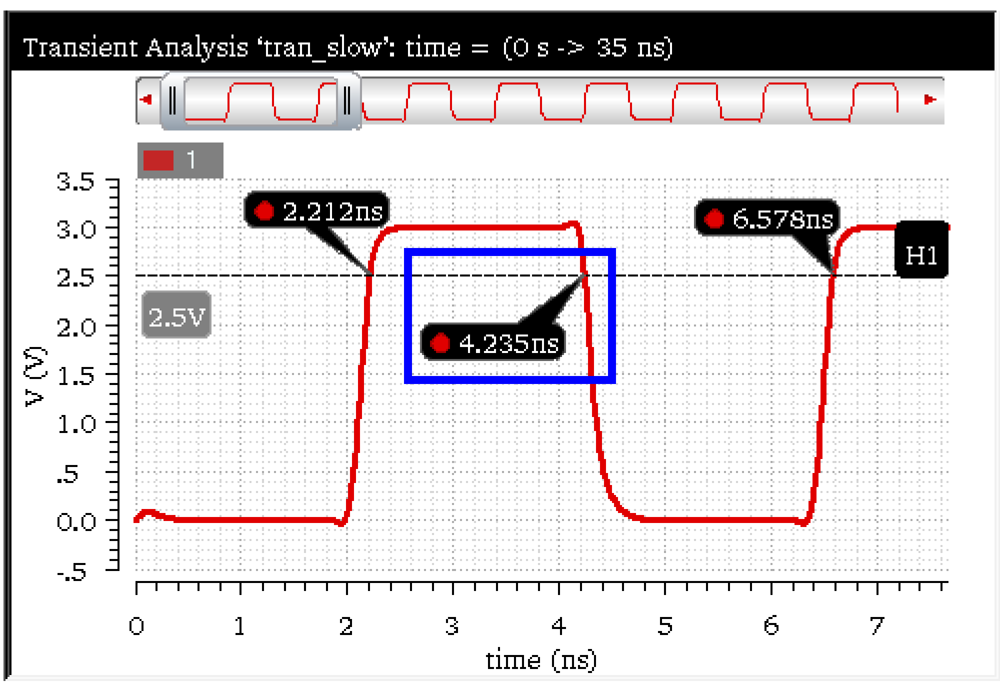
The corresponding expression created in the Buffer is as follows:

```
root(v("1" ?result "tran_slow-tran" ?resultsDir "./nand2_ring.raw")
2.5 2 )
```

When you evaluate this expression, the scalar output is displayed in the Buffer that indicates the second crossing point value at which the input signal crosses the threshold `2.5`V.



You can also verify the output generated by placing a horizontal marker in the graph at `Y-axis = 2.5`V, as displayed in the figure below:



Related OCEAN Function

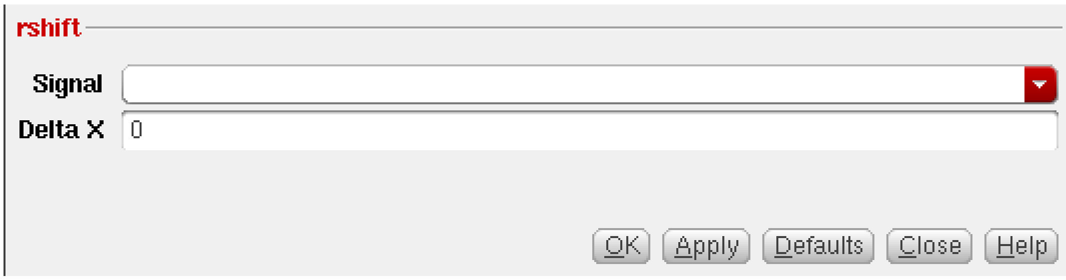
The equivalent OCEAN command for `root` is:

```
root( o_waveform n_rootVal x_n )
     => o_waveform/n_value/l_value/nil
```

For more information, see `root` in *OCEAN Reference*.

rshift

Shifts the data in the graph window to the right by the specified amount. A negative value shifts the data to the left. This function is available only in the SKILL mode.

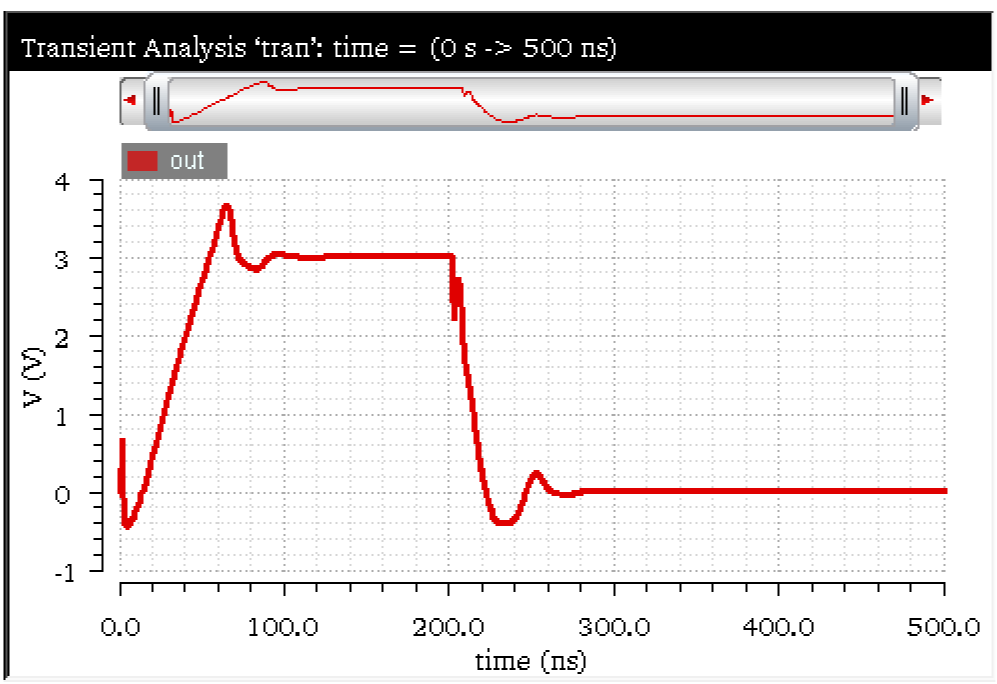


This function includes the following arguments:

- *Signal*—Name of the signal you want to right shift.
- *Delta X*—Amount on X-axis by which you want to shift the data.

Example

Consider the following input waveform from transient analysis:



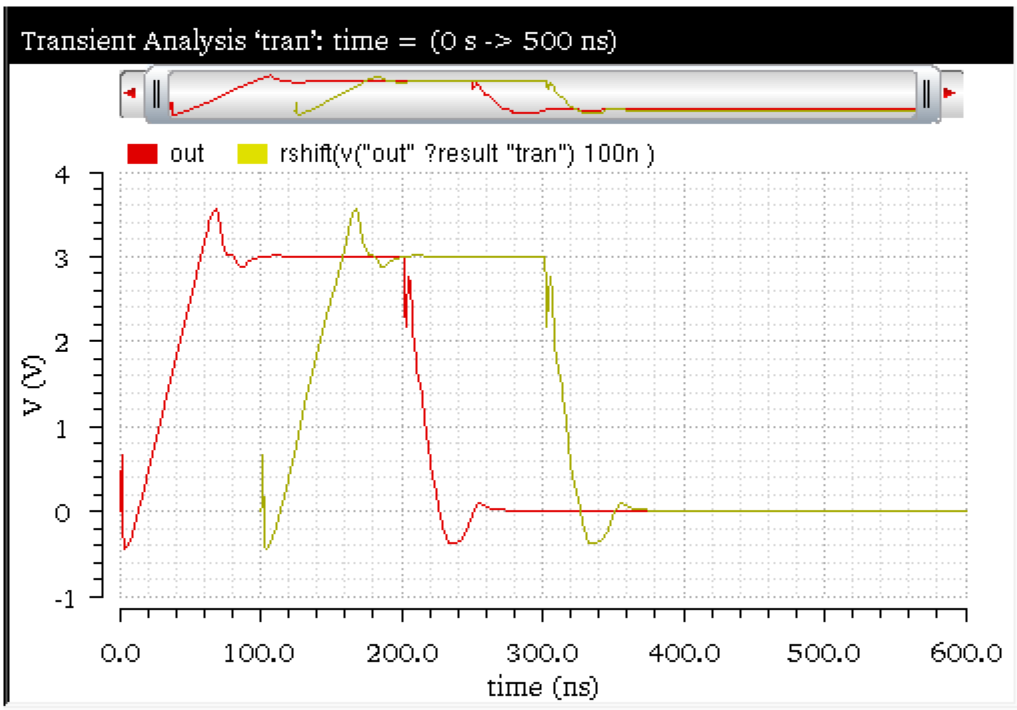
The `rshift` function is applied to this input waveform with the following arguments:

- *Signal*—`v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")`
- *Delta X*—`100n`

The corresponding expression created in the Buffer is as follows:

```
rshift(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw") 100n )
```

When you evaluate this expression and plot the result in the same graph window in the append mode, the following output waveform is displayed in the graph window:

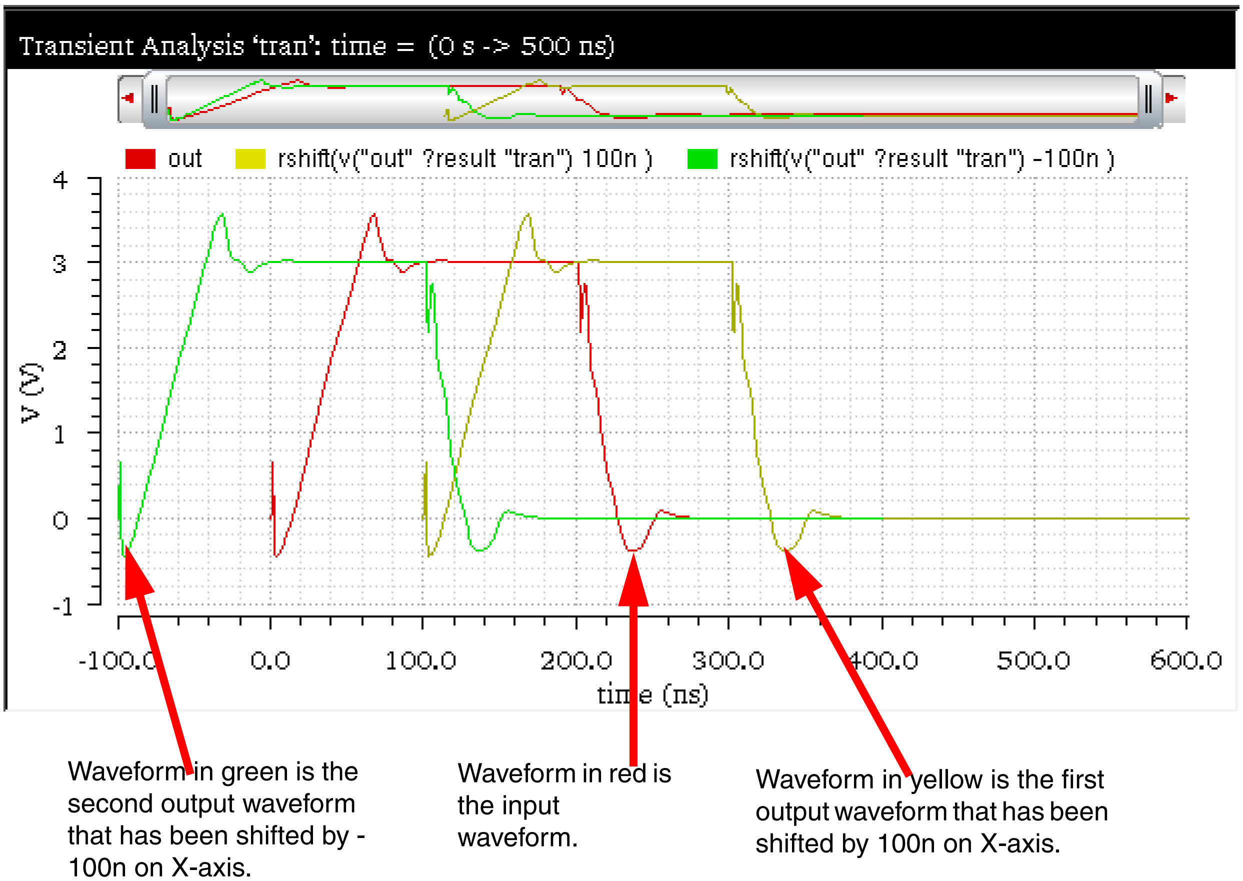


Note that the output waveform starts from time=`100n`, which indicates that the X-axis has been shifted by `100n` from the starting point `0.0n`.

Now, again apply the rshift function on the input signal with Delta=-100. The expression created in the Buffer is as follows:

```
rshift(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw") -100n
)
```

Now, when you evaluate this expression and plot the results in an appned mode, the following output waveform appears in the graph window.



Note that the output waveform starts from time=`-100n`, which indicates that the X-axis has been shifted by `-100n` from the starting point `-100n`.

Related OCEAN Function

The equivalent OCEAN command for `rshift` is:

```
rshift( o_waveform n_delta )
     => o_waveform/nil
```

For more information, see `rshift` in *OCEAN Reference*.

sample

Returns a waveform representing a sample of the signal based on step size or points per decade.

This function includes the following arguments in SKILL mode:

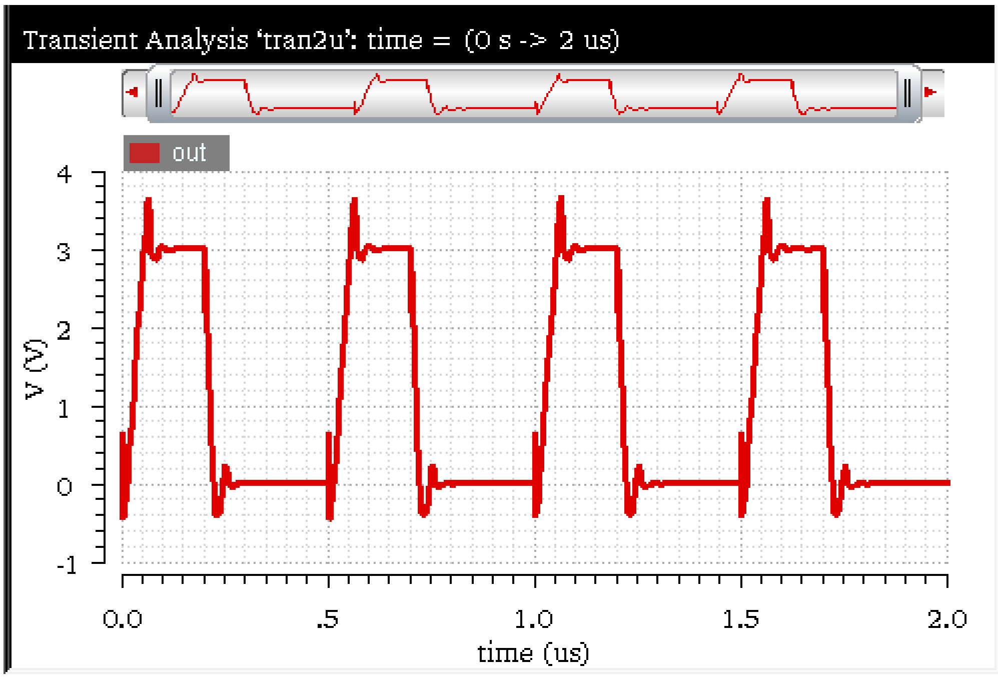
- *Signal*—Name of the signal.
- *From*—X-axis value at which the sampling begins.
- *To*—X-axis value at which the sampling stops
- *Type*—Specifies whether the sample should be linear or logarithmic. The default value is `linear`.
- *By*—Specifies the step size for the sample (if *type* is `linear`) or the points per decade (if *type* is `logarithmic`).

In the MDL mode,

- *sig*—Signal name
- *from*—X-axis value at which the sampling begins.
- *to*—X-axis value at which the sampling stops.
- *type*—Specifies whether the sample should be linear or logarithmic.
- *by*—Specifies the step size for the sample(if *type* is `'linear`) or the points per decade (if *type* is `'logarithmic`)

Example

Consider the following input signal from the transient analysis:



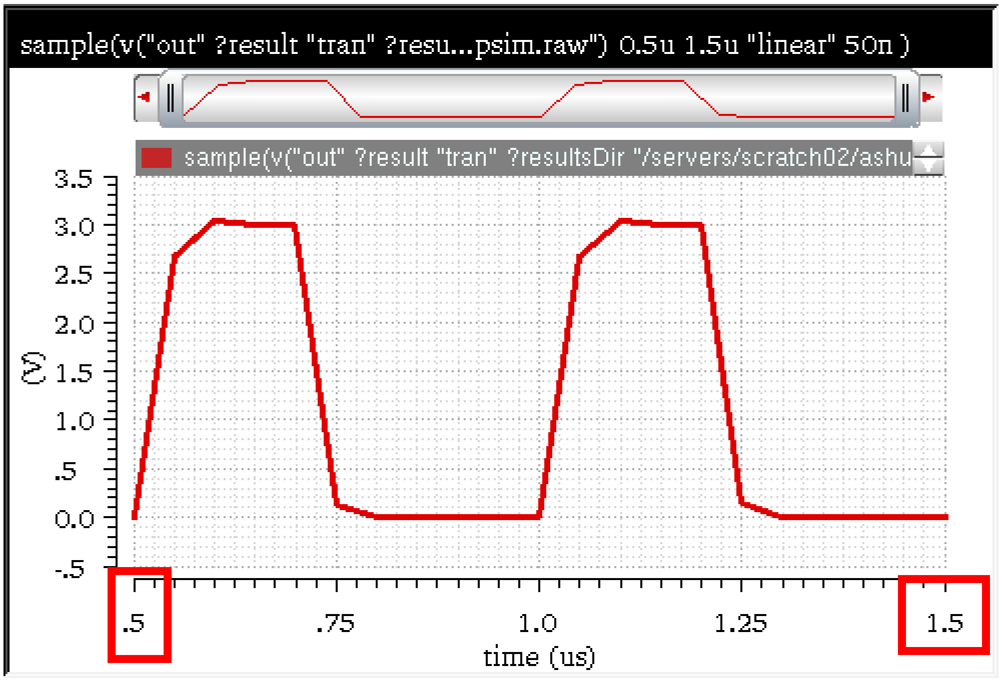
The `sample` function is applied on this input waveform with the following arguments:

- *Signal*—`v("out" ?result "tran" ?resultsDir "./ampsim.raw")`
- *From*—`0.5u`
- *To*—`1.5u`
- *Type*—`linear`
- *By*—`50n`

The corresponding expression created in the Buffer is as follows:

```
sample(v("out" ?result "tran" ?resultsDir "/ampsim.raw") 0.5u 1.5u
"linear" 50n )
```

When you evaluate this expression, the sample of the signal is calculated in the X-axis range `0.5u` to `1.5u`. The following output waveform is displayed in a new graph window:



Related OCEAN Function

The equivalent OCEAN command for `sample` is:

```
sample( o_waveform n_from n_to t_type n_by )
     => o_waveform/n_number/nil
```

For more information, see `sample` in *OCEAN Reference*.

settlingTime

Calculates the time required by a signal to settle at a final value within a specified limit.

This function includes the following arguments in the SKILL mode:

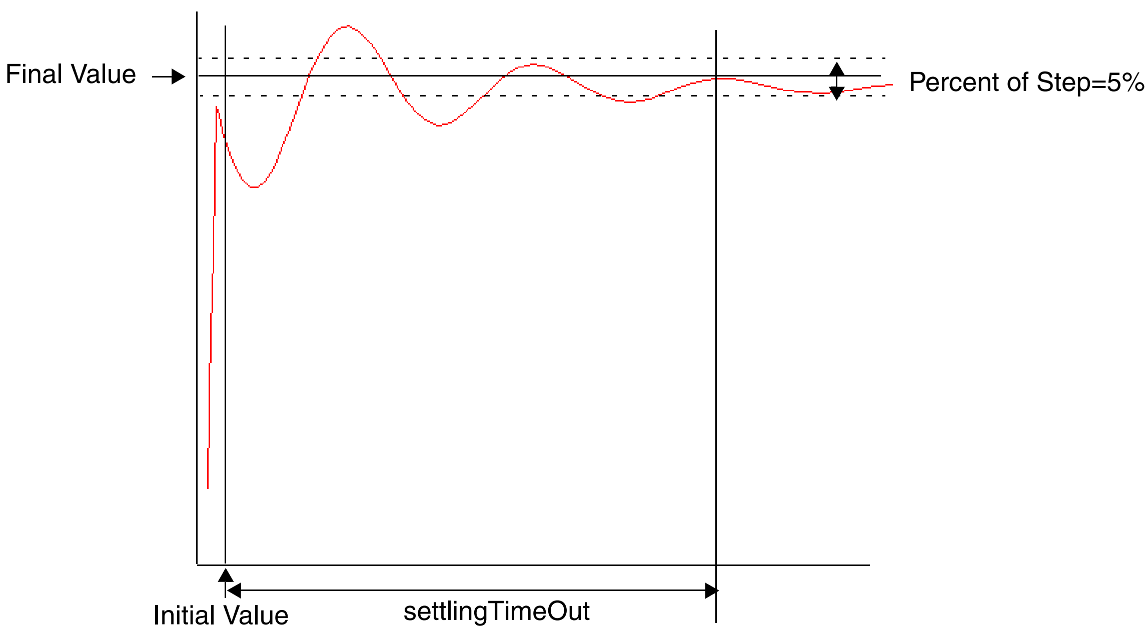
- *Signal*—Name of the signal.
- *Initial Value Type*—Specifies whether the initial value is the Y-axis value at the specified X-axis value (`y at x`) or Y-axis value (`y`). The default value is `y`.
- *Initial Value*—Starting value for the measurement.
- *Final Value Type*—Specifies whether the final value is the Y-axis value at the specified X-axis value (`y at x`) or Y-axis value (`y`). The default value is `y`.
- *Final Value*—Final value for the measurement.
- *Percent of Step*—Percentage (*Final value* - *Initial Value*) within which the signal has to settle. The default value is `5`.
- *Number of Occurences*—Specifies whether you want to retrieve only one occurrence of a settling time event for the given waveform (*single*), or all occurrences of settling time for the given waveform which you can later plot or print (*multiple*). The default value is `single`.
- *Plot/print vs.*—Specifies whether you want to retrieve settling time data against *time*(or another X-axis parameter for non-transient data) or *cycle*. Cycle numbers refer to the n'th occurrence of the delay event in the input waveform. The default value is `time`. The value in this field is ignored when you specify *Number of Occurrences* as *single*.

In the MDL mode:

- *sig*—Name of the signal.
- *initval*—Starting value for the measurement.
- *finalval*—Final value for the measurement.
- *inittype*—Specifies whether the initial value is an X-axis (`'x`) or Y-axis value (`'y`).
- *finaltype*—Specifies whether the final value is an X-axis (`'x`) or Y-axis value (`'y`)
- *theta*—Percentage of (*finalval*-*initval*) within which the signal has to settle.

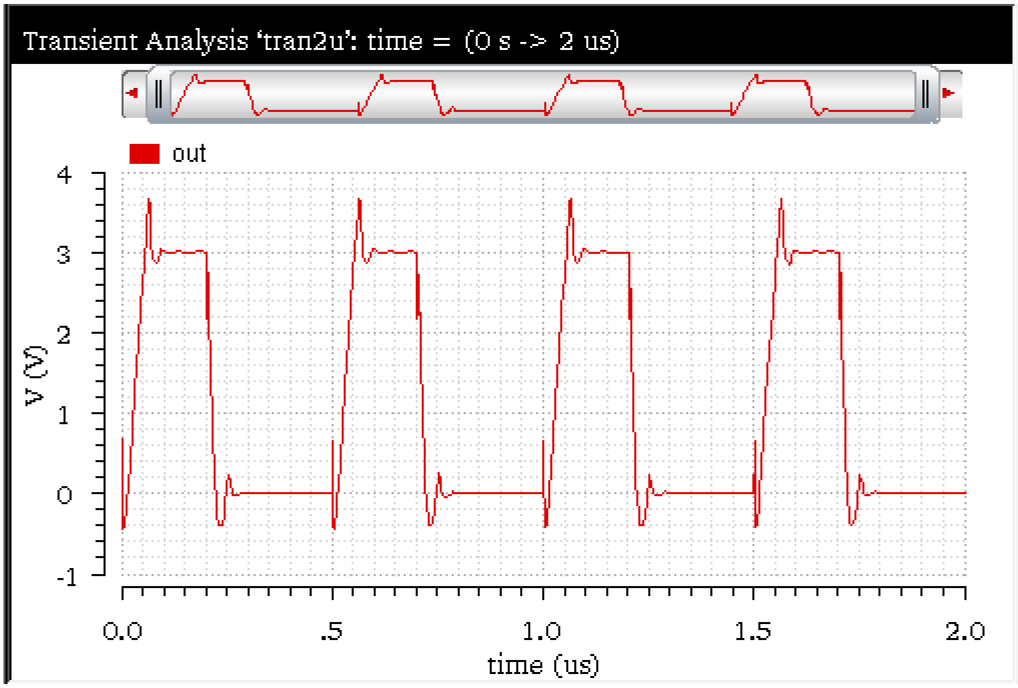
Example

The following diagram illustrates how the result with the values *signal*=`V(out)`, *Initial Value Type*=`y`, *Initial Value*=`0`, *Final Value Type*=`y at x`, *Final Value*=`1.0`, and *Percent of Step*=`5` is determined.



Example

Consider the following input waveform from the transient analysis:



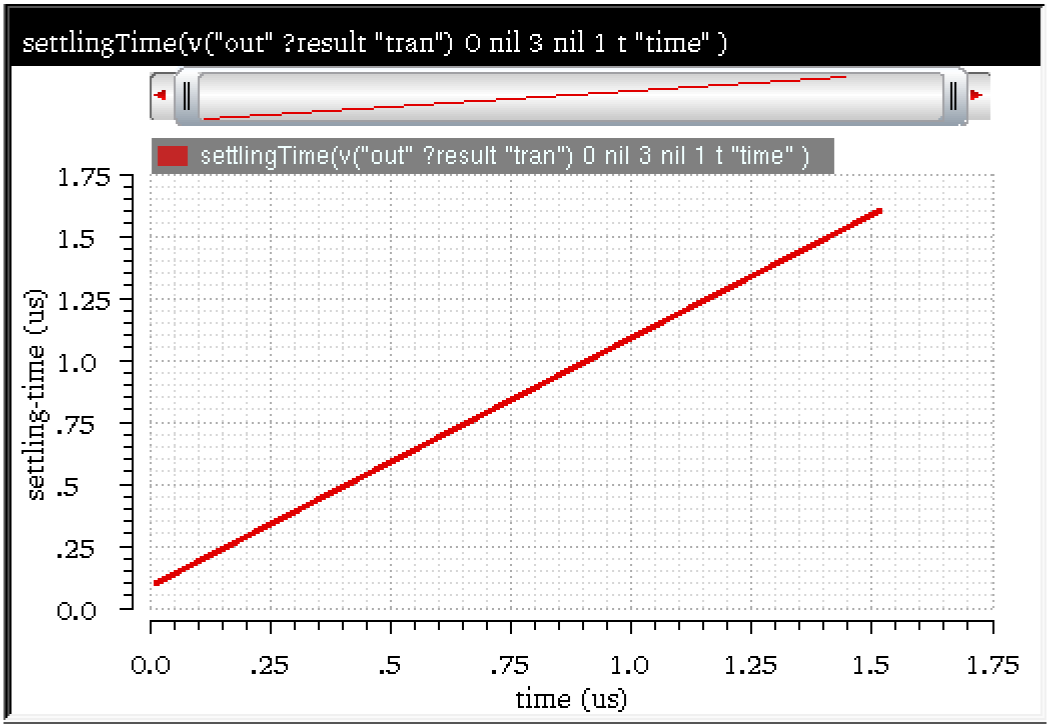
The `settlingTime` function is applied on this input waveform with the following arguments:

- *Signal*—`v("out" ?result "tran" ?resultsDir "./ampsim.raw"))`
- *Initial Value Type*—`y`
- *Initial Value*—`0`
- *Final Value Type*—`y`
- *Final Value*—`3`
- *Percent of Step*—`1`
- *Number of occurrences*—`multiple`
- *Plot/print vs.*—`time`

The corresponding expression created in the Buffer is as follows:

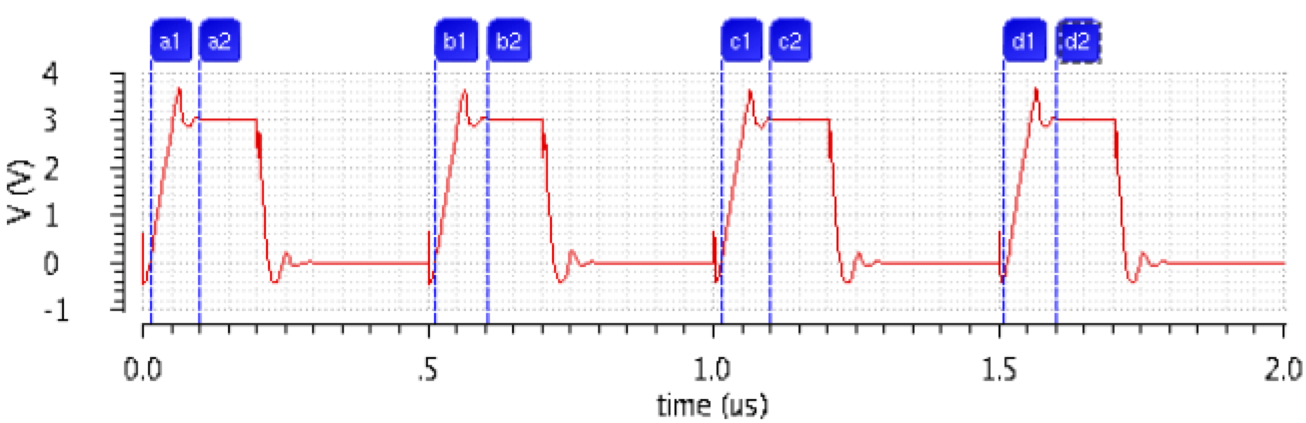
```
settlingTime(v("out" ?result "tran") 0 nil 3 nil 1 t "time" )
```

Now, when you evaluate this expression, the settling time is calculated and following output waveform is displayed in the graph window:



The above example calculates multiple settling time values because the *Number of occurrences* field has been set to `multiple`.

Perform the steps listed below to measure the individual settling times for the curves, such as `a2-a1,...,d2-d1` marked on the input waveform shown below,



1. Apply the `xval` function on the settling time waveform expression to measure the X values from the settling time signal. The corresponding expression created in the Buffer is as follows:

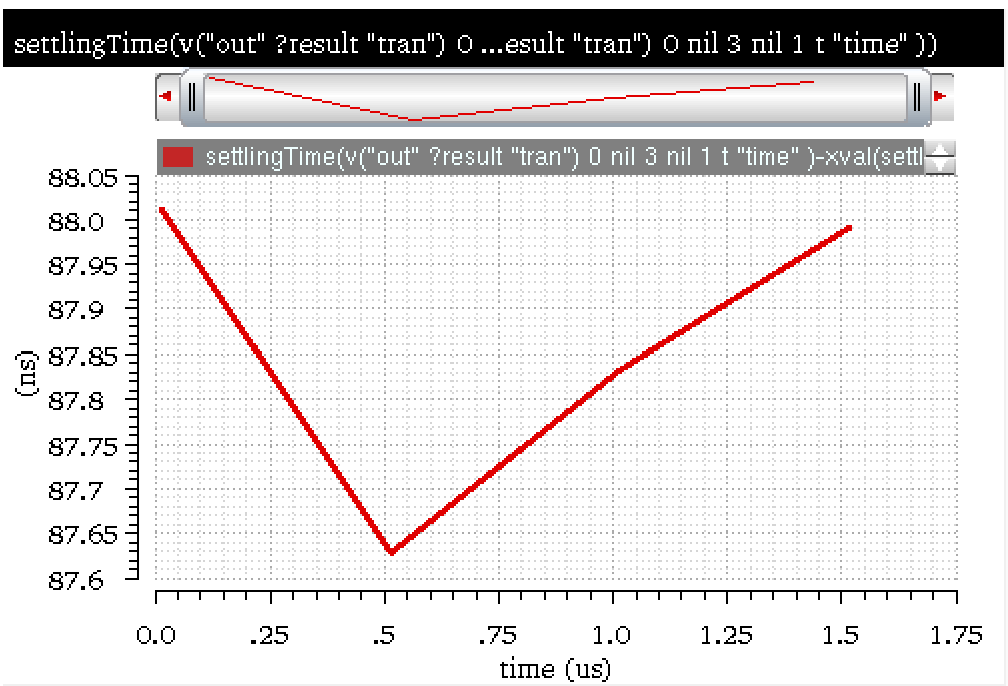
```
xval(settlingTime(v("out" ?result "tran") 0 nil 3 nil 1 t "time"
))
```

2. Calculate the difference between the two waveforms by subtracting the two waveforms. The corresponding expression created in the Buffer is as follows:

```
settlingTime(v("out" ?result "tran") 0 nil 3 nil 1 t "time" )-
xval(settlingTime(v("out" ?result "tran") 0 nil 3 nil 1 t "time"
))
```

3. Evaluate the expression.

   The following output waveform appears in the graph window that indicates the individual settling time for each curve in the waveform.



Related OCEAN Function

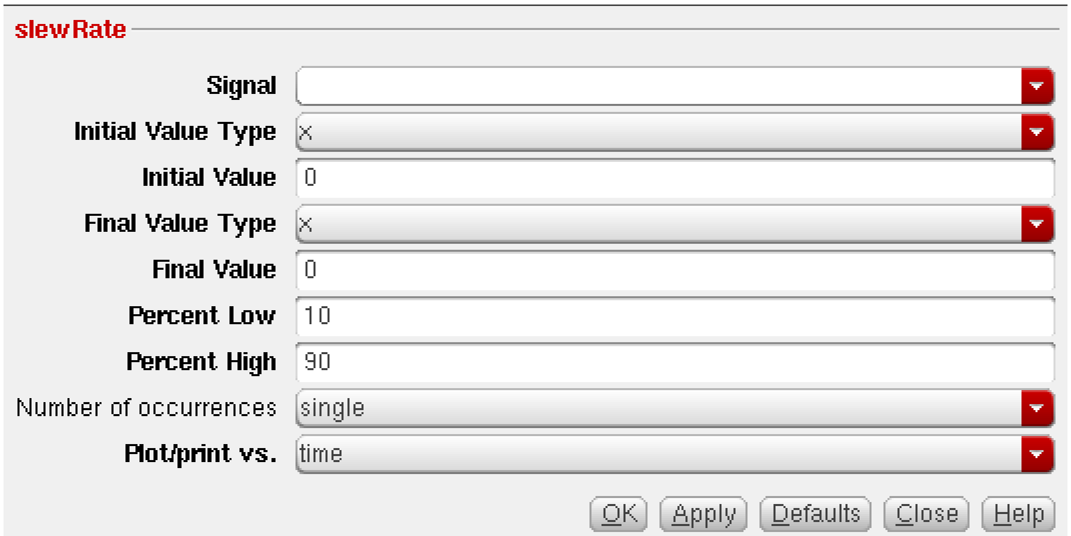
The equivalent OCEAN command for `settlingTime` is:

```
settlingTime( o_waveform n_initVal g_initType n_finalVal g_finalType
     n_theta [g_multiple [s_Xname]] )
     => o_waveform/n_value/nil
```

For more information, see `settlingTime` in *OCEAN Reference*.

slewrate

Computes the average rate at which the Buffer expression changes from percent low to percent high of the difference between the initial value and the final value.



This function includes the following arguments in the SKILL mode,

- *Signal*—Name of the signal.
- *Initial Value Type*—Specifies whether the initial value is an X-axis (`x`) or Y-axis value (`y`). The default value is `y`.
- *Initial Value*—Starting value for the measurement.
- *Final Value Type*—Specifies whether the final value is an X-axis (`x`) or Y-axis value (`y`). The default value is `y`.
- *Final Value*—Final value for the measurement.
- *Percent Low*—Percent low. The default value is `10`.
- *Percent High*—Percent high. The default value is `90`.
- *Number of Occurences*—Specifies whether you want to retrieve only one occurrence of a slewrate event for the given waveform (*single*), or all occurrences of slewrate for the given waveform which you can later plot or print (*multiple*). The default value is `single`.
- *Plot/print vs*—Specifies whether you want to retrieve slewrate risetime data against *time*(or another X-axis parameter for non-transient data) or *cycle*. Cycle numbers refer to the n'th occurrence of the delay event in the input waveform. The default value is `time`. The value in this field is ignored when you specify *Number of Occurrences* as *single*.

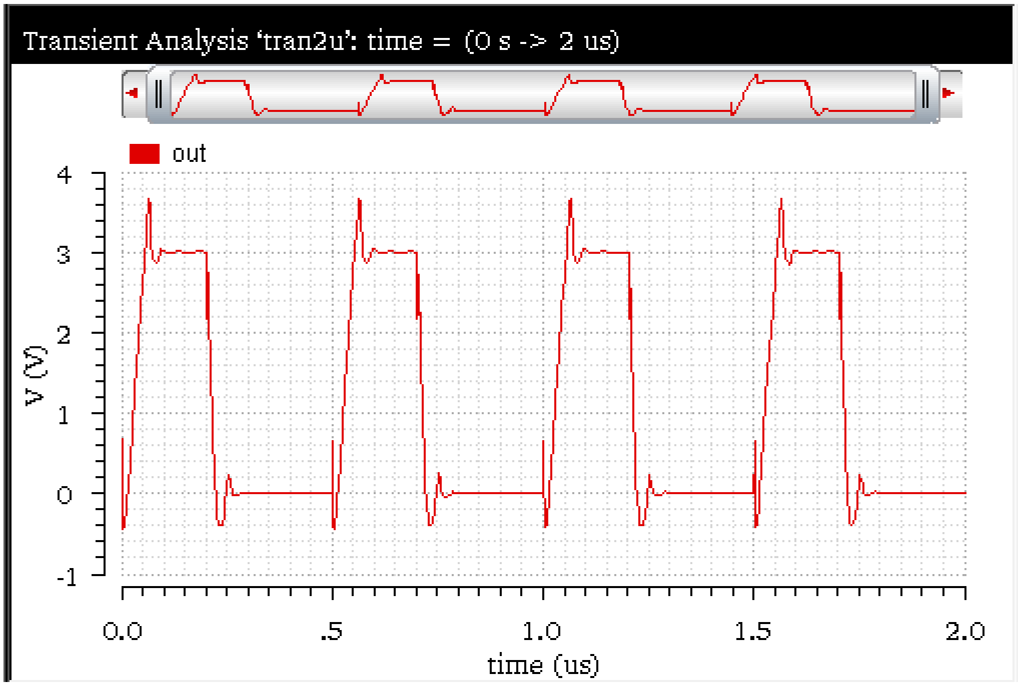
In the MDL mode:

- *sig*—Name of the signal.
- *initval*—Starting value for the measurement.
- *finalval*—Final value for the measurement.
- *inittype*—Specifies whether the initial value is an X-axis (`'x`) or Y-axis value (`'y)`.
- *finaltype*—Specifies whether the final value is an X-axis (`'x`) or Y-axis value (`'y)`
- *theta*—Percent low.
- *theta2*—*P*ercent high.
- *xtol*—Absolute tolerance in the X direction.
- *ytol*—Absolute tolerance in the Y direction.
- *accuracy*—Specifies that the function uses interpolation in the SKILL mode.

  In the MDL mode, *accuracy* specifies whether the function should use interpolation, or use iteration controlled by the absolute tolerances to calculate the value. `'interp` directs the function to use interpolation, and `'exact` directs the function to consider the xtol and yval values.

Example

Consider the following input waveform from transient analysis:



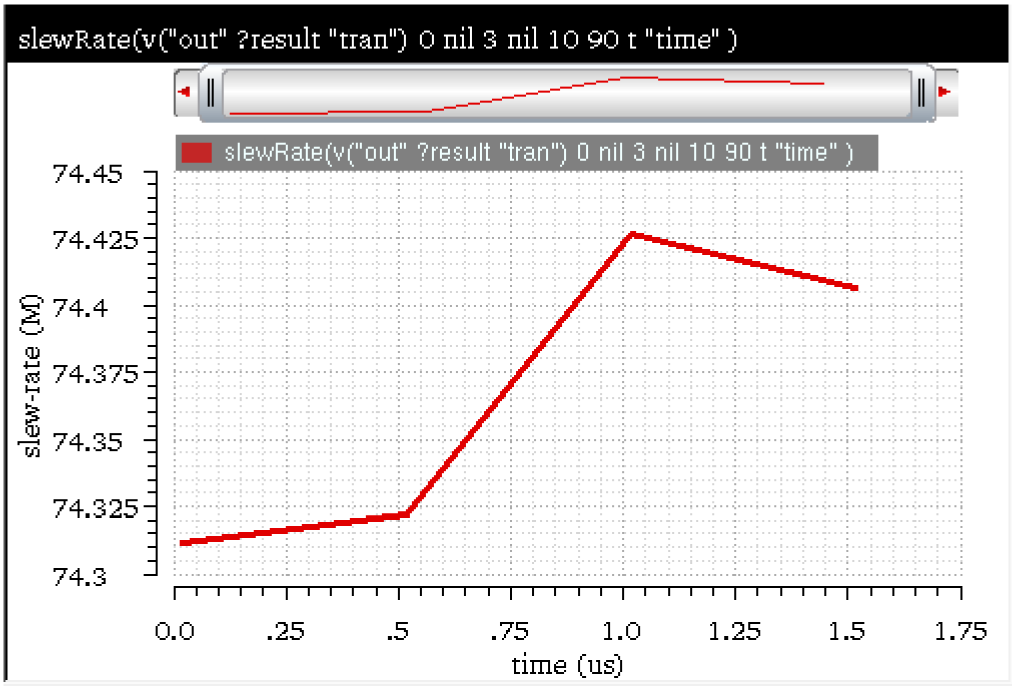
The `slewrate` function is applied on this input waveform with the following arguments:

- *Signal*—`v("out" ?result "tran" ?resultsDir "ampsim.raw")`
- *Initial Value Type*—`y`
- *Initial Value*—`0`
- *Final Value Type*—`y`
- *Final Value*—`3`
- *Percent Low*—`10`
- *Percent High*—`90`
- *Number of occurrences*—`multiple`
- *Plot/print vs.*—`time`

The corresponding expression created in the Buffer is as follows:

```
slewRate(v("out" ?result "tran") 0 nil 3 nil 10 90 t "time" )
```

When you evaluate this expression, the slewrate is calculated and the output waveform is displayed in a new graph window. Note that multiple slewrate values are calculated in this example because the *Number of occurrences* field has been set to `multiplte`.



Related OCEAN Function

The equivalent OCEAN command for `slewRate` is:

```
slewRate( o_waveform n_initVal g_initType n_finalVal g_finalType
     n_theta1 n_theta2
     [g_multiple [s_Xname]][g_histoDisplay][x_noOfHistoBins] )
     => o_waveform/n_value/nil
```

For more information, see `slewRate` in *OCEAN Reference*.

spectralPower

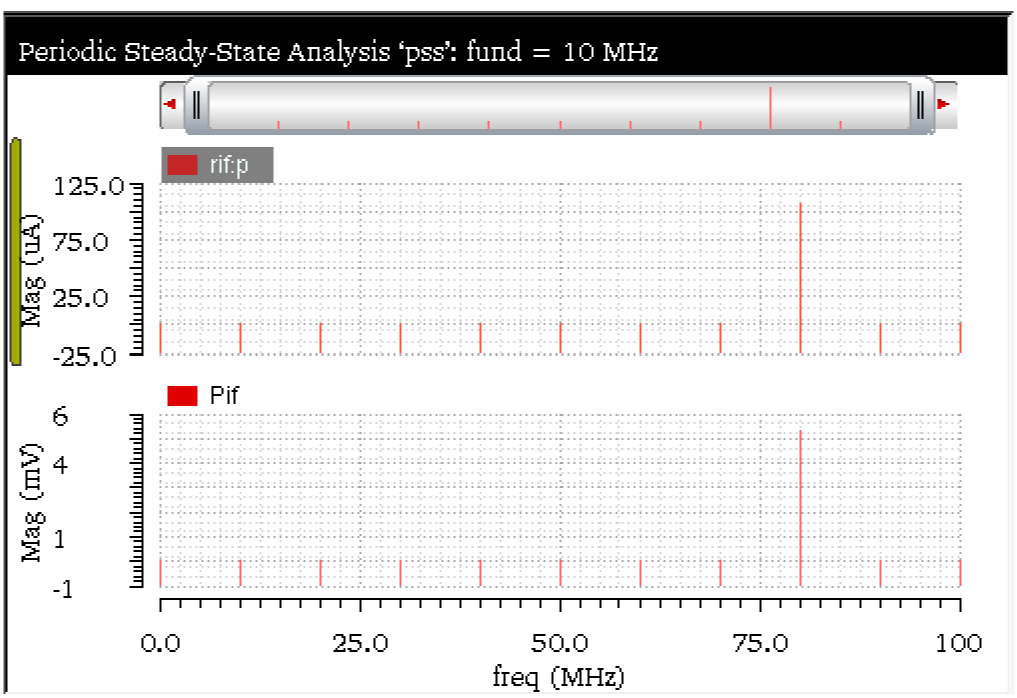
Plots the spectral power for the specified current waveform and voltage waveform. This function is available only in the SKILL mode.

This function includes the following arguments:

- *Current waveform*—Waveform for the current signal which you which you want to calculate the spectral power.
- *Voltage waveform*—Waveform for the voltage signal for which you which you want to calculate the spectral power.

Example

Consider the following input waveforms from PSS analysis:



When you send these voltage and current waveforms to the Calculator, the waveform expression that is created in the Buffer is based on their magnitude, as shown below:

```
mag(i("rif:p" ?result "pss_fd" ?resultsDir "./pss_ne600.raw"))
```

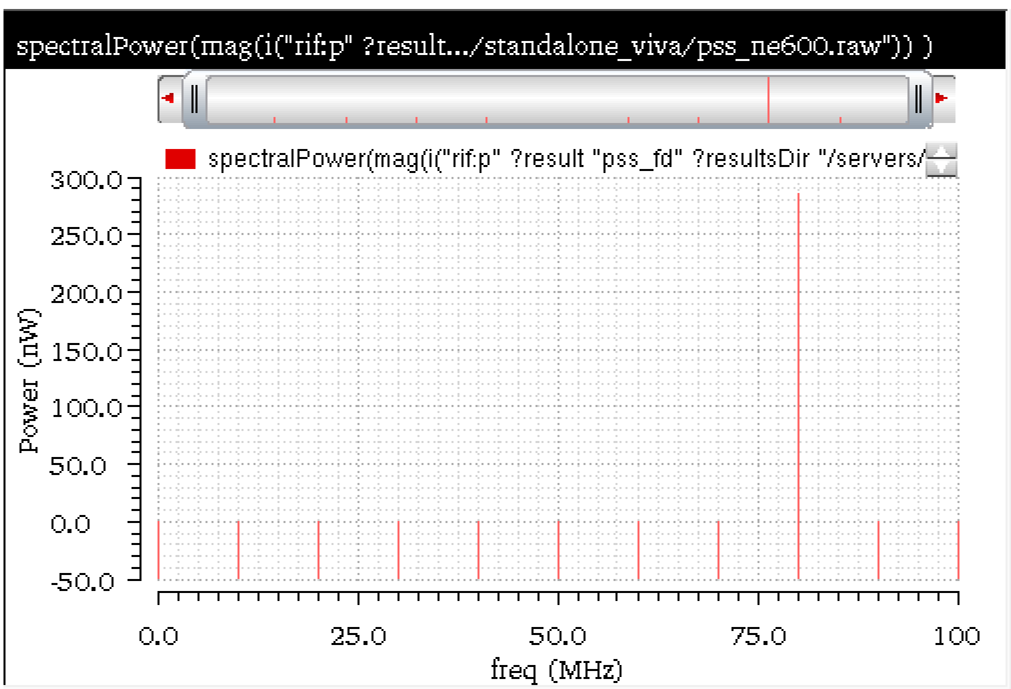
Edit this expression to remove the magnitude and apply the `spectralPower` function with the following arguments:

- *Current Waveform*—`i("rif:p" ?result "pss_fd" ?resultsDir "./pss_ne600.raw")`
- *Voltage Waveform*—`v("Pif" ?result "pss_fd" ?resultsDir "./pss_ne600.raw")`

The corresponding expression created in the Buffer is as follows:

```
spectralPower(i("rif:p" ?result "pss_fd" ?resultsDir "./
pss_ne600.raw") v("Pif" ?result "pss_fd" ?resultsDir "./
pss_ne600.raw") )
```

When you evaluate this expression, the spectral power for these voltage and current signals is calculated and the following output waveform is displayed in the graph window:



Related OCEAN Function

The equivalent OCEAN command for `spectralPower` is:

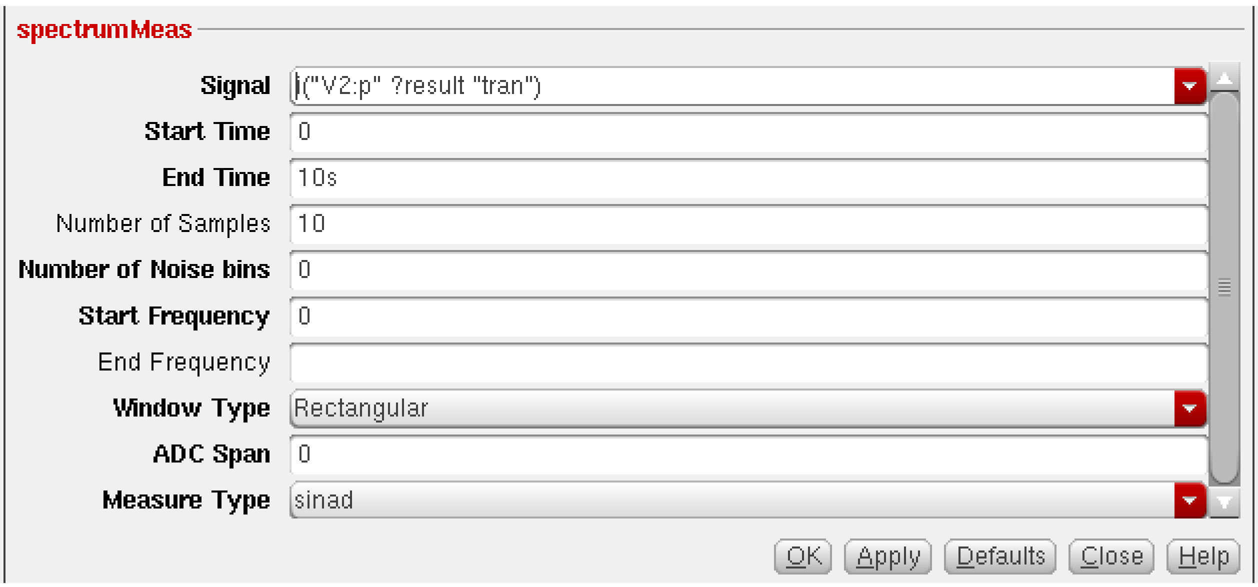
```
spectralPower( o_current o_voltage )
     => o_power/nil
```

For more information about, see `spectralPower` in *OCEAN Reference*.

spectrumMeas

Calculates Signal-to-Noise-and-Distortion Ratio (SINAD), Spurious Free Dynamic Range (SFDR), Effective Number of Bits (ENOB), and Signal-to-Noise Ratio (without distortion) by using discrete Fourier transform of the clipped portion of any given input signal.

The spectrum measure is used for characterizing A-to-D converters and is typically supported for transient simulation data.This function is available only in the SKILL mode



This function includes the following arguments:

- *Signal*—Signal to be measured.
- *Start Time*—Time to start clipping the signal in time domain.
- *End time*—Time to end clipping in time domain.
- *Number of Samples*—Number of sampled points used for the FFT.
  Valid values: Any integer power of two greater than zero. For a value that is not a power of two, the function rounds it up to the next closest power of two.
  Default value: Number of data points in the *Signal*.
- *Number of Noise bins*—Number of noise bins where the size of one bin is the reciprocal of the data window width. For example, 1 ms of transient data creates a bin size of 1 kHz.
  Valid values: Any integer power of two greater than or equal to zero.
  Default value: `0`, implying that no signal is spilling into the bins. A frequency band of bin-size times the number of bins is calculated and adjusted as a function of the selected

window. Frequency components in each band to the left and right of the fundamental or the harmonics are set to zero and do not contribute to any output result.

- *Start Frequency*—Lower limit of frequency range for the spectrum measures. Default value: First frequency point of the FFT.
- *End Frequency*—Upper limit of frequency range for the spectrum measures. Default value: Last frequency point of the FFT.
- *Window Type*—Windowing function applied to the input waveform.
  Valid values: `Blackman`, `Cosine2`, `Cosine4`, `ExtCosBell`, `HalfCycleSine`, `HalfCycleSine3`, `HalfCycleSine6`, `Hanning, Hamming`, `Kaiser`, `Parzen`, `Rectangular`, and `Triangular`.
  Default value: `Rectangular`.
- *ADC Span*—Full-scale span ignoring any DC offsets. This is used in ENOB calculation.
  Valid values: Any floating point number.
  Default value: If *ADC Span* is not specified or is nil, it is assumed to be `0` and is taken to be the peak-to-peak value of the fundamental.
- *Measure Type*—Result specifier.
  Valid values: `sinad`, `sfdr(db)`, `enob`, and `snhr`.
  Default value: `sinad`

Additional Information

The `spectrumMeas` function uses the same algorithm to calculate measurement values as that of the `spectrumMeasurement` SKILL function. The following table displays the mapping in the arguments for `spectrumMeas` and `spectrumMeasurement` functions:

spectrumMeas	spectrumMeasurement	Description
waveform	*waveform*	Specifies the waveform object.
NA	*isTimeWave*	This argument is available only in `spectrumMeasurement` function. The value of this argument is `nil` if the waveform sweep vector is of frequency domain, and the value is `t` if it is of time domain. In `spectrumMeas` function, internally the unit of X-Vector is checked for `Hz` to know whether it is frequency domain or not.

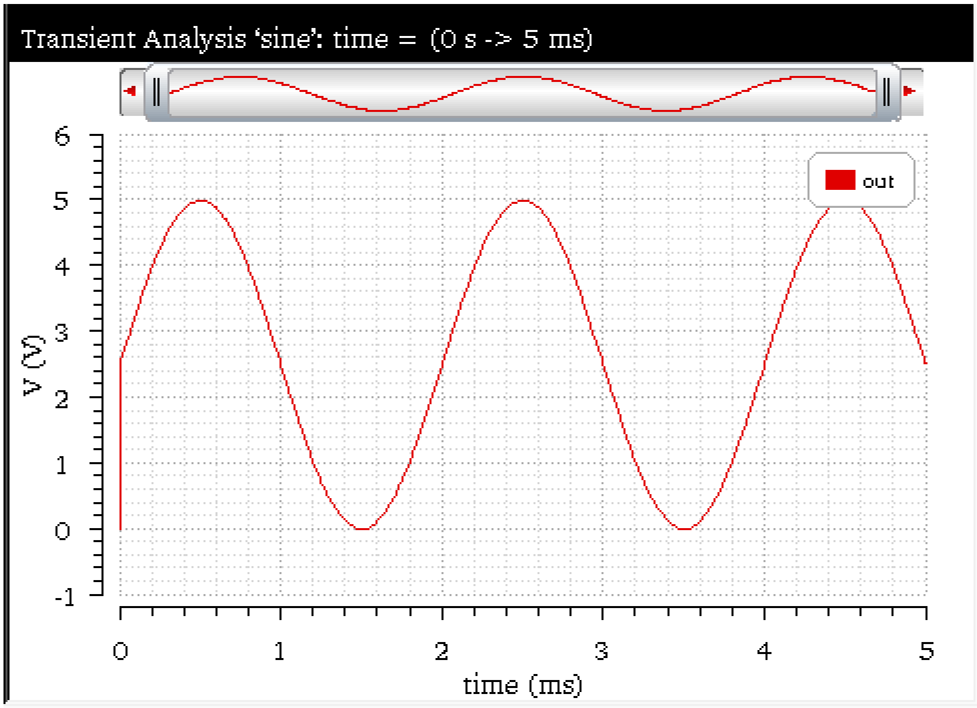
spectrumMeas	spectrumMeasurement	Description
from	*from*	The X-axis start value of the portion of input *o_waveform* to be used for FFT and subsequent calculations.
to	*to*	The X-axis end value of the portion of input *o_waveform* to be used for FFT and subsequent calculations.
numSamples	*numSamples*	Number of sampled points used for the FFT. Valid values: Any integer power of two greater than zero. For a value that is not a power of two, the function rounds it up to the next closest power of two. Default value: Number of data points in the *Signal*.
noiseBins	*signalBins*	In `spectrumMeas`, *Number of Noise bins* is the number of noise bins where the size of one bin is the reciprocal of the data window width. For example, 1 ms of transient data creates a bin size of 1 kHz. Valid values: Any integer power of two greater than or equal to zero. Default value: `0`, implying that no signal is spilling into the bins In `spectrumMeasurement`, *signalBins* specifies the number of signal bins. When you select a window type, this field displays the default number of bins for the selected window type. Default value: `0` to indicate the rectangular window type.
startFreq	*startFreq*	Lower limit of frequency range for the spectrum measures. Default value: First frequency point of the FFT.
endFreq	*endFreq*	Upper limit of frequency range for the spectrum measures. Default value: Last frequency point of the FFT.

spectrumMeas	spectrumMeasurement	Description
windowName	*windowName*	Windowing function applied to *o_wave* while applying the FFT for measurement calculations. Valid values: Blackman, Cosine2, Cosine4, ExtCosBell, HalfCycleSine, HalfCycleSine3, HalfCycleSine6, Hanning, Hamming, Kaiser, Parzen, Rectangular, and Triangular. Default value: Rectangular
adcSpan	*satLvl*	In spectrumMeas, *ADC Span* is the full-scale span ignoring any DC offsets. This is used in ENOB calculation. Valid values: Any floating point number. In spectrumMeasurement, *satLvl* specifies the peak saturation level of the FFT waveform. Magnitude of the FFT wave is divided by the Peak Sat Level before using it in calculations. Peak sat level is the full-scale span ignoring any DC offsets and used in ENOB calculation. Valid values: Any floating point number.
NA	*isNoiseAnalysis*	This argument is present only in the spectrumMeasurement function. It specifies whether the analysis type is Noise Analysis.
NA	*noOfHarmonics*	This argument is available only in spectrumMeasurement function. This specifies the number of harmonics for the waveform that you want to plot. Default value: 1

spectrumMeas	spectrumMeasurement	Description
measType	*measType*	Result specifier. This argument is common for both the functions, but includes the following differences: ■ `sfdr(db)` of `spectrumMeas` is same as `sfdr` of `spectrumMeasurement` or Spectrum assistant ■ `snhr` of `spectrumMeas` is same as `snr` of `spectrumMeasurement` or Spectrum assistant. ■ `spectrumMeas` supports the following measurements—`sinad`, `sfdr(db)`, v, `enob`, `thd`. However, `spectrumMeasurement` supports more measurements in addition to the measurements supported by `spectrumMeas`.

Example

Consider the following input waveform from a transient analysis:



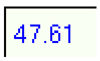
The `spectrumMeas` function is applied on this waveform with the following arguments:

- *Signal*—`v("out" ?result "tran" ?resultsDir "./mixed/test/adc_8bit_ideal_1.raw")`
- *Start Time*—`2.48m`
- *End Time*—`4.48m`
- *Number of Samples*—`65536`
- *Number of Noise bins*—`0`
- *Start Frequency*—`0`
- *End Frequency*—<blank>
- *Window Type*—`Rectangular`
- *ADC Span*—`0`
- *Measure Type*—`sinad`

The expression created in the Buffer is as follows:

```
spectrumMeas(v("out" ?result "tran" ?resultsDir "./mixed/test/
adc_8bit_ideal_1.raw") 2.48m 4.48m 65536 0 0 nil "Rectangular" 0
"sinad")
```

When you evaluate this expression, the following output value is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN command for `spectrumMeas` is:

```
spectrumMeas( o_waveform n_from n_to x_numSamples x_noiseBins n_startFreq
     n_endFreq t_windowName n_adcSpan t_measType )
     => o_spectrumWaveform/g_value/nil
```

For more information, see `spectrumMeas` in *OCEAN Reference*.

stddev

Computes the standard deviation of a waveform (or a family of waveforms) over its entire range. Standard deviation (stddev) is defined as the square-root of the variance where variance is the integral of the square of the difference of the expression f(x) from average (f(x)), divided by the range of x.This function is available only in the SKILL mode.

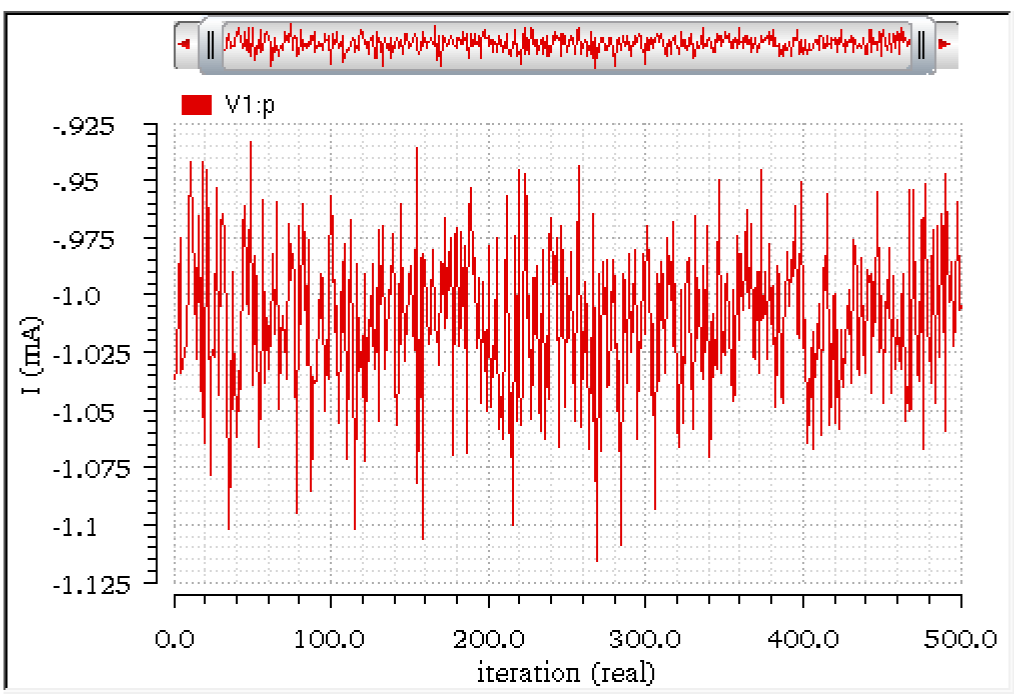
For example, if `y=f(x)`

$$stddev(y) = \frac{\int_{from}^{to} (y - average(y))^2}{to - from}$$

If you want a different range, use the clip function to clip the waveform to the range you want.

Example

Consider the following input waveform from the Monte Carlo results:



The expression for this signal is displayed in the Buffer. Now, when you select the `stddev` function from the Function Panel, the function is applied to the signal expression in the Buffer. The corresponding expression created in the Buffer is as follows:

```
stddev(i("V1:p" ?result "dcOp" ?resultsDir "./monte.out/monte.raw"))
```

When you evaluate this expression, the standard deviation is calculated and the following output is displayed in the Buffer:

32.51E-6

Related OCEAN Function

The equivalent OCEAN command for `stddev` is:

```
stddev( o_waveform )
     => n_stddev/o_waveformStddev/nil
```

For more information, see `stddev` in *OCEAN Reference*.

tangent

(Tangent Line)

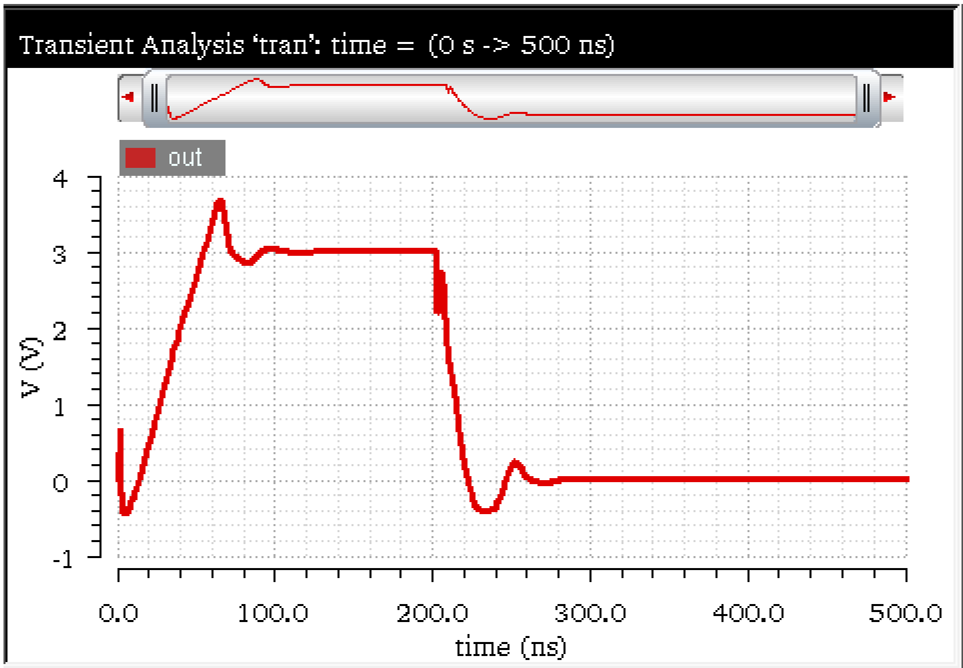
Plots a line that passes through x and y coordinates and the slope that you specify. This function is available only in the SKILL mode.

This function includes the following fields:

- *Signal*—Name of the signal.
- *X Point*—X-axis value you specify.
- *Y Point*—Y-axis value you specify.
- *Slope*—Slope you specify.

Example

Consider the following input waveform from the transient analysis:



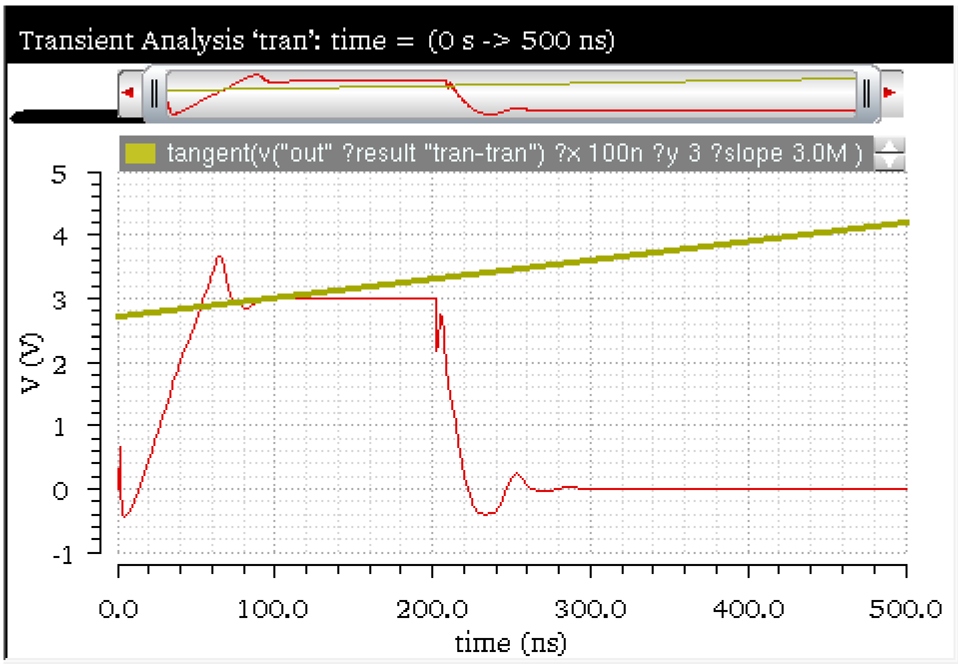
The tangent function is applied on this input waveform with the following arguments:

- *Signal*—`v("out" ?result "tran-tran" ?resultsDir "ampsim.raw")`
- *X Point*—`100n`
- *Y Point*—`3`
- *Slope*—`3.0M`

The corresponding expression created in the Buffer is as follows:

```
tangent(v("out" ?result "tran-tran" ?resultsDir "ampsim.raw") ?x 100n
?y 3 ?slope 3.0M )
```

When you evaluate this expression and plot results in the append mode, a tangent line is displayed in the graph as shown in the figure below:



Related OCEAN Function

The equivalent OCEAN command for `tangent` is:

```
tangent( o_waveform [ ?x n_x ] [ ?y n_y ] [ ?slope n_slope ] )
     => o_waveform/nil
```

For more information, see tangent in *OCEAN Reference*.

thd

(Total Harmonic Distortion)

Computes the percentage of THD of a signal with respect to the fundamental frequency and is expressed as a voltage percentage.

This function is available only in the SKILL mode.

- *Signal*—Name of the signal.
- *From*—Starting time for the DFT sample window.
- *To*—Ending time of the DFT sample window.
- *Number* of Samples—Number of time domain points to be used.
- *Fundamental (Hz)*—Fundamental frequency of the signal.

Additional Information

The computation uses the dft function. Assume that the *dft* function returns complex coefficients A0, A1..., A*f* , ... Note that fundamental frequency *f* is the frequency contributing to the largest power in the signal. A0 is the complex coefficient for the DC component and Ai is the complex coefficient for the *i*th harmonic where *i≠0,f*. Then, total harmonic distortion is computed as:

$$THD = \frac{\sqrt{\sum_{i = 1,\ i \neq 0}^{f} |A_i|^2}}{|A_f|} \times 100\%$$

The accuracy of the total harmonic distortion measurement depends on simulator options and the analysis parameters. To view the simulator options in the ADE L, do the following:

- Choose *Simulation – Options – Analog*.

  The *Simulator Options* form appears.

If you are using the Spectre simulator, the options listed in the table below are displayed in the main tab of the Simulator Options form. For an accurate measurement, set the following simulation options:

Option	**Description**
reltol	Relative convergence criterion. Default value is `1e-3`
residualtol	Tolerance ratio for residual (multiplies reltol).
vabstol	Voltage absolute tolerance convergence criterion. Default value is `1e-6`
iabstol	Current absolute tolerance convergence criterion. Default value is `1e-12`

Set the option `strobeperiod` to the period of the signal to be analyzed divided by `128`, and simulate enough cycles of this signal so that circuit can reach a steady-state. If that occurs in the 10th cycle, end the simulation just after the 10th cycle.

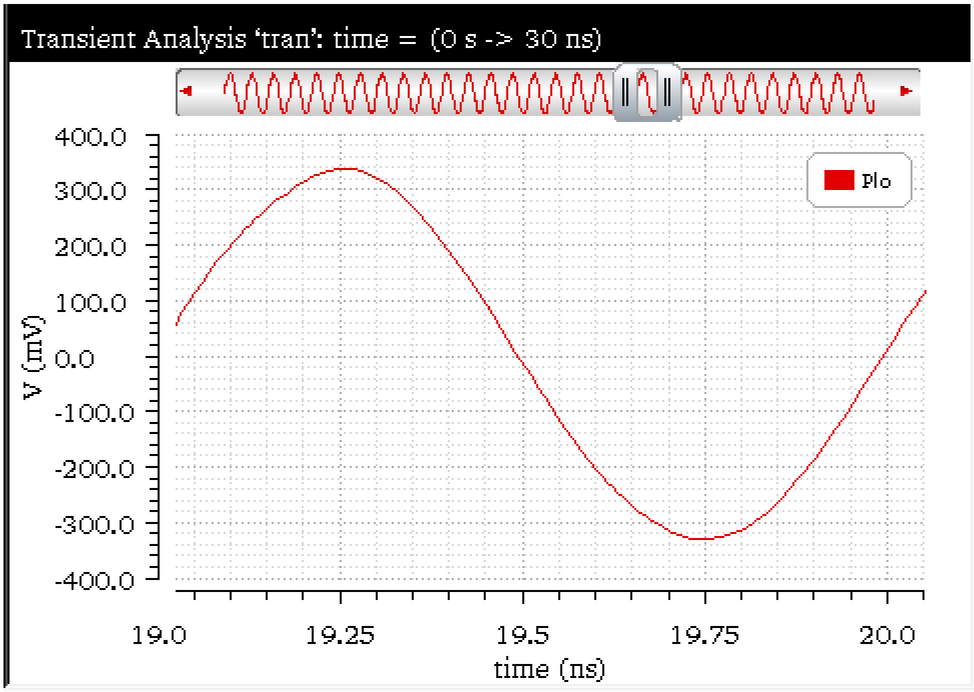
For the spectre, and some other simulator interfaces, the Options buttons in each Choosing Analyses form let you set options that apply to the specific analysis. For more information, see Chapter 6, *Running a Simulation*, in *Virtuoso Analog Design Environment L User Guide*.

If you are using the HSPICE simulator, the following Tolerance options are displayed:

Option	**Suggested Value**
reltol	`1e-5`
abstol	`1e-13`
vntol	`3e-8`
trtol	`1`
method	`gear`
maxord	`3`

Example

This example displays the total harmonic value generated when you apply the `thd` function on the following input waveform:



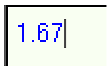
The following arguments are specified in this example:

- *Signal*—`v("Plo" ?result "tran")`
- *From*—`19n`
- *To*—`20n`
- *Number of Samples*—`8192`
- *Fundamental (HZ)*—`1G`

The following expression is created in the Buffer:

```
thd(v("Plo" ?result "tran" ?resultsDir "./pss_ne600.raw") 19n 20n
8192 1G )
```

When you evaluate this expression, the output value generated is displayed in the Buffer (as shown in the figure below):



Related OCEAN Function

The equivalent OCEAN function for `thd` is:

```
thd( o_waveform n_from n_to x_num n_fund)
     => o_waveform/n_thdValue/nil
```

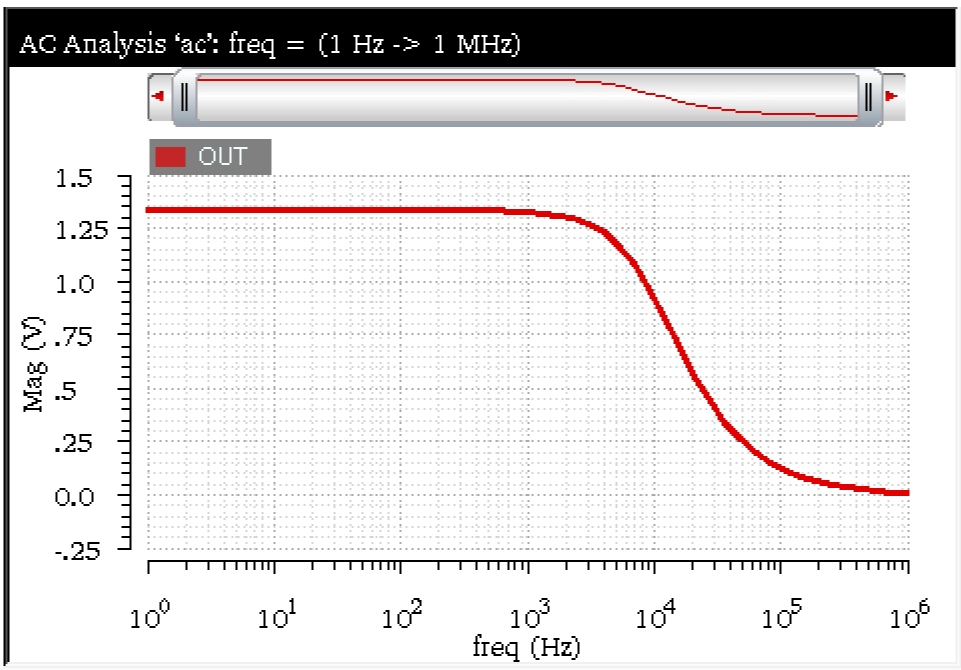
For more information about the OCEAN function, see thd in *OCEAN Reference*.

unityGainFreq

Computes and reports the frequency at which the gain is unity. This function is available only in the SKILL mode.

Example

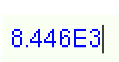
Consider the following input waveform from AC analysis:



The expression for this signal is displayed in the Buffer. Now, when you select the `unityGainFreq` function from the Function Panel, the function is applied to the signal expression in the Buffer. The corresponding expression created in the Buffer is as follows:

```
unityGainFreq(v("OUT" ?result "ac" ?resultsDir "./aExamples.raw"))
```

When you evaluate this expression, the following output value indicating the frequency is displayed in the Buffer:



Related OCEAN Function

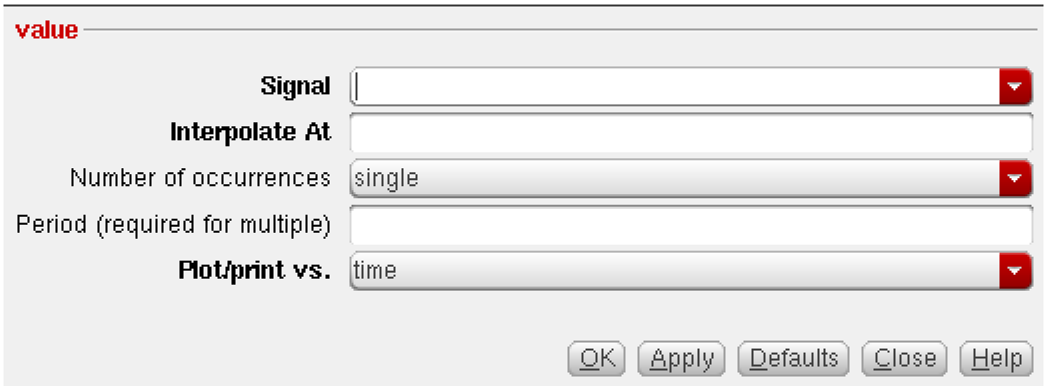
The equivalent OCEAN command for `unityGainFreq` is:

```
unityGainFreq( o_gainFreqWaveform )
     => n_frequency/nil
```

For more information, see `unityGainFreq` in *OCEAN Reference*.

value

Computes the Y-axis value of the waveform at the specified X-axis value.



This function includes the following fields:

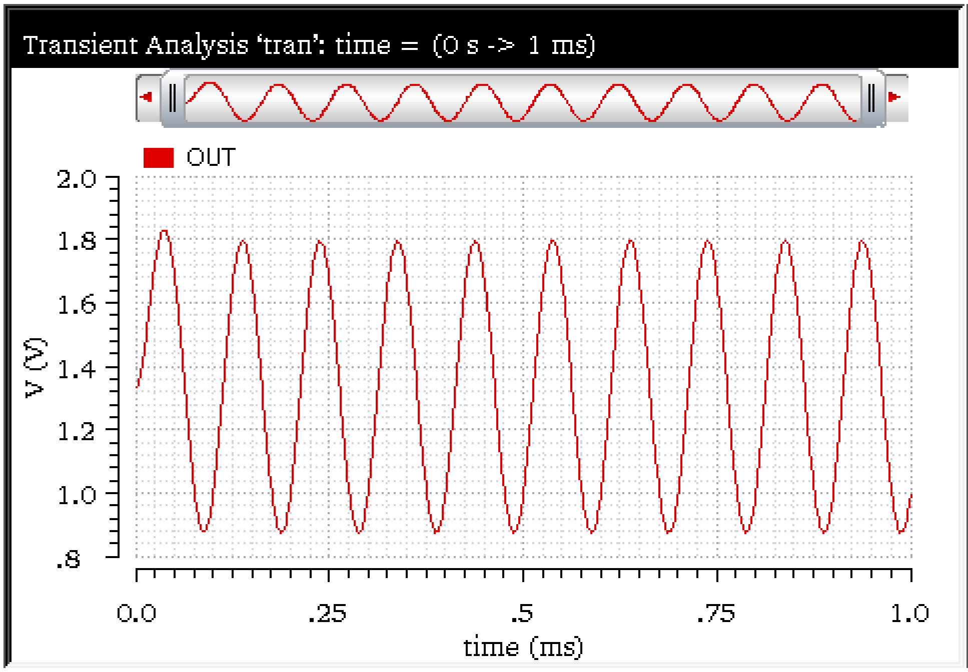
- *Signal*—Name of the signal.
- *Interpolate At*—X-axis value for which you want the Y-axis value to be computed.
- *Number of occurrences*—Specifies whether you want to retrieve only one occurrence of an interpolated value for the given waveform (`single`) or all interpolated values for the given waveform (`multiple`), which you can later plot or print. The default value is `single`.
- *Period (required for multiple)*—Time between samples that result in the multiple Y values. You must specify this if you selected multiple in the *Number of Occurrences* field.
- *Plot/print vs.*—Specifies whether you want to retrieve value data against time (or another X-axis parameter. The default value is `time`.

Additional Information

The X-axis interpolation is linear across the independent axis of the waveform.

Example

Consider the following input waveform from transient analysis:



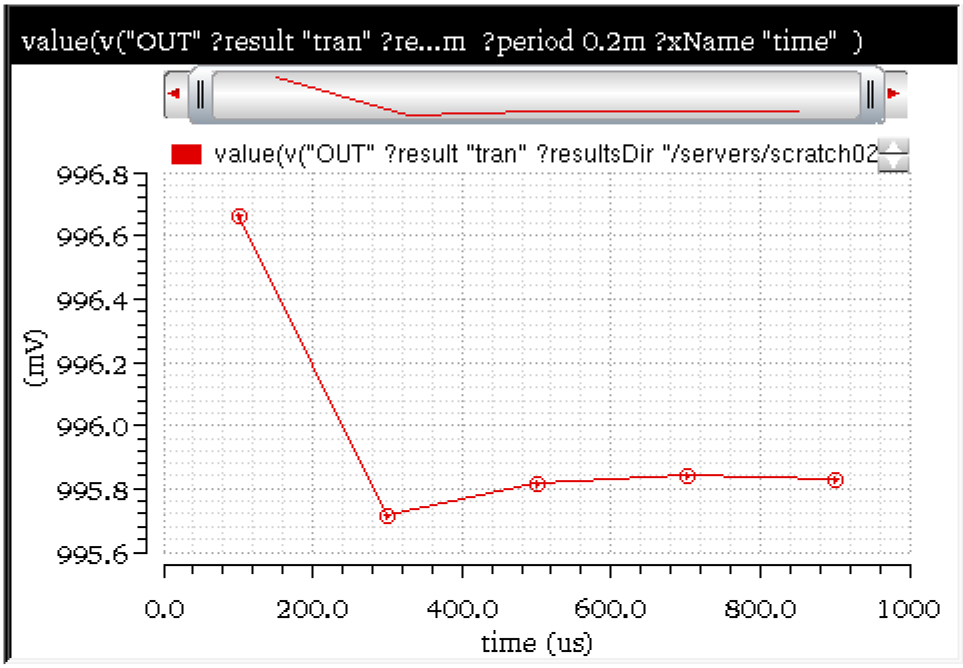
The value function is applied on this input signal with the following arguments:

- *Signal*—`v("OUT" ?result "tran" ?resultsDir "./aExamples.raw")`
- *Interpolate At*—`01.m`
- *Number of occurrences*—`multiple`
- *Period (required for multiple)*—`0.2m`
- *Plot/print vs.*—`time`

The corresponding expression created in the Buffer is as follows:

```
value(v("OUT" ?result "tran" ?resultsDir "./aExamples.raw") 0.1m
?period 0.2m ?xName "time" )
```

When you evaluate this expression, the Y points are measured and the following output waveform is displayed in a new graph window (Note that the symbols are enabled in this output waveform).



Related OCEAN Function

The equivalent OCEAN command for `value` is:

```
value( o_waveform [s_name] g_value ?period n_period
     [g_multiple [s_Xname]] [g_histoDisplay][x_noOfHistoBins])
     => o_waveform/g_value/nil
```

For more information, see `value` in *OCEAN Reference*.

xmax

Computes the value of the independent variable x at which the expression attains its maximum value, that is, the value of x that maximizes y=f(x).

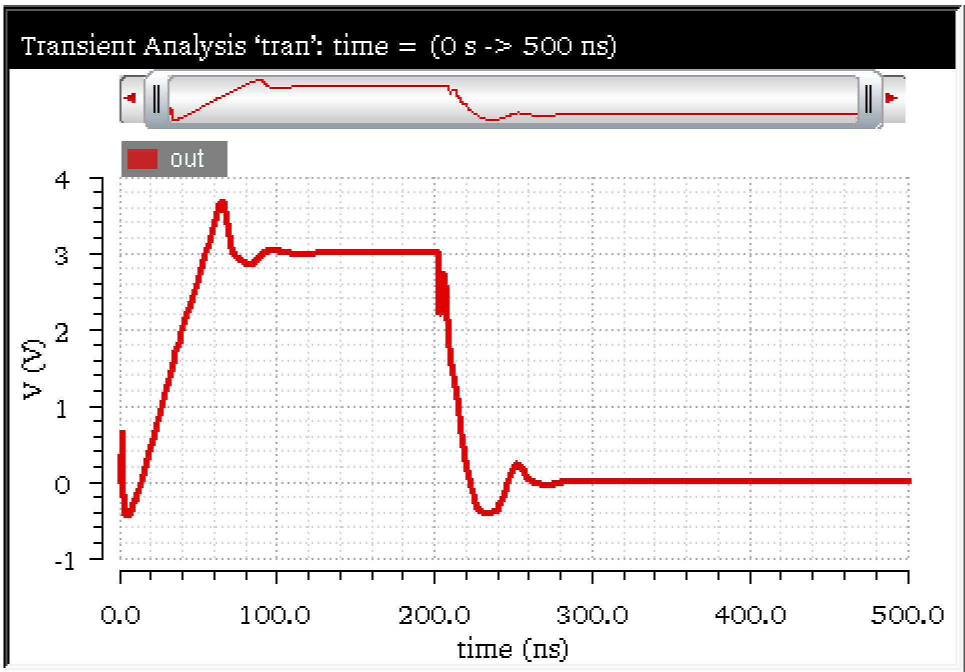
The maximum might occur at more than one point on the X-axis, so you must choose (in the *Nth Maximizer* field) which maximum value you want to see. The Calculator returns the value of the *Nth Maximizer* counting from the left, that is, toward increasing X-axis values. If you enter a negative integer, the direction of search is reversed toward decreasing X-axis values (counting from the right). This function is available only in the SKILL mode

This function includes the following arguments:

- *Signal*—Name of the signal.
- *Nth Maximizer*—Maximum value of the expression.

Example

Consider the following input waveform from the transient analysis:



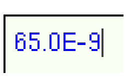
The `xmax` function is applied on this input waveform with the following arguments:

- *Signal*—`v("out" ?result "tran-tran" ?resultsDir “ampsim.raw”)`
- *Nth Maximizer*—`1`

The corresponding expression created in the Buffer is as follows:

```
xmax(v("out" ?result "tran-tran") 1 )
```

When you evaluate this expression, the following output indicating the maximum X-value is displayed in the Buffer.



Related OCEAN Function

The equivalent OCEAN command for `xmax` is:

```
xmax( o_waveform x_numberOfPeaks )
     => o_waveform/g_value/l_value/nil
```

For more information, see `xmax` in *OCEAN Reference*.

xmin

Computes the value of the independent variable x at which the expression has its minimum value, that is, the value of x that minimizes y=f(x).

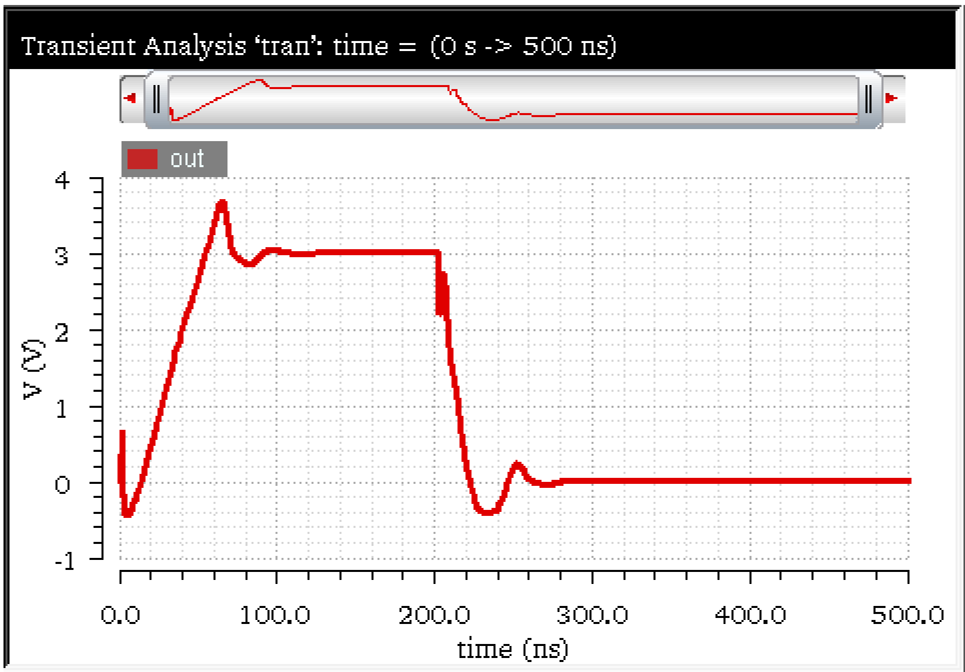
The minimum might occur at more than one point on the x axis, so you must choose (in the *Nth Minimizer* field) which minimum value you want to see. The calculator returns the value of the *Nth Minimizer*, counting from the left, that is, toward increasing X-axis values. If you enter a negative integer, the direction of search is reversed toward decreasing X-axis values (counting from the right). This function is available only in the SKILL mode.

This function includes the following arguments:

- *Signal*—Name of the signal.
- *Nth Minimizer*—Minimum value of the expression.

Example

Consider the following input waveform from the transient analysis:



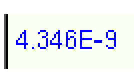
The `xmin` function is applied on this input waveform with the following arguments:

- *Signal*—`v("out" ?result "tran-tran" ?resultsDir "ampsim.raw")`
- *Nth Maximizer*—`1`

The corresponding expression created in the Buffer is as follows:

```
xmin(v("out" ?result "tran-tran") 1 )
```

When you evaluate this expression, the following output indicating the minimum X value is displayed in the Buffer.



Related OCEAN Function

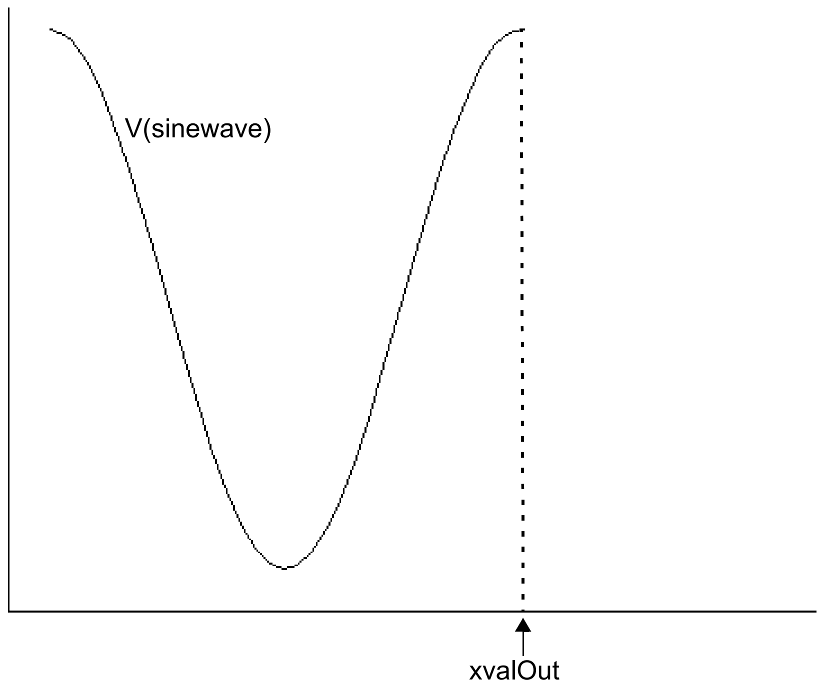
The equivalent OCEAN command for `xmin` is:

```
xmin( o_waveform x_numberOfValleys )
     => o_waveform/g_value/l_value/nil
```

For more information, see `xmin` in *OCEAN Reference*.

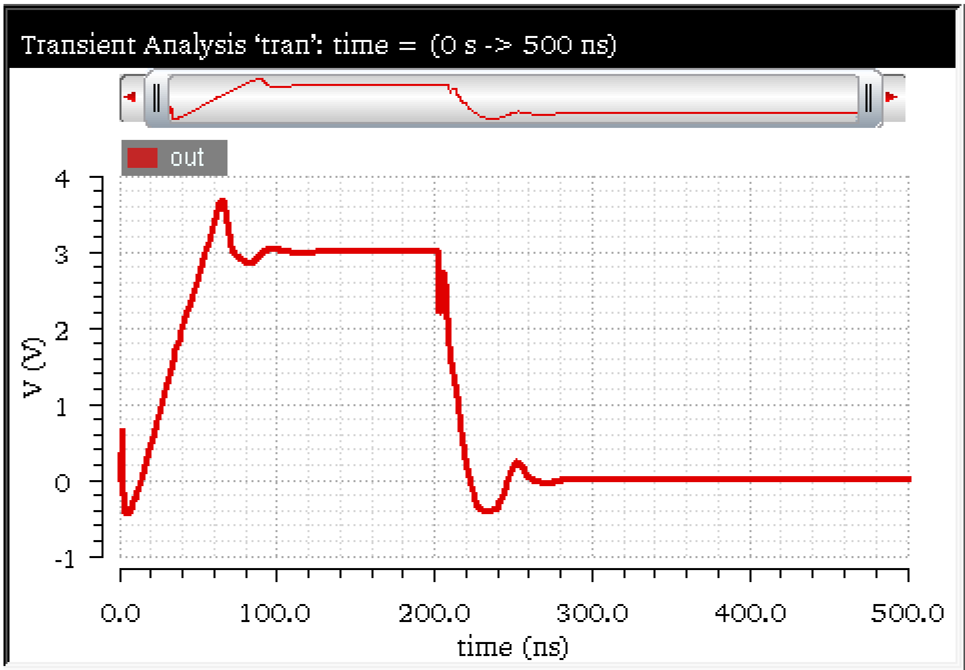
xval

Returns the vector consisting of the X-axis values of the points in the signal. The following figure illustrates this function.



Example

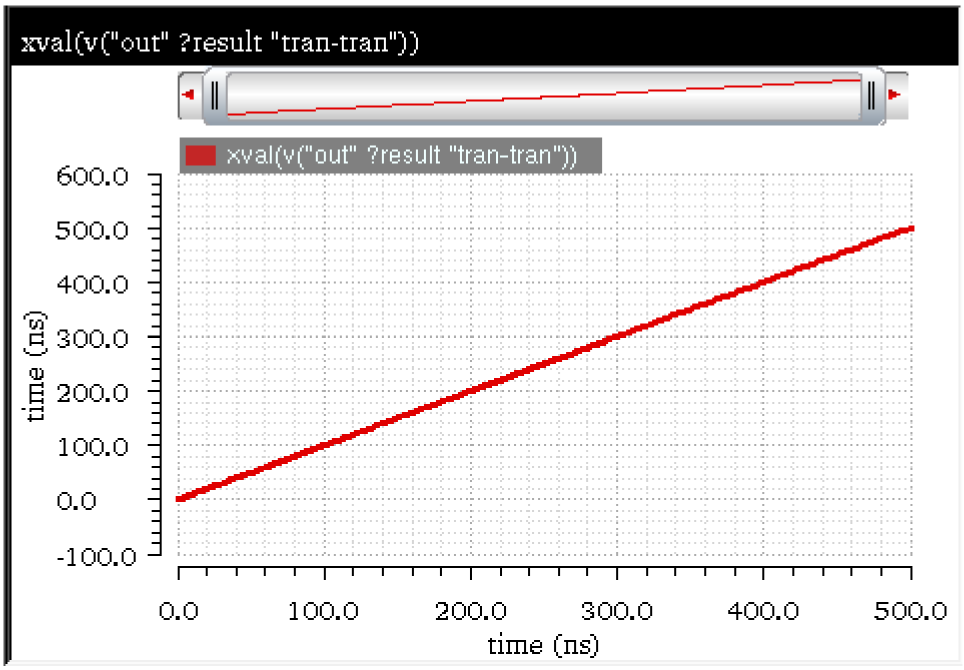
Consider the following input waveform from the transient analysis:



The expression for this input signal is displayed in the Buffer. Now, when you select the `xval` function from the Function Panel, the function is applied to the signal expression in the Buffer. The corresponding expression created in the Buffer is as follows:

```
xval(v("out" ?result "tran-tran"))
```

When you evaluate this expression, the corresponding X vector values are plotted in a new graph window as shown in the figure below:



Related OCEAN Function

The equivalent OCEAN command for `xval` is:

```
xval( o_waveform )
     => o_waveform/nil
```

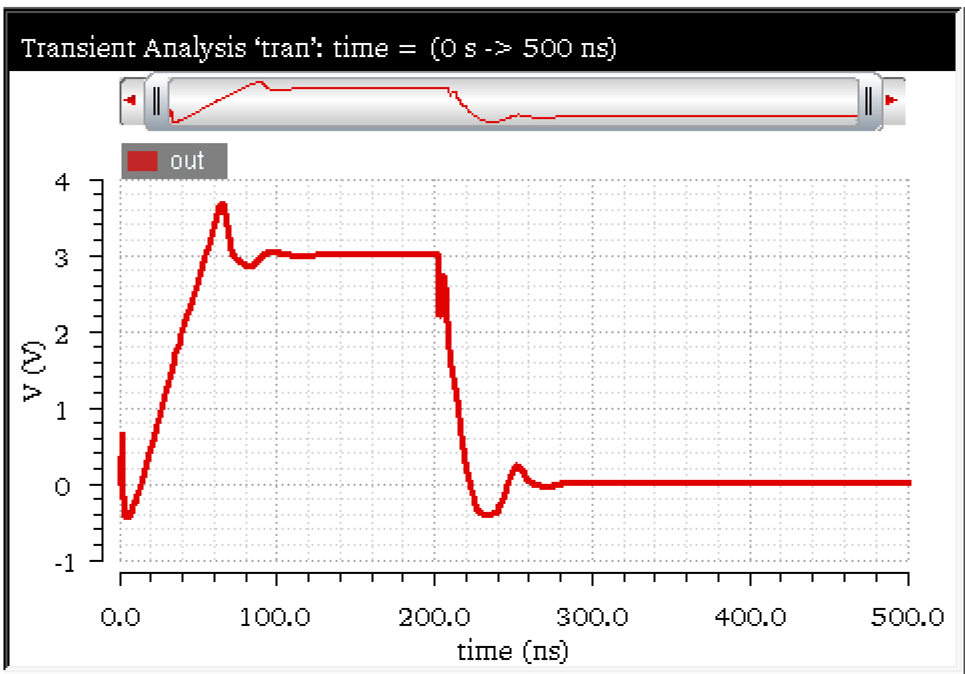
For more information, see `xval` in *OCEAN Reference*.

ymax

Computes the maximum Y-axis value of the expression y=f(x). This function is available only in the SKILL mode.

Example

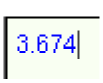
Consider the following input waveform from the transient analysis:



The expression for this input signal is displayed in the Buffer. Now, when you select the `ymax` function from the Function Panel, the function is applied to the signal expression in the Buffer. The corresponding expression is displayed in the Buffer as follows:

```
ymax(v("out" ?result "tran-tran"))
```

When you evaluate this expression, the following output showing maximum Y-axis value is displayed in the Buffer:



Related OCEAN Function

The equivalent OCEAN command for `ymax` is:

```
ymax( o_waveform )
     => n_max/o_waveformMax/nil
```

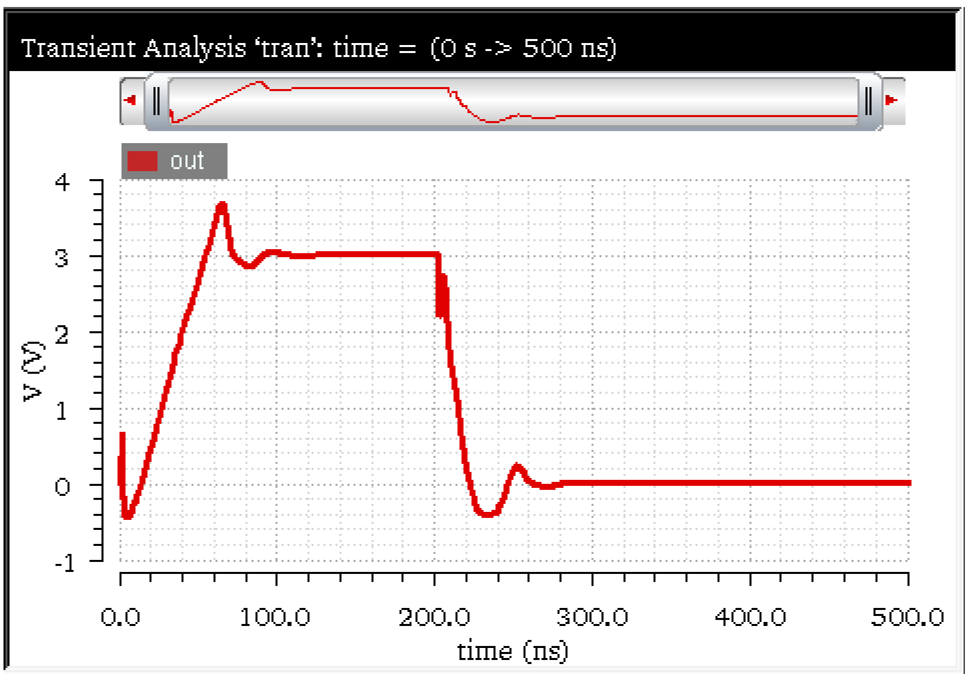
For more information, see `ymax` in *OCEAN Reference*.

ymin

Computes the minimum Y-axis value of the expression y=f(x). This function is available only in the SKILL mode.

Example

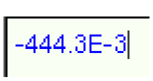
Consider the following input waveform from the transient analysis:



The expression for this input signal is displayed in the Buffer. Now, when you select the `ymin` function from the Function Panel, the function is applied to the expression in the Buffer. The corresponding expression created in the Buffer is as follows:

```
ymin(v("out" ?result "tran-tran"))
```

When you evaluate this expression, the following output showing minimum Y-axis value is displayed in the Buffer:



When you use this function to evaluate data for measurement across corners, a new key argument, overall, is added to this function. If you do not specify a value for this argument, it takes it

Related OCEAN Function

The equivalent OCEAN command for `ymin` is:

```
ymin( o_waveform )
     => n_min/o_waveformMin/nil
```

For more information, see `ymin` in *OCEAN Reference.*

Modifier Functions

This section describes the following modifier functions available in the Virtuoso Visualization and Analysis XL Calculator:

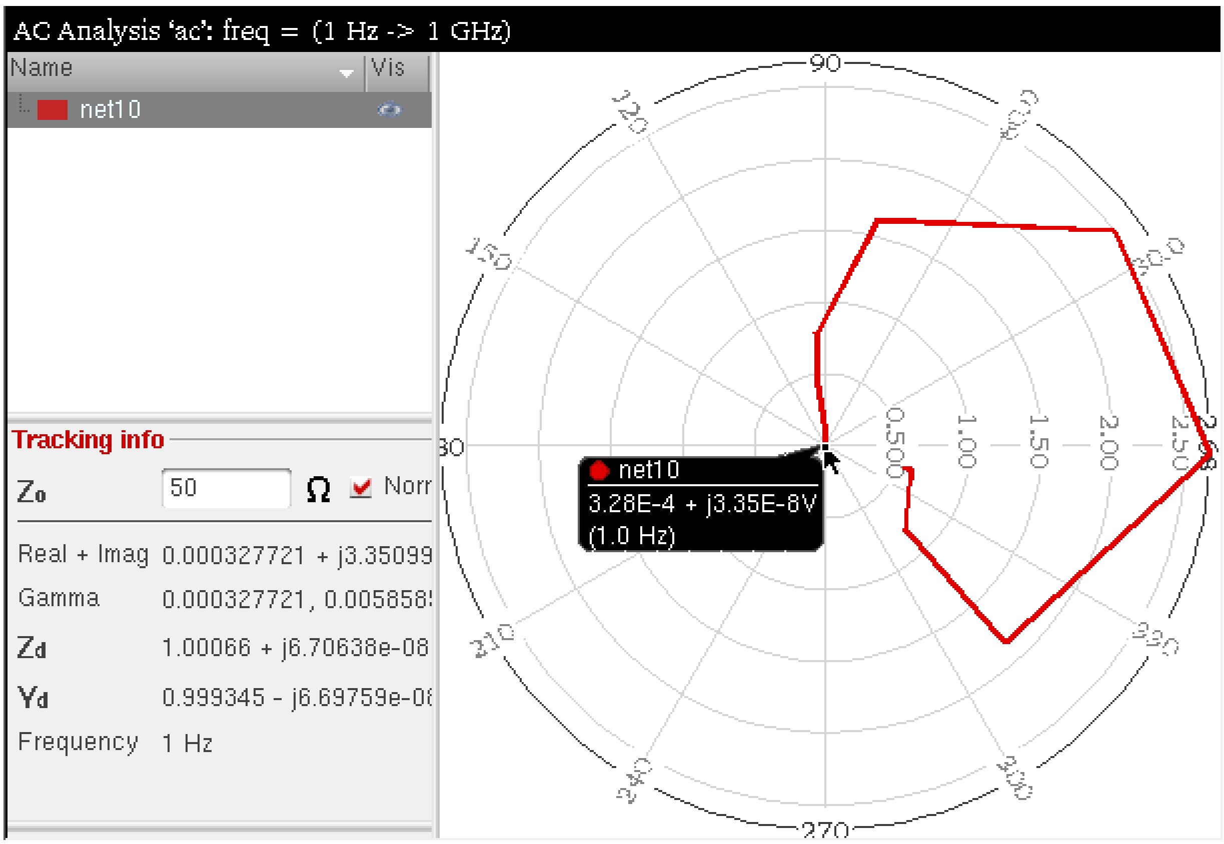
- conjugate
- dB10
- dB20
- imag
- real
- mag
- phase
- phaseRad

conjugate

Returns the conjugate of a complex number. In the conjugate, the magnitudes of real and imaginary parts of the complex conjugate are same as that of the given complex number, and the imaginary part is of the opposite sign. For example, the conjugate of `i+jk` is `i-jk`.

Example

Consider the following signal expression from the AC analysis for the complex data:



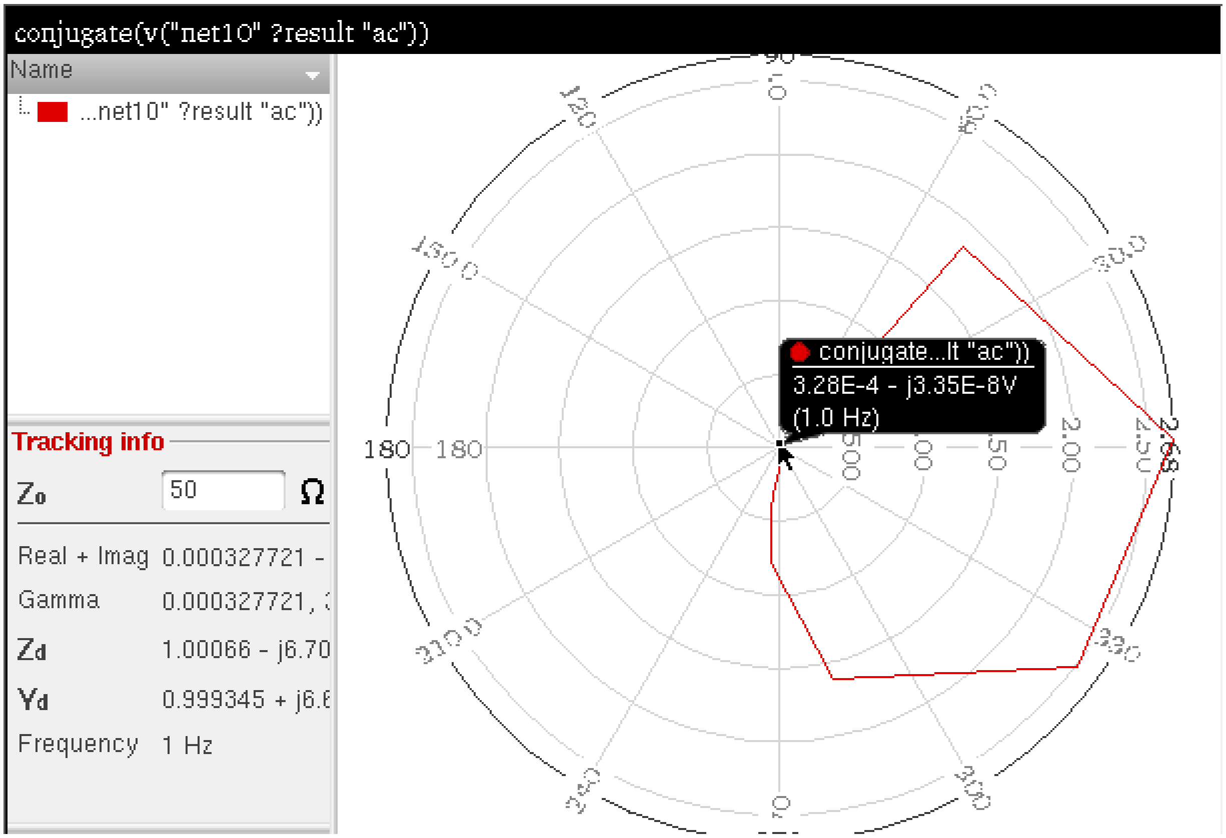
When you send this signal to Calculator, the following expression is displayed in the Buffer:

```
mag(v("net10" ?result "ac" ?resultsDir "./ampsim.raw"))
```

By default, the expression is displayed with the `mag` function. You need to remove this function from the Buffer expression manually When you select the `conjugate` function from the Function Panel, the function is applied on the expression in the Buffer. The expression created in the Buffer is as follows:

```
conjugate(v("net10" ?result "ac" ?resultsDir "./ampsim.raw"))
```

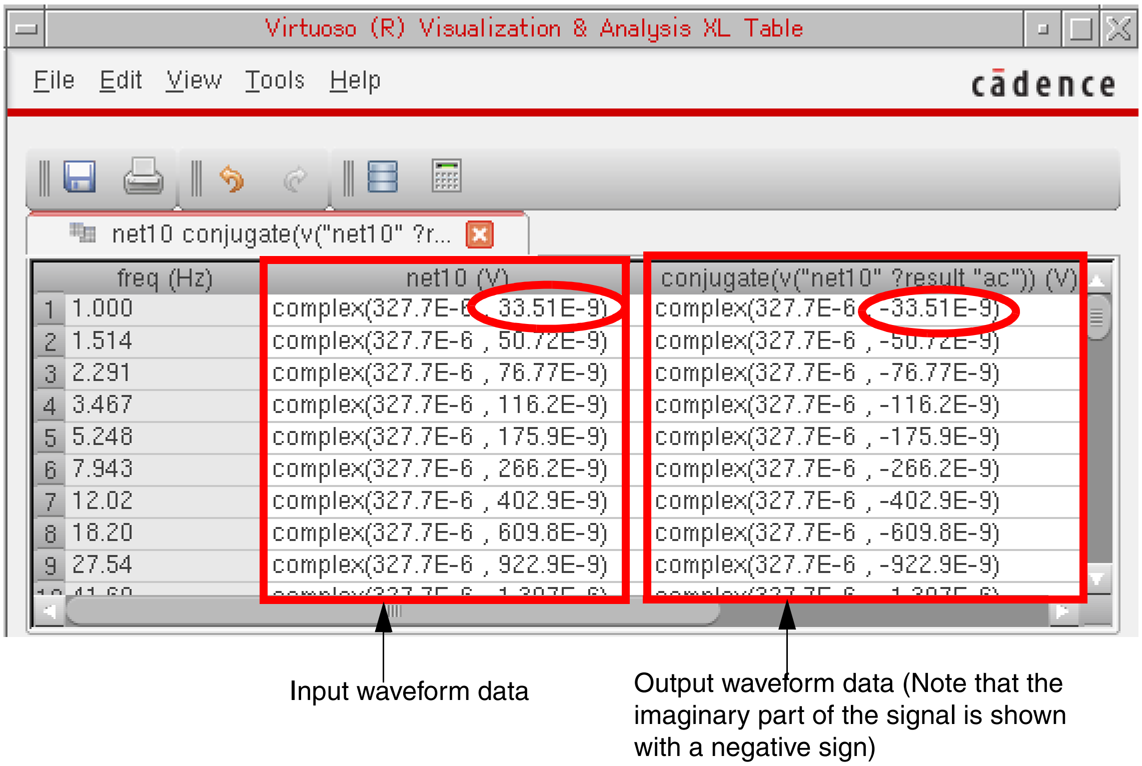
When you evaluate this expression, the following waveform is displayed in a new graph window that represents the conjugate of the input waveform.



To analyze these generated output values, you can display the input and output signals in the Virtuoso Visualization and Analysis XL Table. To send the input signal to the table, right-click the signal and choose *Send To – Table – New Window*. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

To view both the real and imaginary values in the table, right-click the table row and choose *Display Complex As – Real and Imaginary*. The following table contents are displayed when you send the input and output signals described in this example to the table and display the values in the complex number format.

The following figure displays the tabular values for input and output signals.



Related OCEAN Function

The equivalent OCEAN command for `conjugate` is:

```
conjugate( {o_waveform | n_x} )
    => o_waveform/n_y/nil
```

For more information, see `conjugate` in *OCEAN Reference*.

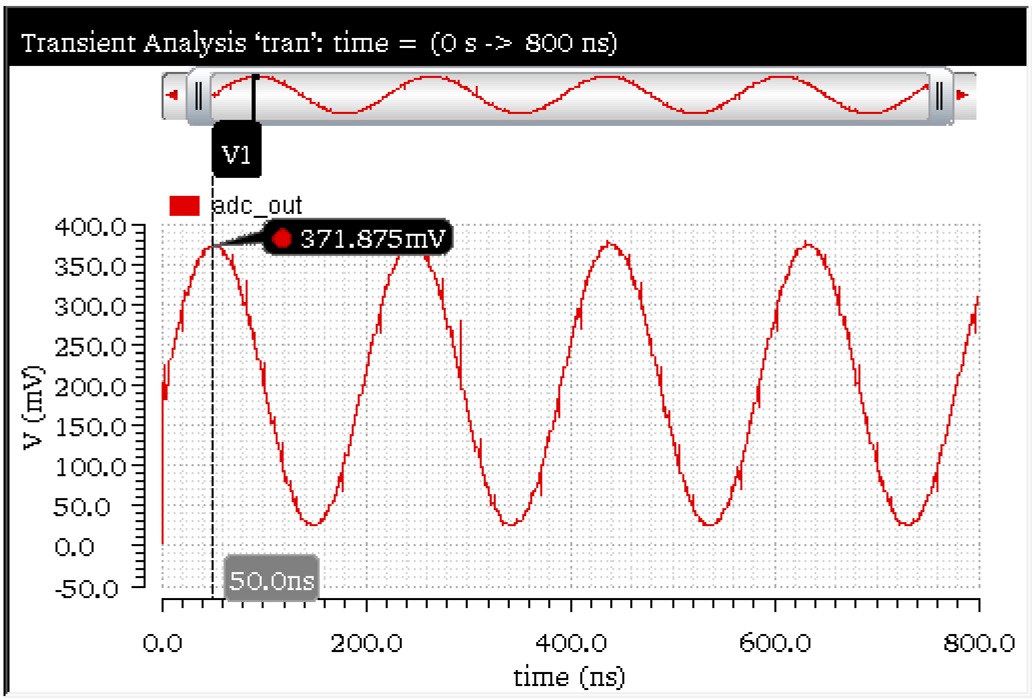
dB10

Converts a signal to dB by using the following equation:

```
db10 =  10/log(10)*log( abs(waveform_yvalue) )
```

Example

Consider the following input signal waveform from the transient analysis:



Note that a vertical marker `V1` is placed at `time=50ns` that displays the voltage value on Y-axis as `371.875mv`.

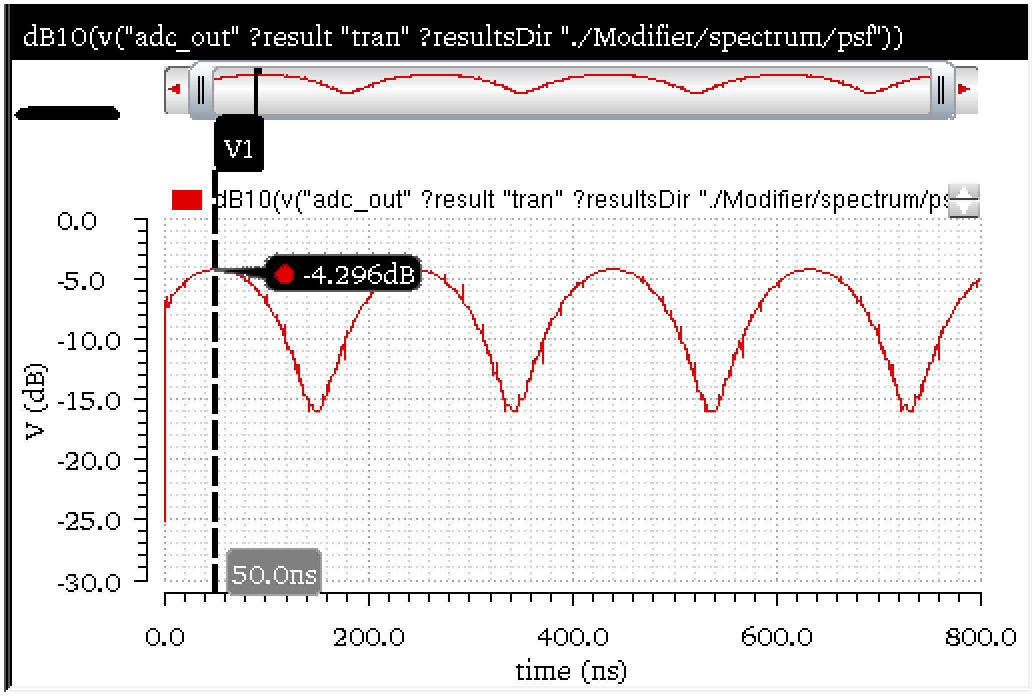
When you send this signal to Calculator, the following expression is displayed in the Buffer:

```
v("adc_out" ?result "tran" ?resultsDir "./Modifier/spectrum/psf")
```

When you select the `dB10` function from the Function Panel, the function is applied on the expression in the Buffer. The expression created in the Buffer is as follows:

```
dB10(v("adc_out" ?result "tran" ?resultsDir "./Modifier/spectrum/
psf"))
```

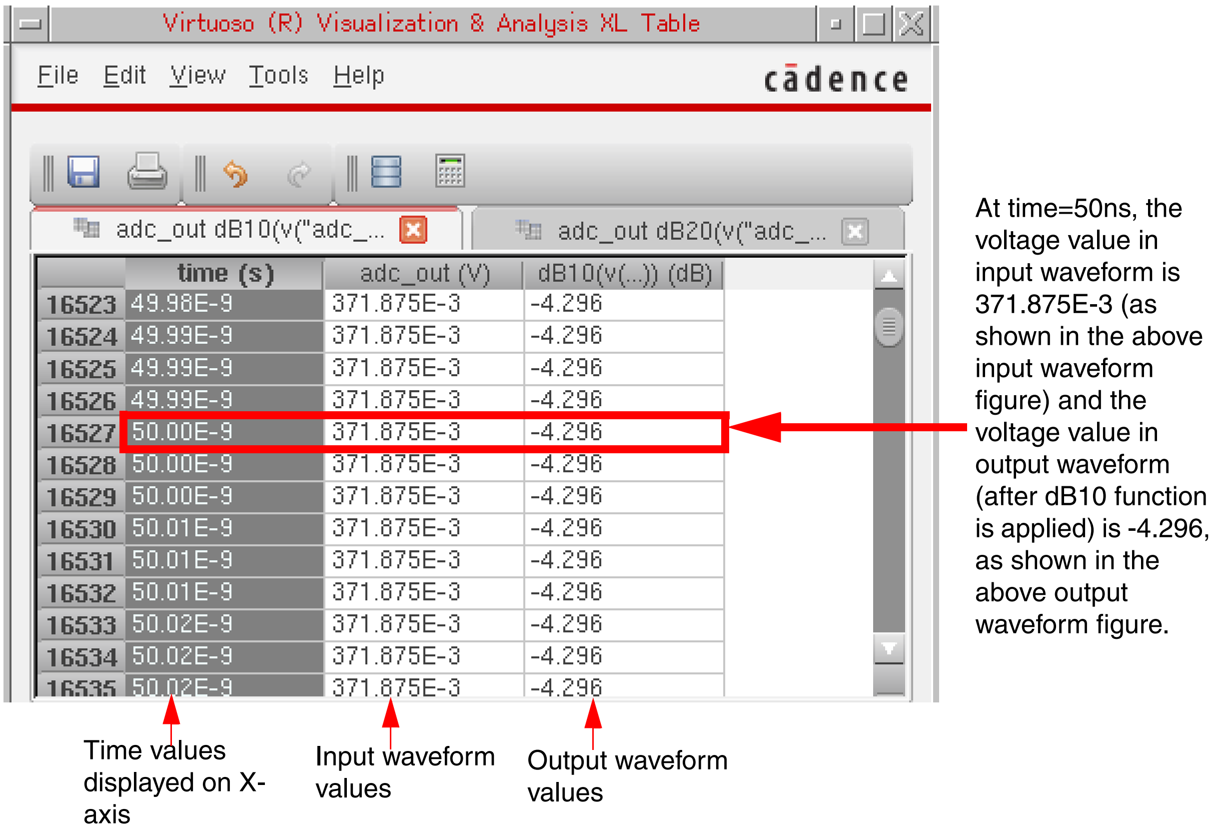
When you evaluate this expression, the following waveform is displayed in a new graph window that represents the dB10 calculation of the input waveform.



Note that a point marker `V2` is placed at `time=50ns` (where the marker `V1` is placed in the input waveform) and after dB10 calculation it displays voltage as `-4.296dB` on the Y-axis.

To analyze these generated output values, you can display the input and output signals in the Virtuoso Visualization and Analysis XL Table. To send the input signal to the table, right-click the signal and choose *Send To – Table – New Window*. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



Related OCEAN Function

The equivalent OCEAN command for `dB10` is:

```
db10( {o_waveform | n_number} )
     => o_waveform/n_number/nil
```

For more information, see `dB10` in *OCEAN Reference*.

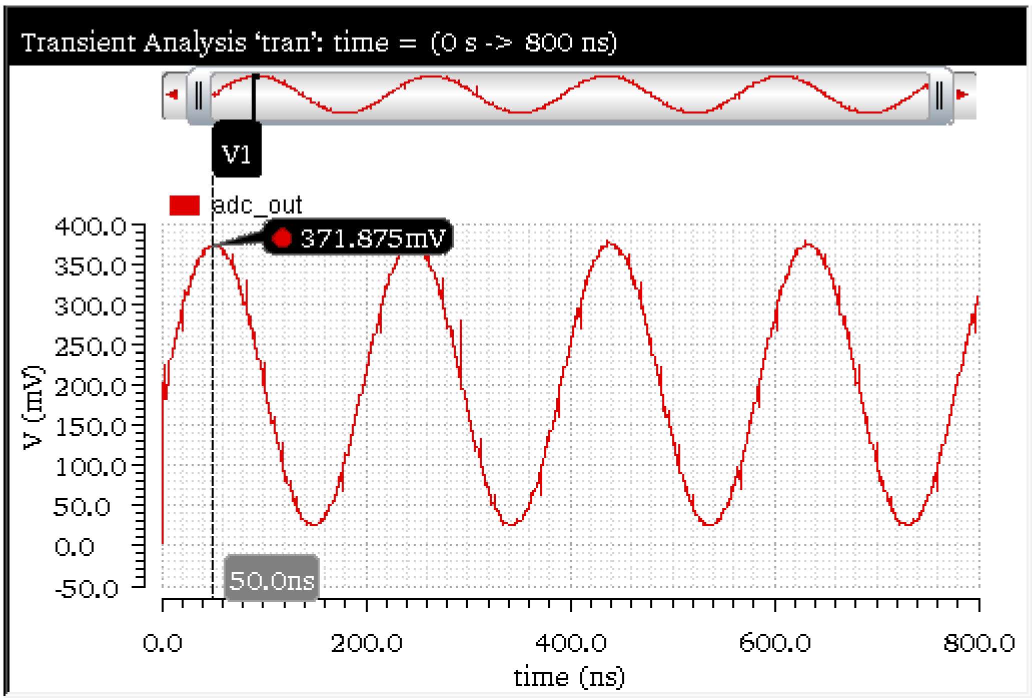
dB20

Converts a signal to dB by using the following equation:

```
db20 =  20/log(10)*log( abs(waveform_yvalue) )
```

Example

Consider the following input signal waveform from the transient analysis:



Note that a vertical marker `V1` is placed at `time=50ns` that displays the voltage value on Y-axis as `371.875mv`.

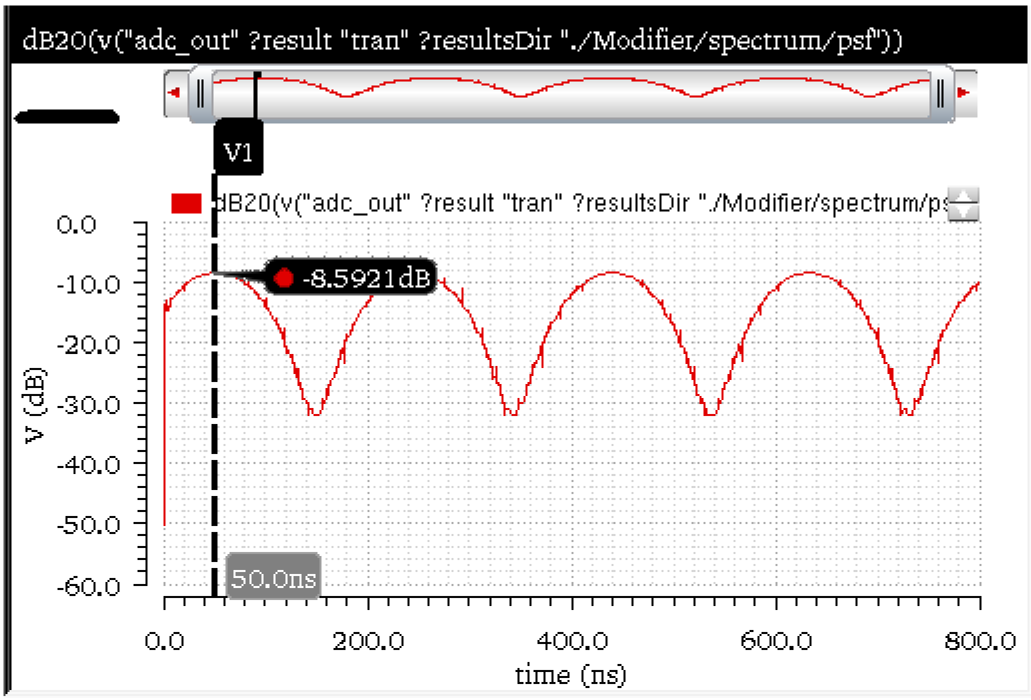
When you send this signal to Calculator, the following expression is displayed in the Buffer:

```
v("adc_out" ?result "tran" ?resultsDir "./Modifier/spectrum/psf")
```

When you select the `dB20` function from the Function Panel, the function is applied on the expression in the Buffer. The expression created in the Buffer is as follows:

```
dB20(v("adc_out" ?result "tran" ?resultsDir "./Modifier/spectrum/
psf"))
```

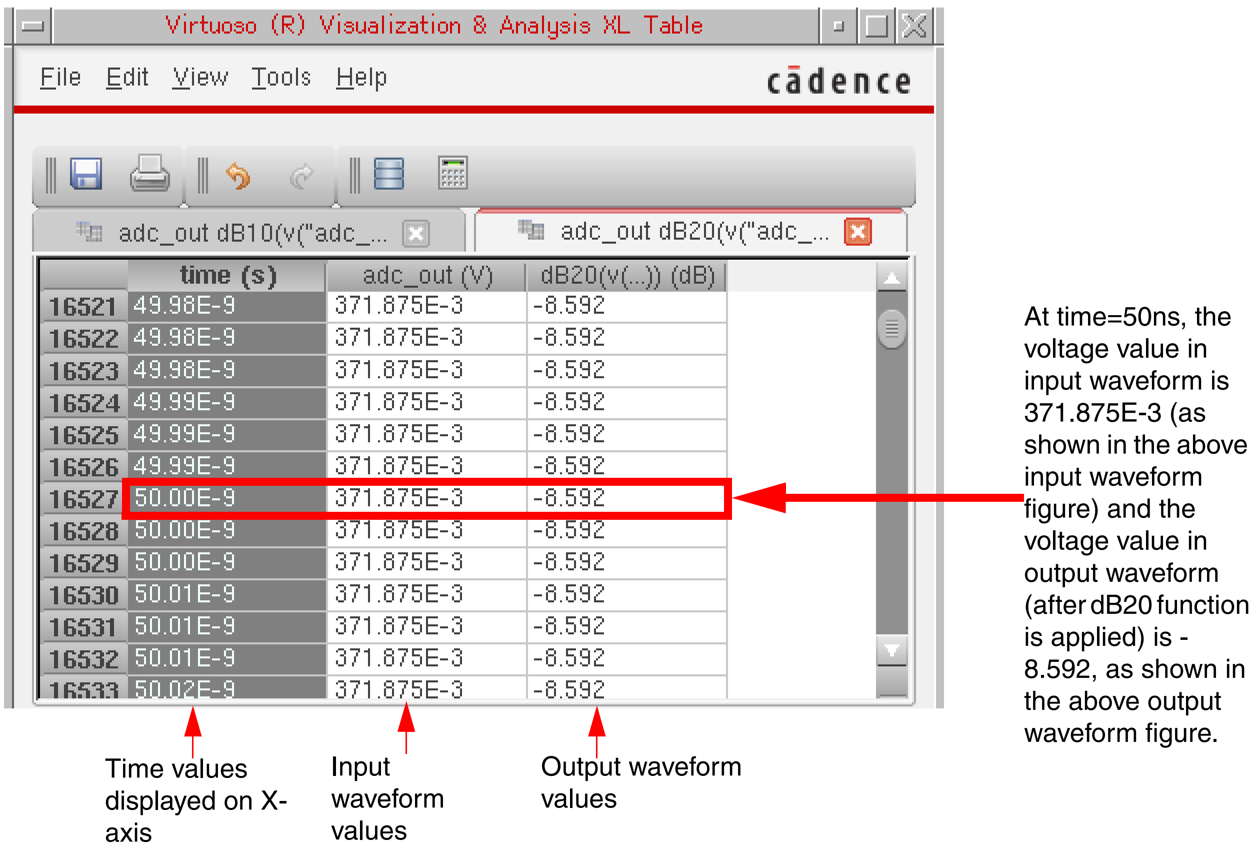
When you evaluate this expression, the following waveform is displayed in a new graph window that represents the dB20 calculation of the input waveform.



Note that a point marker `M2` is placed at `time=50ns` (where the marker `M1` is placed in the input waveform) and after dB20 calculation it displays voltage as `-8.5921dB` on the Y-axis.

To analyze these generated output values, you can display the input and output signals in the Virtuoso Visualization and Analysis XL Table. To send the input signal to the table, right-click the signal and choose *Send To – Table – New Window*. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	adc_out (V)	dB20(v(...)) (dB)
16521	49.98E-9	371.875E-3	-8.592
16522	49.98E-9	371.875E-3	-8.592
16523	49.98E-9	371.875E-3	-8.592
16524	49.99E-9	371.875E-3	-8.592
16525	49.99E-9	371.875E-3	-8.592
16526	49.99E-9	371.875E-3	-8.592
16527	50.00E-9	371.875E-3	-8.592
16528	50.00E-9	371.875E-3	-8.592
16529	50.00E-9	371.875E-3	-8.592
16530	50.01E-9	371.875E-3	-8.592
16531	50.01E-9	371.875E-3	-8.592
16532	50.01E-9	371.875E-3	-8.592
16533	50.02E-9	371.875E-3	-8.592

Related OCEAN Function

The equivalent OCEAN command for `dB20` is:

```
db20( {o_waveform | n_number} )
     => o_waveform/n_number/nil
```

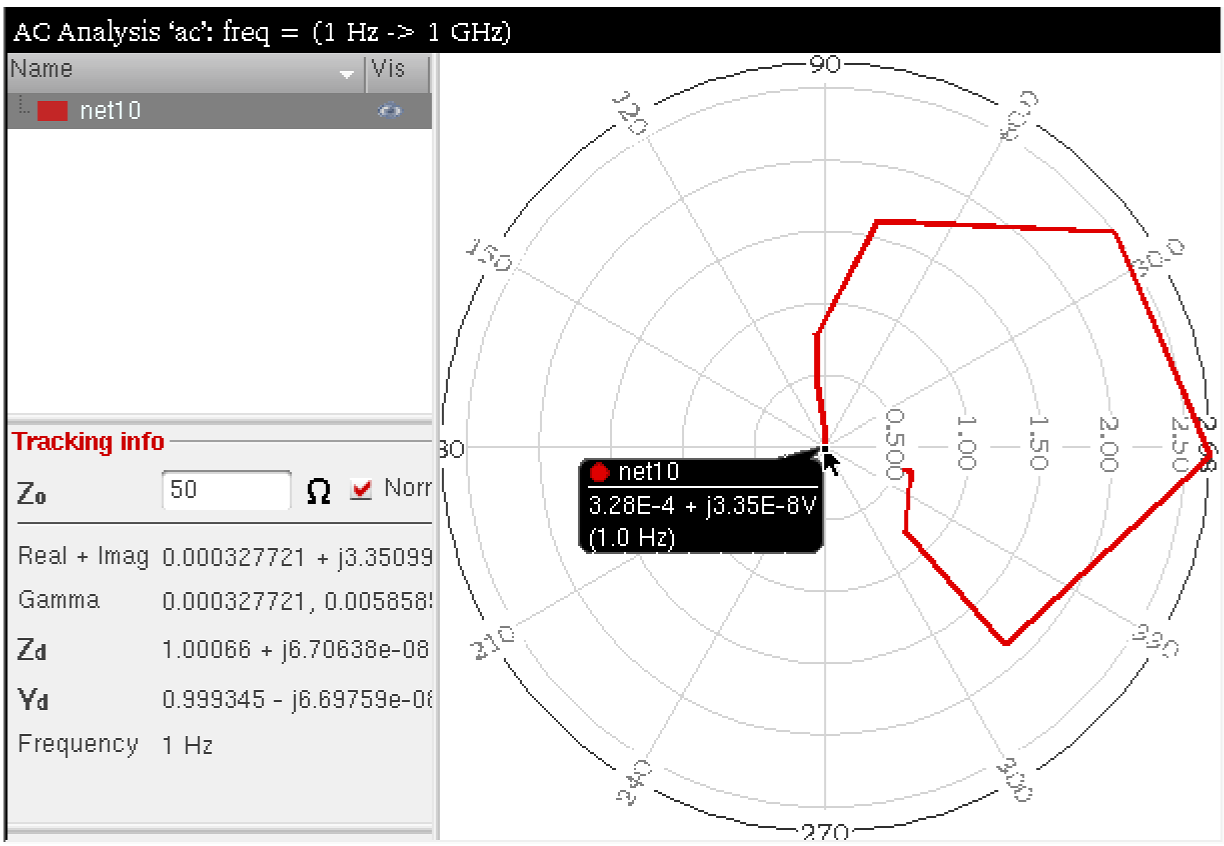
For more information, see `dB20` in *OCEAN Reference*.

imag

Returns the imaginary component of the input waveform. This function is available only in the SKILL mode.

Example

Consider the following signal expression from the AC analysis for the complex data:



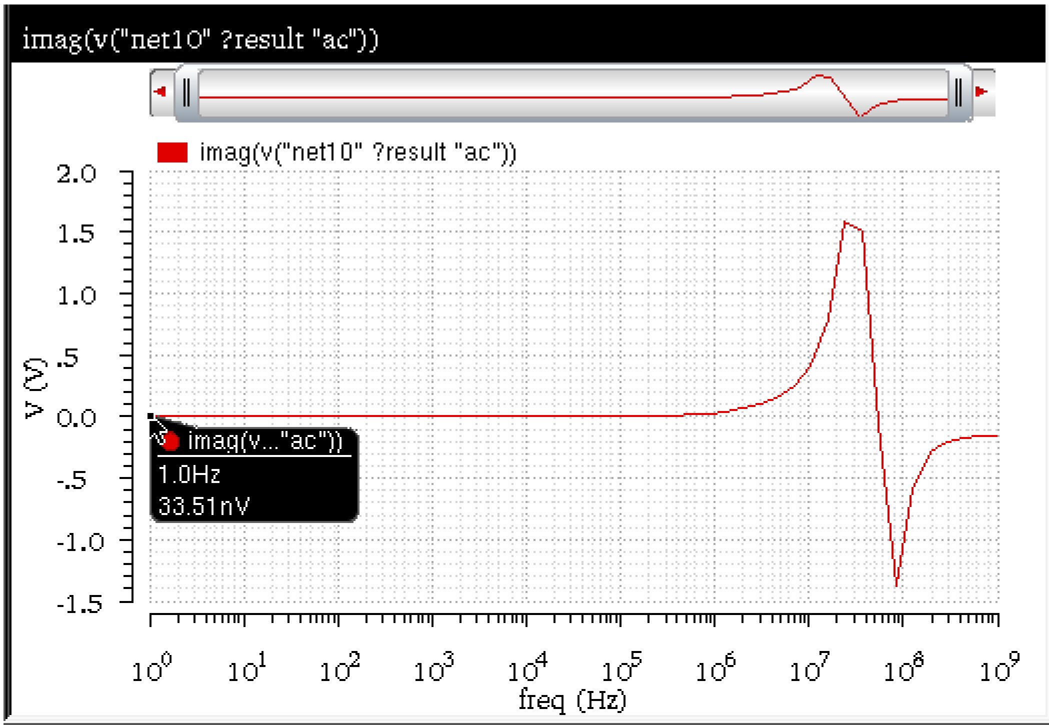
When you send this signal to Calculator, the following expression is displayed in the Buffer:

```
mag(v("net10" ?result "ac" ?resultsDir "./ampsim.raw"))
```

By default, the expression is displayed with the `mag` modifier function. You need to remove this `mag` function from the Buffer expression manually. When you select the `imag` function from the Function Panel, the function is applied on the signal expression in the Buffer. The new expression created in the Buffer is as follows:

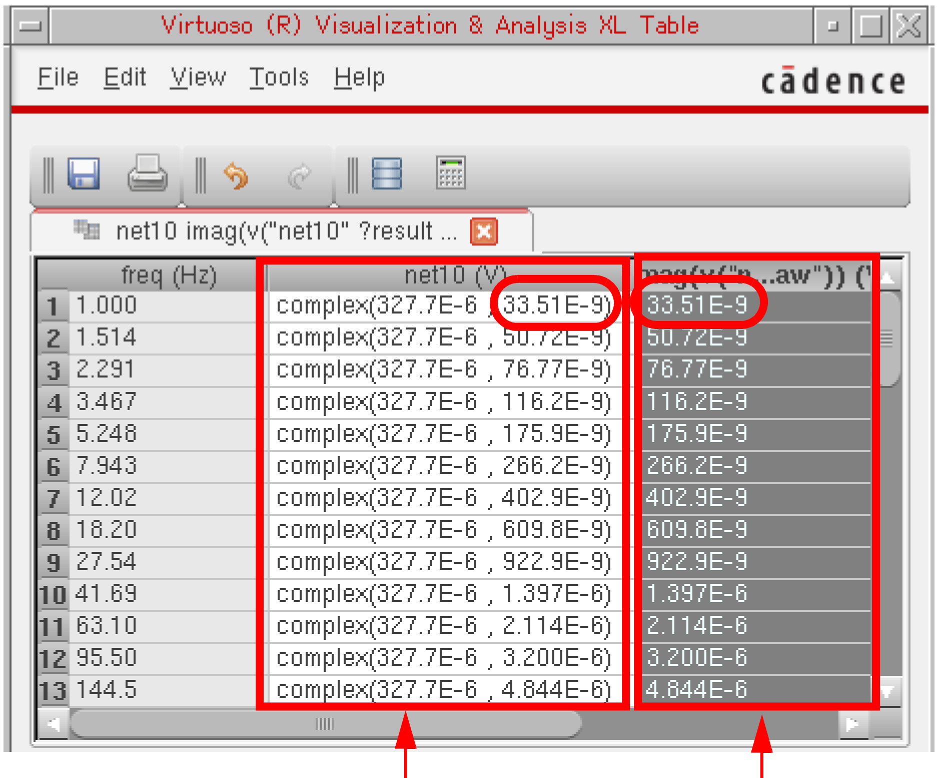
```
imag(v("net10" ?result "ac" ?resultsDir "./ampsim.raw"))
```

When you evaluate this expression, the following waveform is displayed in a new graph window that represents the conjugate of the input waveform.



To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

To view both the real and imaginary values in the table, right-click the table row and choose *Display Complex As – Real and Imaginary*. The following table contents are displayed when you send the input and output signals described in this example to the table and display the values in the complex number format.



Input waveform displaying both real and imaginary values of the signal.

Output waveform displaying only imaginary values.

Related OCEAN Function

The equivalent OCEAN command for `imag` is:

```
imag( {o_waveform | n_input} )
     => o_waveformImag/n_numberImag/nil
```

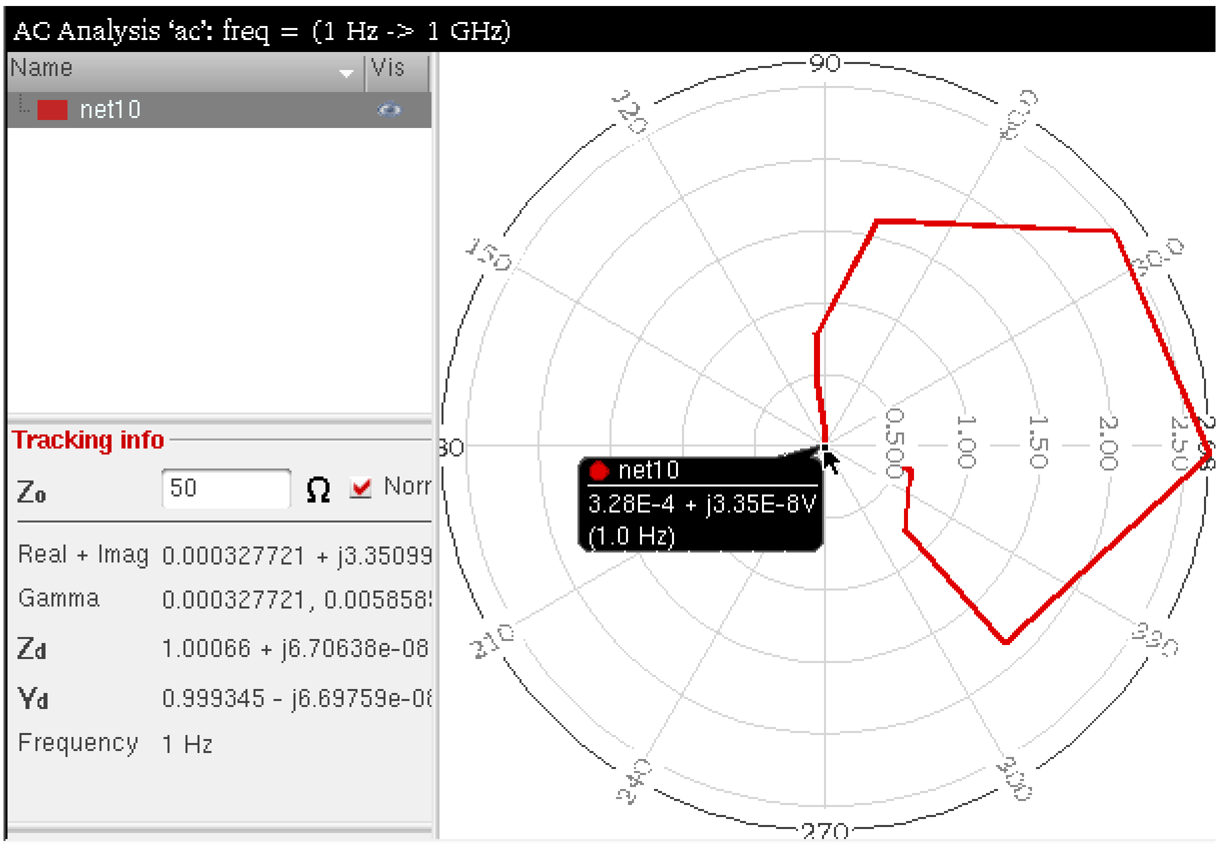
For more information, see `imag` in *OCEAN Reference*.

real

Returns the real component of the input signal.

Example

Consider the following signal expression from the AC analysis for the complex data:



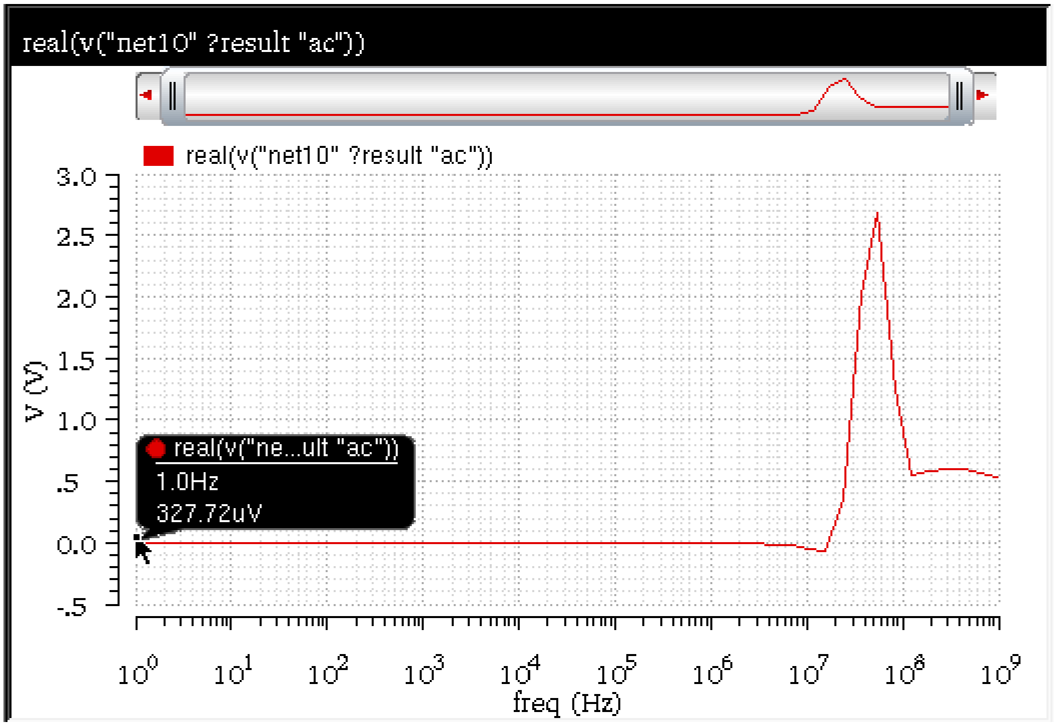
When you send this signal to Calculator, the following expression is displayed in the Buffer:

```
mag(v("net10" ?result "ac" ?resultsDir "./ampsim.raw"))
```

By default, the expression is displayed with the `mag` modifier function. You need to remove this `mag` function from the Buffer expression manually. When you select the `real` function from the Function Panel, the function is applied on the signal expression in the Buffer. The new expression created in the Buffer is as follows:

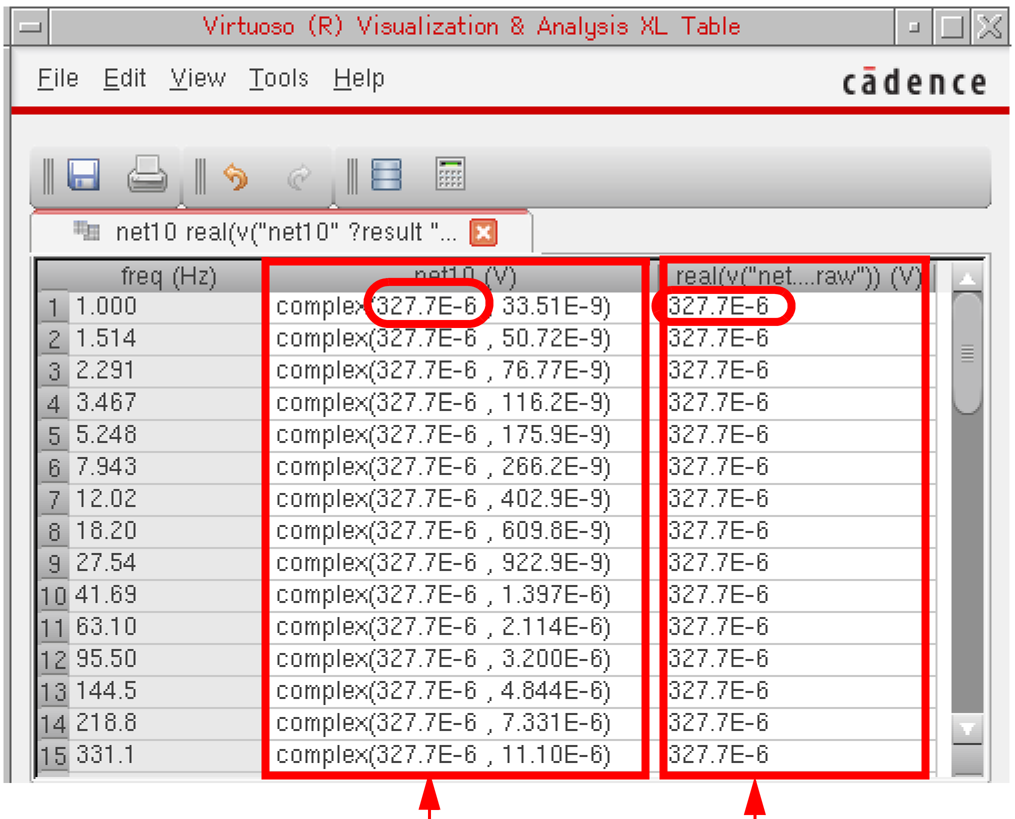
```
real(v("net10" ?result "ac" ?resultsDir "./ampsim.raw"))
```

When you evaluate this expression, the following waveform is displayed in a new graph window that represents the conjugate of the input waveform.



To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

To view both the real and imaginary values in the table, right-click the table row and choose *Display Complex As – Real and Imaginary*. The following table contents are displayed when you send the input and output signals described in this example to the table and display the values in the complex number format.



	freq (Hz)	net10 (V)	real(v("net....raw")) (V)
1	1.000	complex(327.7E-6 , 33.51E-9)	327.7E-6
2	1.514	complex(327.7E-6 , 50.72E-9)	327.7E-6
3	2.291	complex(327.7E-6 , 76.77E-9)	327.7E-6
4	3.467	complex(327.7E-6 , 116.2E-9)	327.7E-6
5	5.248	complex(327.7E-6 , 175.9E-9)	327.7E-6
6	7.943	complex(327.7E-6 , 266.2E-9)	327.7E-6
7	12.02	complex(327.7E-6 , 402.9E-9)	327.7E-6
8	18.20	complex(327.7E-6 , 609.8E-9)	327.7E-6
9	27.54	complex(327.7E-6 , 922.9E-9)	327.7E-6
10	41.69	complex(327.7E-6 , 1.397E-6)	327.7E-6
11	63.10	complex(327.7E-6 , 2.114E-6)	327.7E-6
12	95.50	complex(327.7E-6 , 3.200E-6)	327.7E-6
13	144.5	complex(327.7E-6 , 4.844E-6)	327.7E-6
14	218.8	complex(327.7E-6 , 7.331E-6)	327.7E-6
15	331.1	complex(327.7E-6 , 11.10E-6)	327.7E-6

Input waveform displaying the real and imaginary values of the signal

Output waveform displaying only the real values

Related OCEAN Function

The equivalent OCEAN command for `real` is:

```
real( {o_waveform | n_input} )
    => o_waveformReal/n_numberReal/nil
```

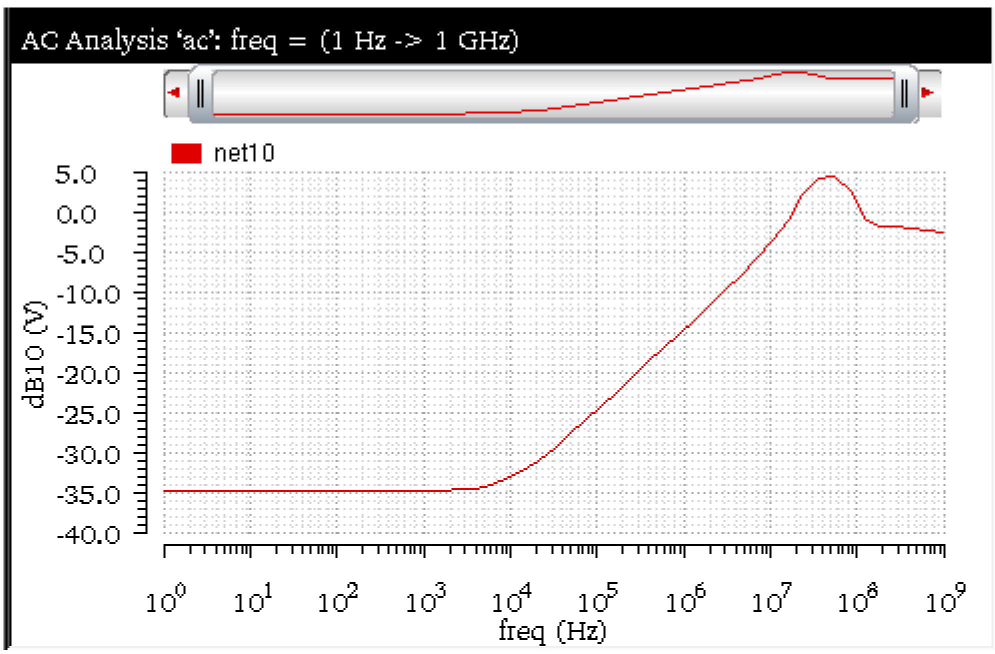
For more information, see `real` in *OCEAN Reference*.

mag

Returns the magnitude of a signal.

Example

When you plot a signal from AC analysis, by default it displays the magnitude values of the signal on Y-axis. This function is basically used when you have another modifier function, such as `real`, `imag`, `dB10`, plotted on Y-axis and you want to view the magnitude of the output signal. For example, consider the following waveform from AC analysis in which the dB10 values are plotted on the Y-axis of the waveform.



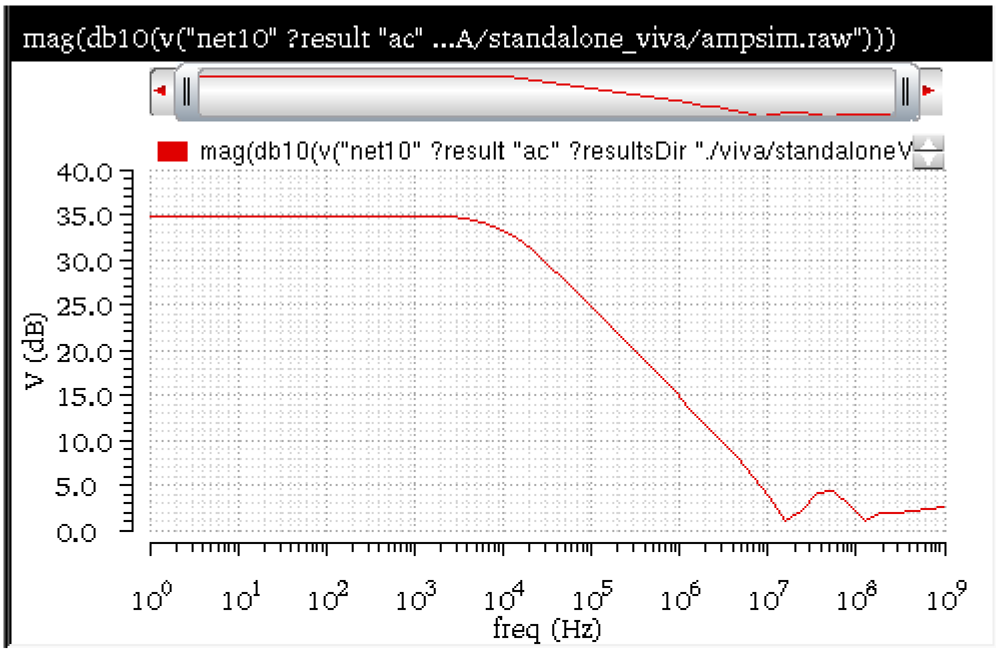
When you send this signal to Calculator, the expression displayed in the Buffer is as follows:

```
db10(v("net10" ?result "ac" ?resultsDir "./ampsim.raw"))
```

By default, the expression is displayed with the `dB10` (`db10`) function. When you select the `mag` function from the Function Panel, the function is applied on the signal expression in the Buffer. The new expression created in the Buffer is as follows:

```
mag(db10(v("net10" ?result "ac" ?resultsDir "./ampsim.raw")))
```

When you evaluate this expression, the following output waveform showing the magnitude of the db10 of the voltage signal (`net10`) is displayed in a new graph window.

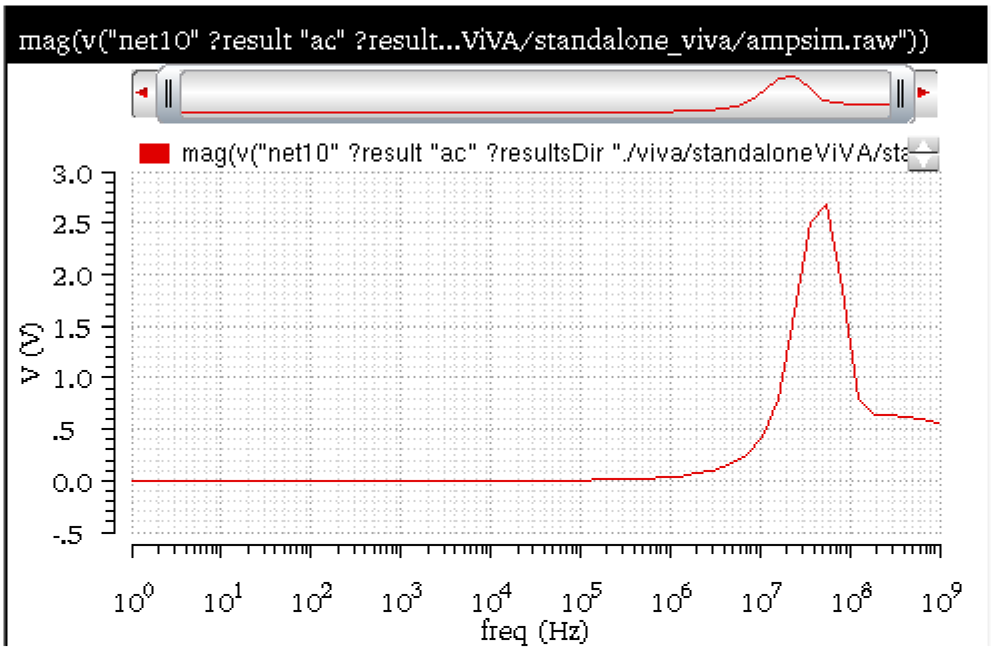


If you manually remove the `db10` function from the input signal and then apply the `mag` function, the following expression is created in the Buffer:

```
mag(v("net10" ?result "ac" ?resultsDir "./ampsim.raw"))
```

Note that this expression calculates the magnitude of the voltage signal, `net10`, where as the previous expression calculated the magnitude of the waveform after the `dB10` modifier function is applied.

When you evaluate this expression, the following output waveform appears in a new graph window.



Related OCEAN Function

The equivalent OCEAN command for `mag` is:

```
mag( {o_waveform | n_number} )
     => o_waveform/n_number/nil
```

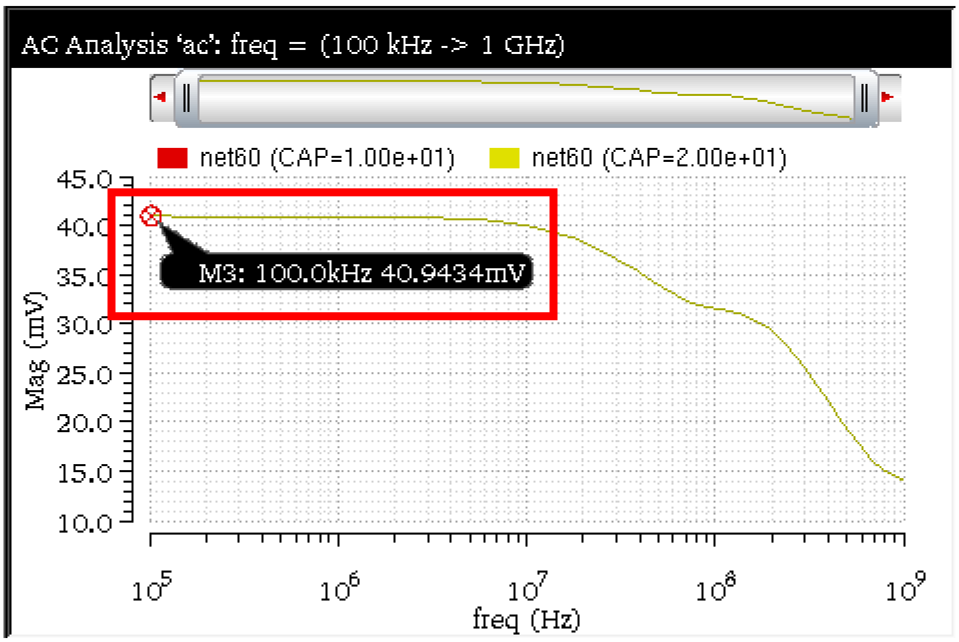
For more information, see `mag` in *OCEAN Reference*.

phase

Returns the phase of a signal in degrees.

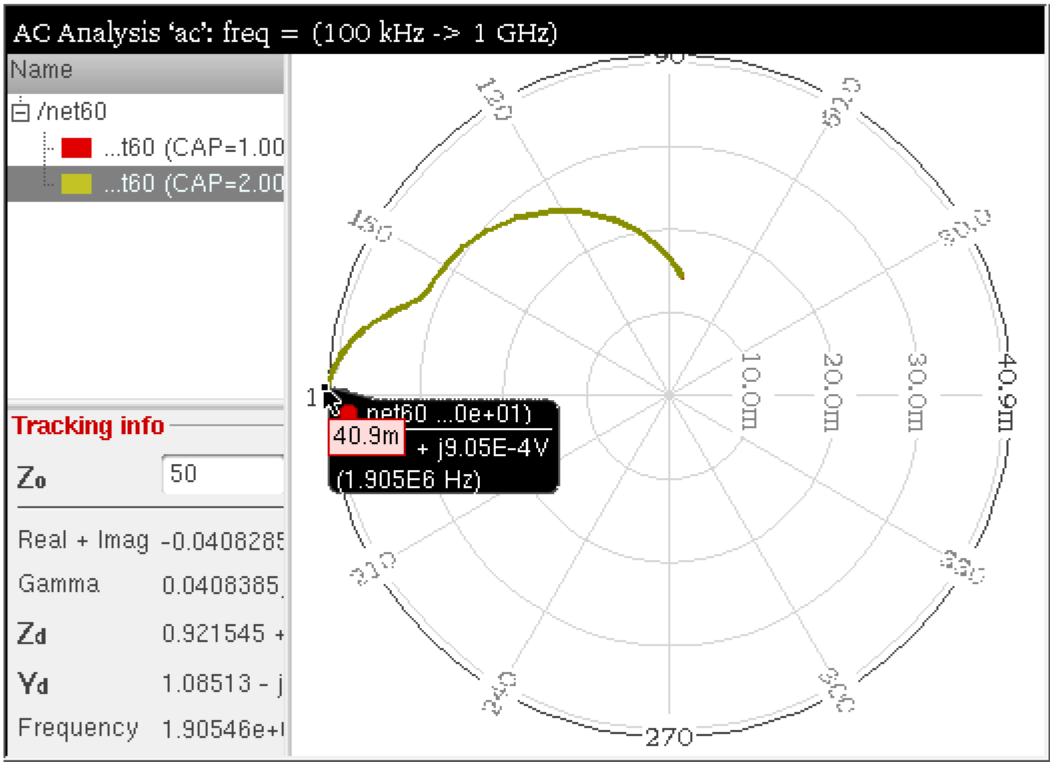
Example

Consider the following input waveform from a AC analysis:



Note that a point marker, `M3`, is placed at `freq=100KHz` and the `voltage=40.9434mV`.

The following graph appears when you plot this input signal in a circular graph (plor plot):



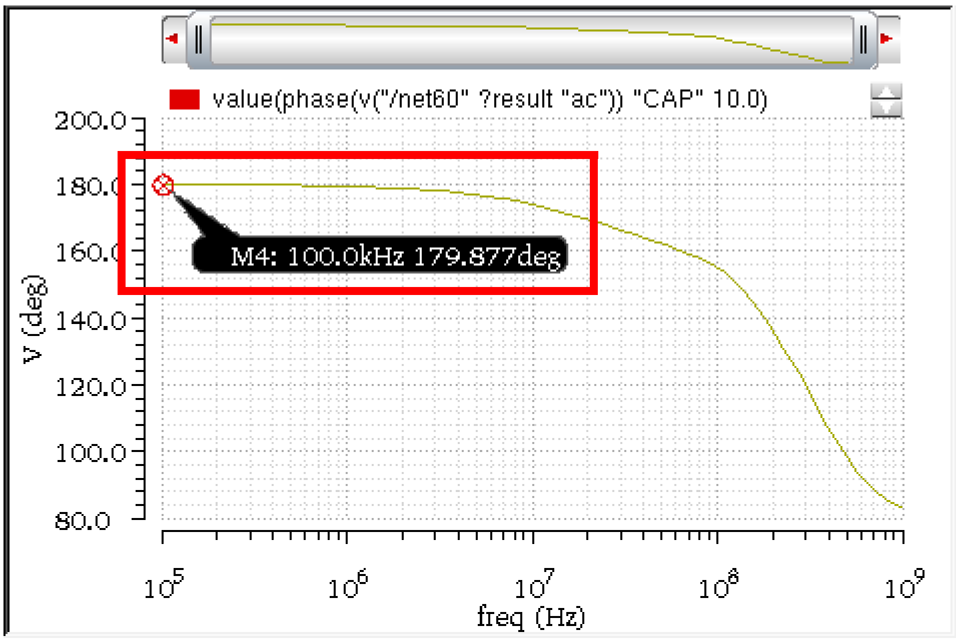
When you send this signal to Calculator, the following expression is displayed in the Buffer:

```
mag(v("net60" ?result "ac" ?resultsDir "./Modifier/peakTest/spectre/
schematic/psf"))
```

By default, the expression is displayed with the `mag` modifier function. You need to remove this `mag` function from the Buffer expression manually. When you select the `phase` function from the Function Panel, the function is applied on the signal expression in the Buffer. The new expression created in the Buffer is as follows:

```
phase(v("net60" ?result "ac" ?resultsDir "./Modifier/peakTest/
spectre/schematic/psf"))
```

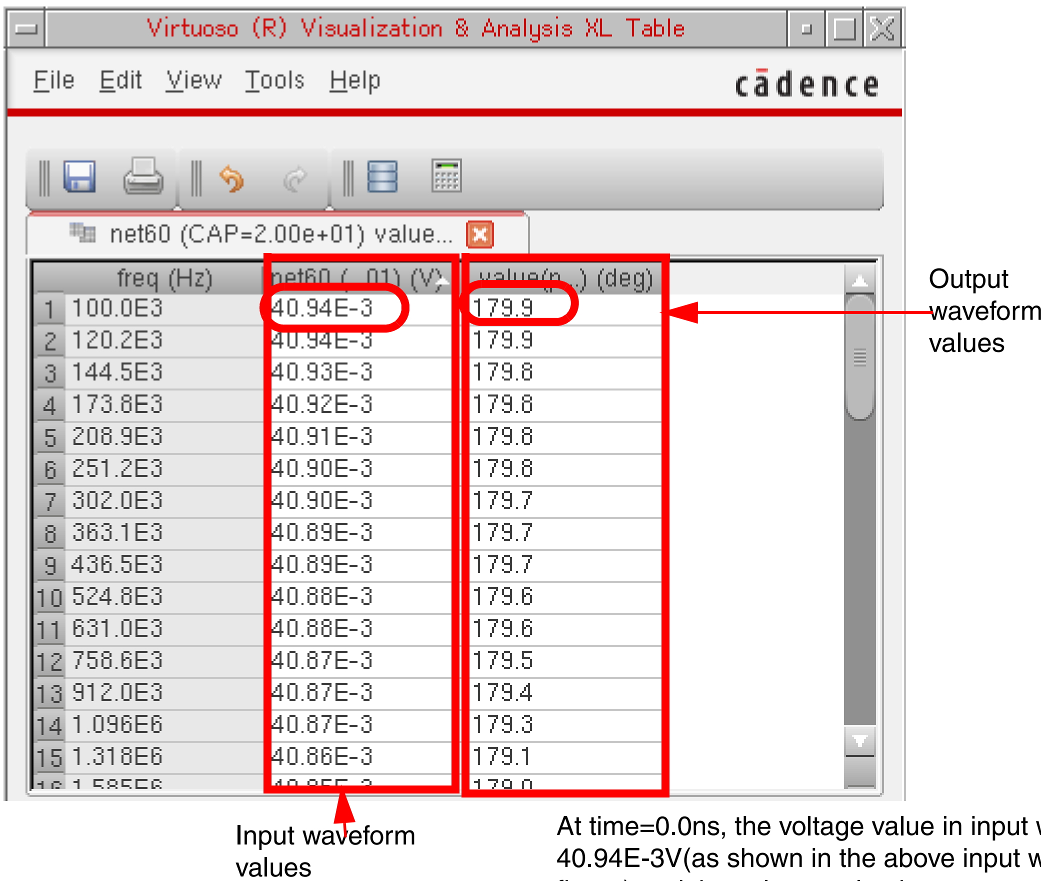
When you evaluate this expression, the following waveform is displayed in a new graph window that represents the phase of the input waveform.



Note that a point marker `M4` is placed at `freq=100.0KHz` (where the point marker `M3` was placed in the input waveform) and after phase calculation, it displays the voltage as `179.877` degrees.

To analyze these generated output voltage values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table.



	freq (Hz)	net60 (...01) (V)	value(p...) (deg)
1	100.0E3	40.94E-3	179.9
2	120.2E3	40.94E-3	179.9
3	144.5E3	40.93E-3	179.8
4	173.8E3	40.92E-3	179.8
5	208.9E3	40.91E-3	179.8
6	251.2E3	40.90E-3	179.8
7	302.0E3	40.90E-3	179.7
8	363.1E3	40.89E-3	179.7
9	436.5E3	40.89E-3	179.7
10	524.8E3	40.88E-3	179.6
11	631.0E3	40.88E-3	179.6
12	758.6E3	40.87E-3	179.5
13	912.0E3	40.87E-3	179.4
14	1.096E6	40.87E-3	179.3
15	1.318E6	40.86E-3	179.1

At time=0.0ns, the voltage value in input waveform is 40.94E-3V(as shown in the above input waveform figure) and the voltage value in output waveform (after dB20 function is applied) is 179.9 deg, as shown in the above output waveform figure.

Related OCEAN Function

The equivalent OCEAN command for `phase` is:

```
phase( {o_waveform | n_number} )
     => o_waveform/n_number/nil
```

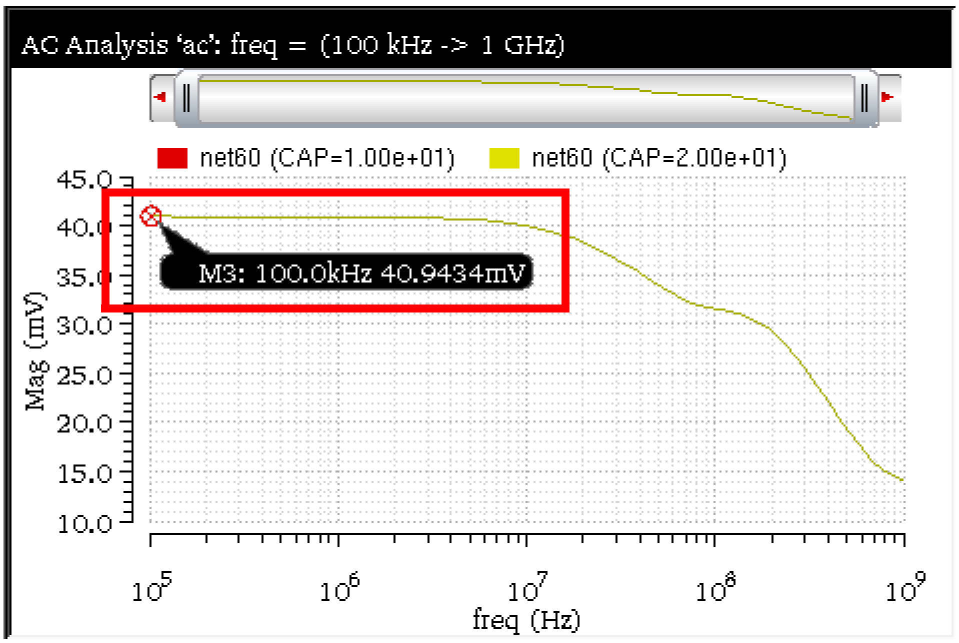
For more information, see phase in *OCEAN Reference*.

phaseRad

Calculates the wrapped (discontinuous) phase in radians of a waveform.

Example

Consider the following input waveform from a AC analysis:



Note that a point marker, `M3`, is placed at `freq=100KHz` and `voltage=40.9434mV.`

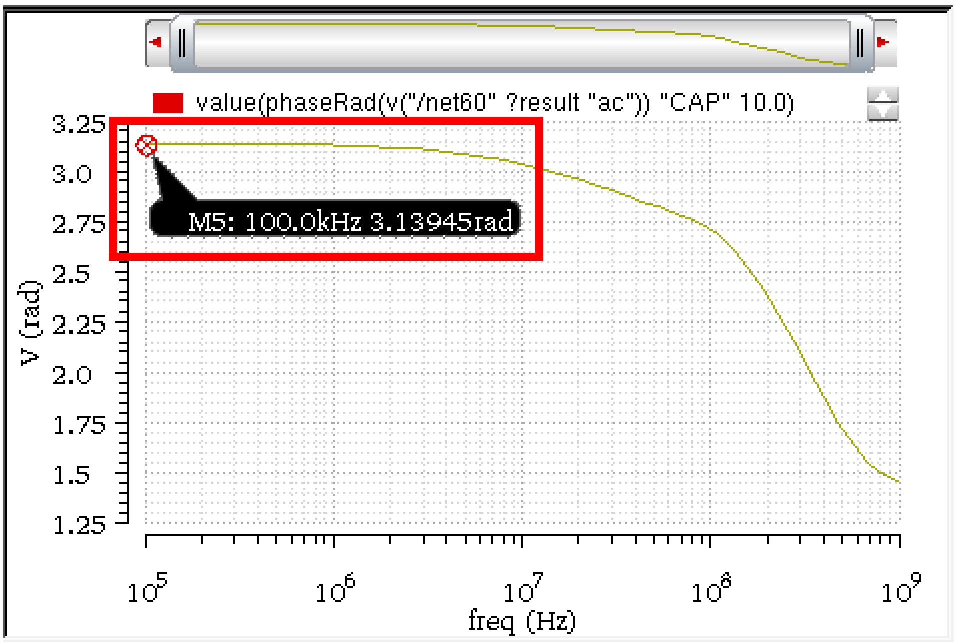
When you send this signal to Calculator, the following expression is displayed in the Buffer:

```
mag(v("net60" ?result "ac" ?resultsDir "./Modifier/peakTest/spectre/
schematic/psf"))
```

By default, the expression is displayed with the `mag` modifier function. You need to remove this `mag` function from the Buffer expression manually. When you select the `phaseRad` function from the Function Panel, the function is applied on the signal expression in the Buffer. The new expression created in the Buffer is as follows:

```
phaseRad(v("net60" ?result "ac" ?resultsDir "./Modifier/peakTest/
spectre/schematic/psf"))
```

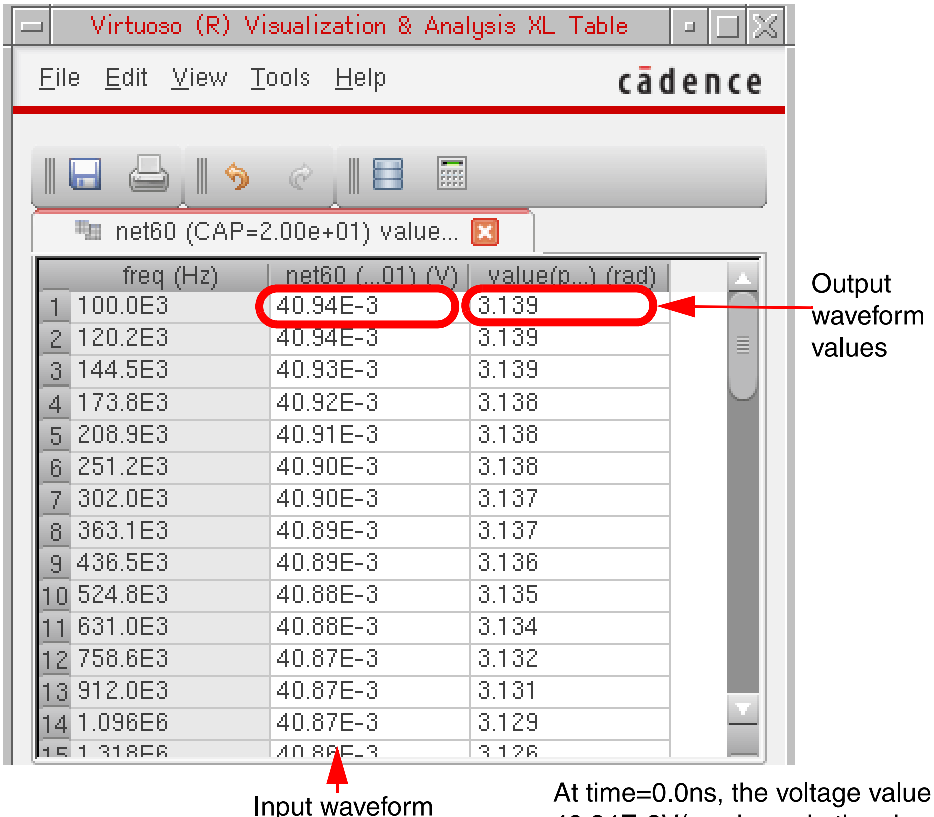
When you evaluate this expression, the following waveform is displayed in a new graph window that represents the phase (in radians) of the input waveform.



Note that a point marker `M5` is placed at `freq=100.0KHz` (where the point marker `M3` was placed in the input waveform) and after phase calculation, it displays the voltage as `3.13945` degrees.

To analyze these generated output voltage values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table.



	freq (Hz)	net60 (...01) (V)	value(p...) (rad)
1	100.0E3	40.94E-3	3.139
2	120.2E3	40.94E-3	3.139
3	144.5E3	40.93E-3	3.139
4	173.8E3	40.92E-3	3.138
5	208.9E3	40.91E-3	3.138
6	251.2E3	40.90E-3	3.138
7	302.0E3	40.90E-3	3.137
8	363.1E3	40.89E-3	3.137
9	436.5E3	40.89E-3	3.136
10	524.8E3	40.88E-3	3.135
11	631.0E3	40.88E-3	3.134
12	758.6E3	40.87E-3	3.132
13	912.0E3	40.87E-3	3.131
14	1.096E6	40.87E-3	3.129
15	1.318E6	[illegible]	3.126

At time=0.0ns, the voltage value in input waveform is 40.94E-3V(as shown in the above input waveform figure) and the voltage value in output waveform (after dB20 function is applied) is 3.139 rad, as shown in the above output waveform figure.

Related OCEAN Function

The equivalent OCEAN command for `phase` is:

```
phaseRad( {o_waveform | n_number} )
     => o_waveform/n_number/nil
```

For more information, see `phase` in *OCEAN Reference*.

Trigonometric Functions

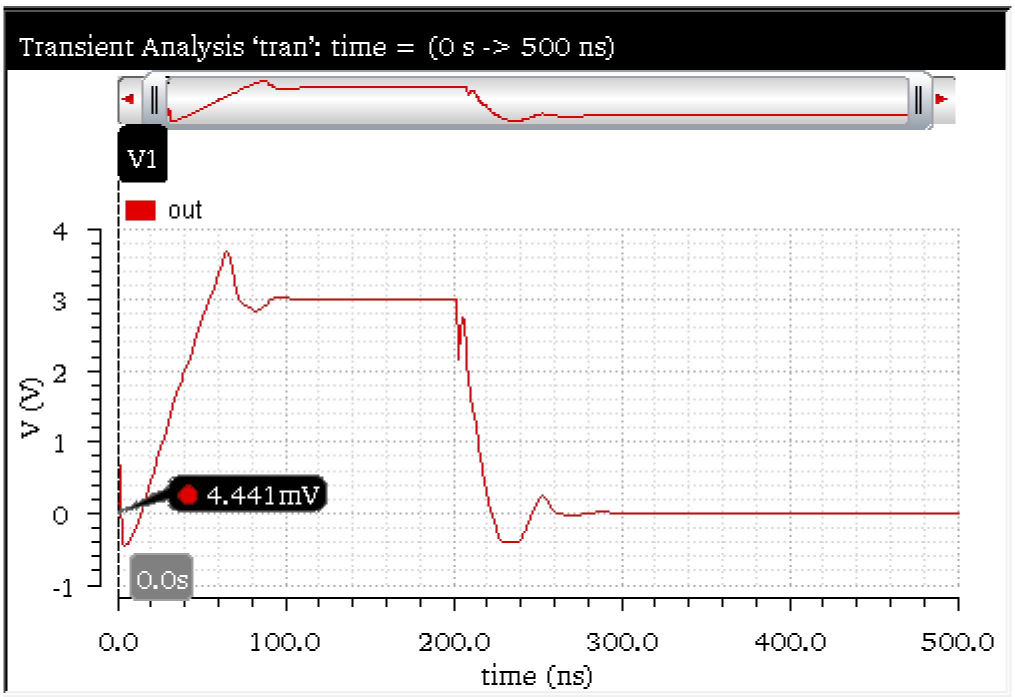
This section covers the following Trigonometeric functions in Calculator:

cos

Returns the cosine of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0ns` and voltage=`4.441mV`.

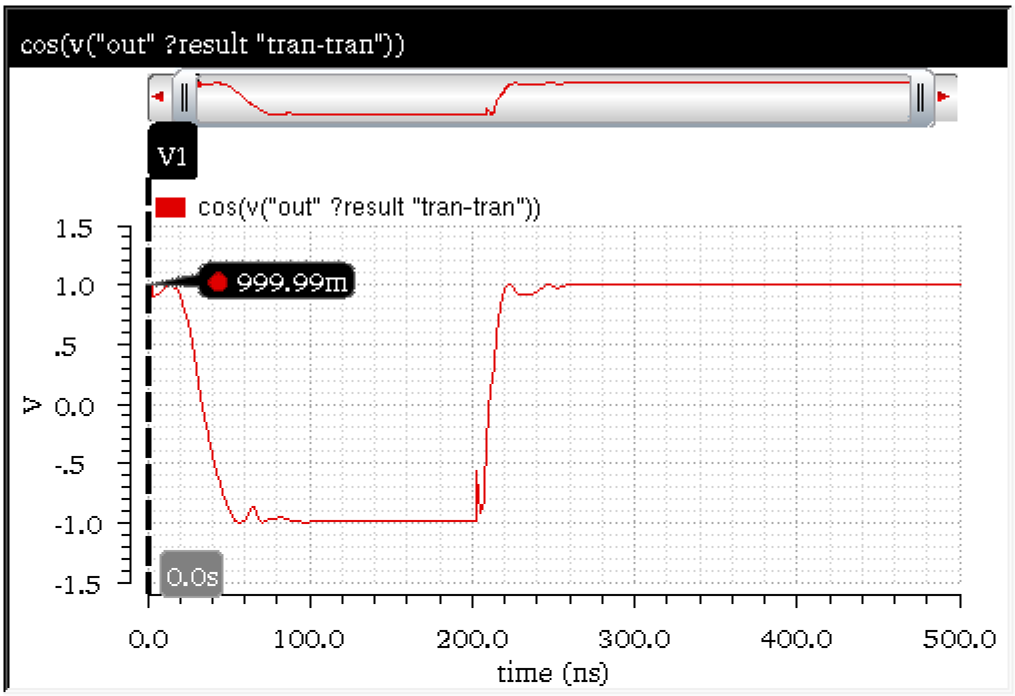
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `cos` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
cos(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

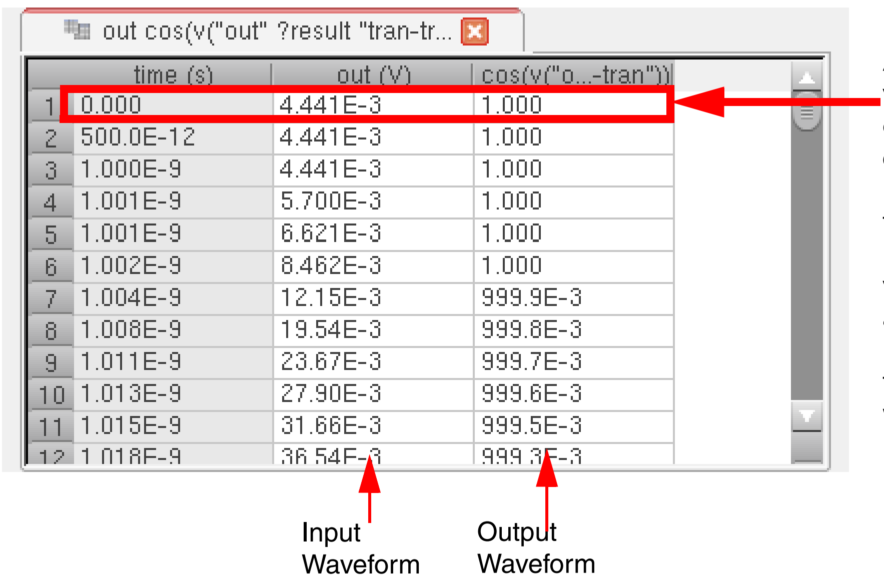
When you evaluate this expression, the function returns the integer voltage values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`999.99`mV when placed at time=`0.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out cos(v("out" ?result "tran-tr...

	time (s)	out (V)	cos(v("o...-tran"))
1	0.000	4.441E-3	1.000
2	500.0E-12	4.441E-3	1.000
3	1.000E-9	4.441E-3	1.000
4	1.001E-9	5.700E-3	1.000
5	1.001E-9	6.621E-3	1.000
6	1.002E-9	8.462E-3	1.000
7	1.004E-9	12.15E-3	999.9E-3
8	1.008E-9	19.54E-3	999.8E-3
9	1.011E-9	23.67E-3	999.7E-3
10	1.013E-9	27.90E-3	999.6E-3
11	1.015E-9	31.66E-3	999.5E-3
12	1.018E-9	36.54E-3	999.3E-3

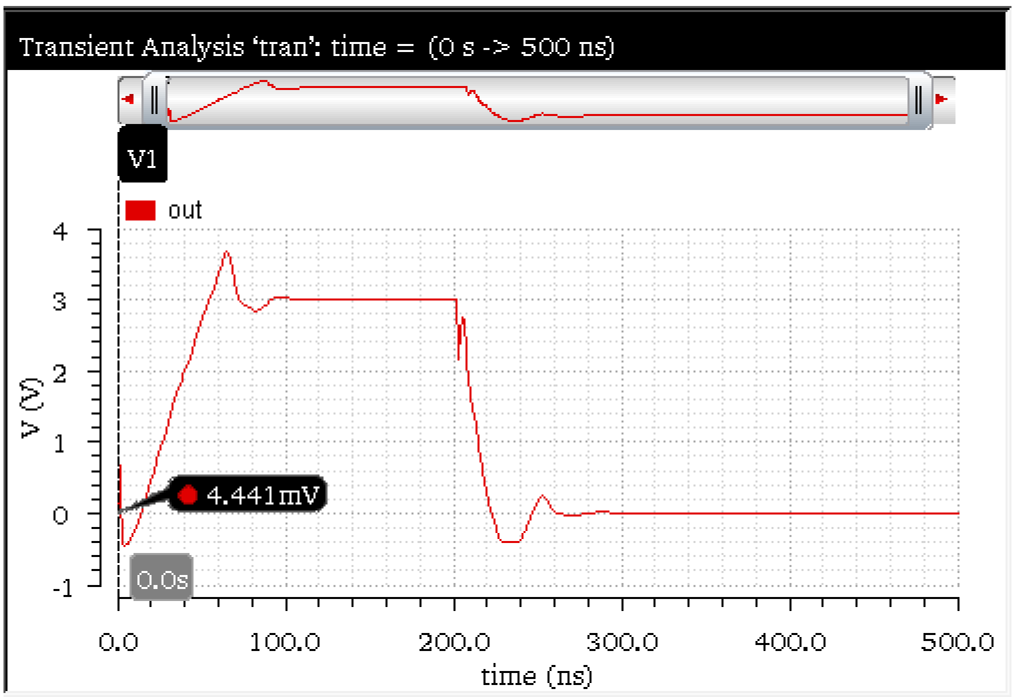
At time=0.0ns, the value of voltage in the input signal1, out(V) is 4.441E-3. After the cos function is applied, the resultant voltage value at time=0.0ns in output signal is 1.000V. This is the cosine value of 4.441E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

acos

Returns the inverse cosine of a ignal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0ns` and voltage=`4.441mV`.

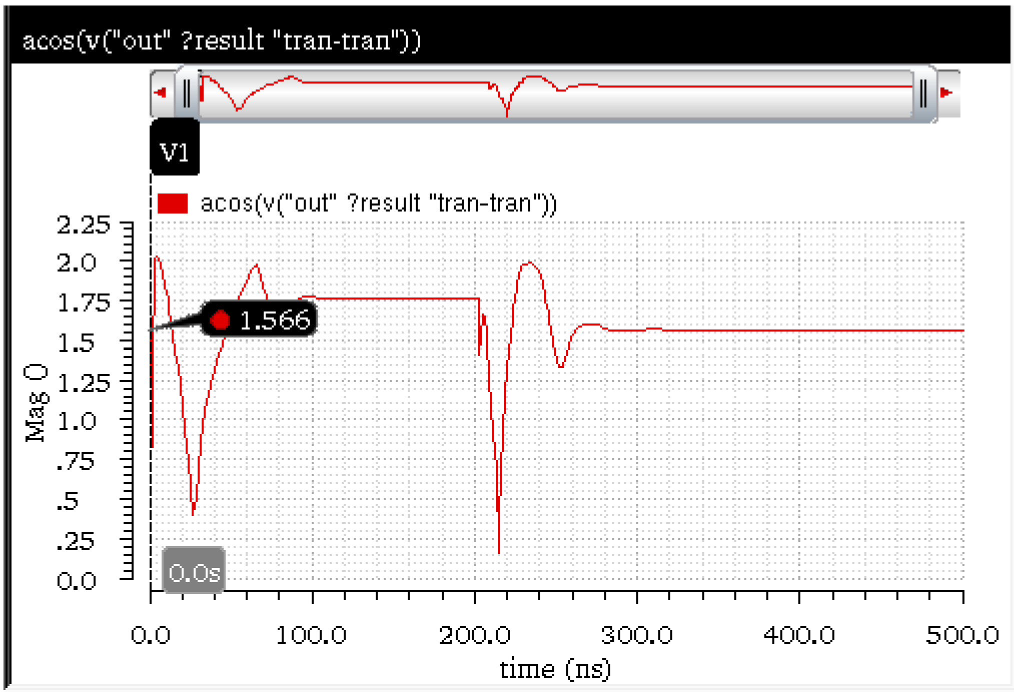
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `acos` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
acos(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

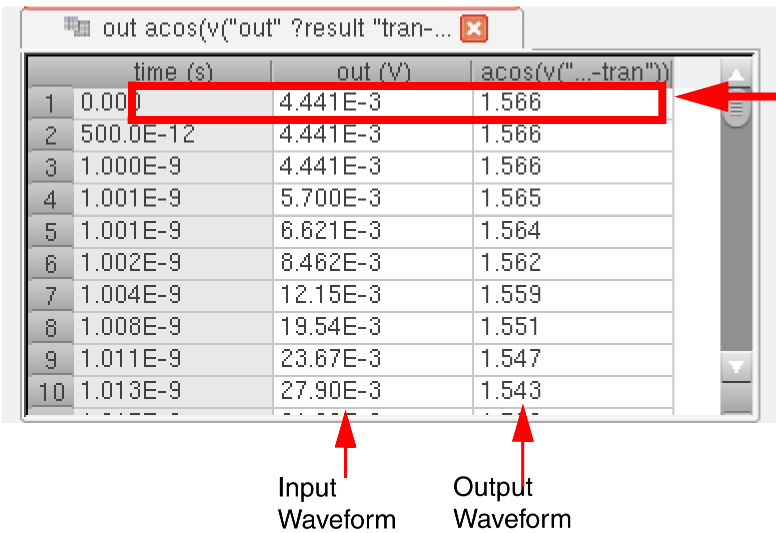
When you evaluate this expression, the function returns the inverse cosine values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=1.566 when placed at time=`0.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out acos(v("out" ?result "tran-...

	time (s)	out (V)	acos(v("...-tran"))
1	0.000	4.441E-3	1.566
2	500.0E-12	4.441E-3	1.566
3	1.000E-9	4.441E-3	1.566
4	1.001E-9	5.700E-3	1.565
5	1.001E-9	6.621E-3	1.564
6	1.002E-9	8.462E-3	1.562
7	1.004E-9	12.15E-3	1.559
8	1.008E-9	19.54E-3	1.551
9	1.011E-9	23.67E-3	1.547
10	1.013E-9	27.90E-3	1.543

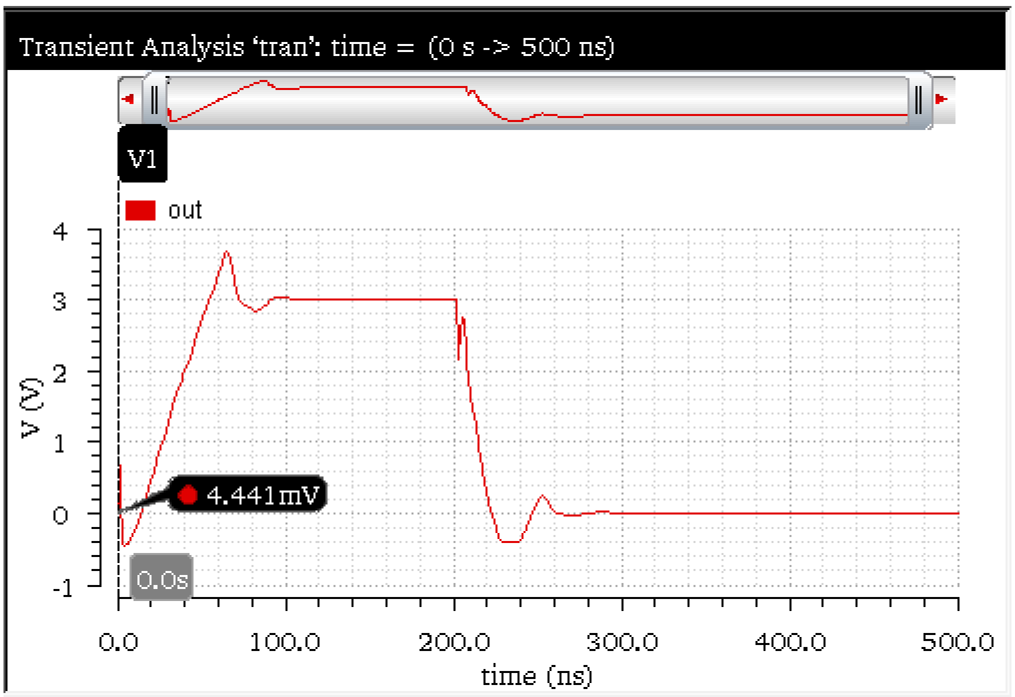
At time=0.0ns, the value of voltage in the input signal, out(V), is 4.441E-3. After the acos function is applied, the resultant voltage value at time=0.0ns in output signal is 1.566V. This is the inverse cosine value of 4.441E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

cosh

Returns the hyperbolic cosine of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0ns` and voltage=`4.441mV`.

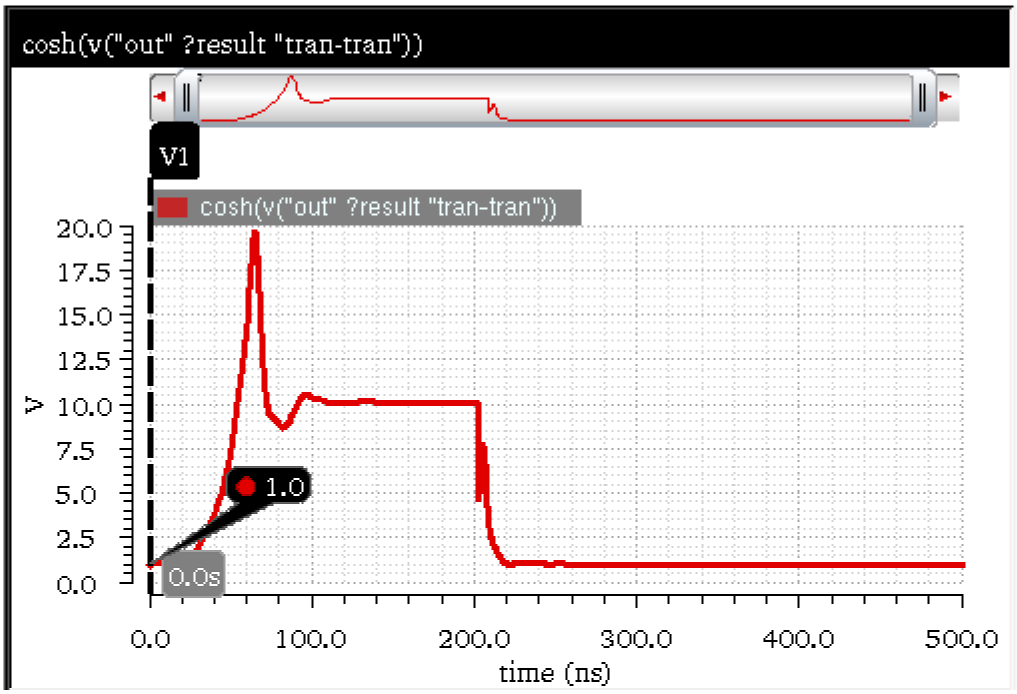
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `cosh` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
cosh(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

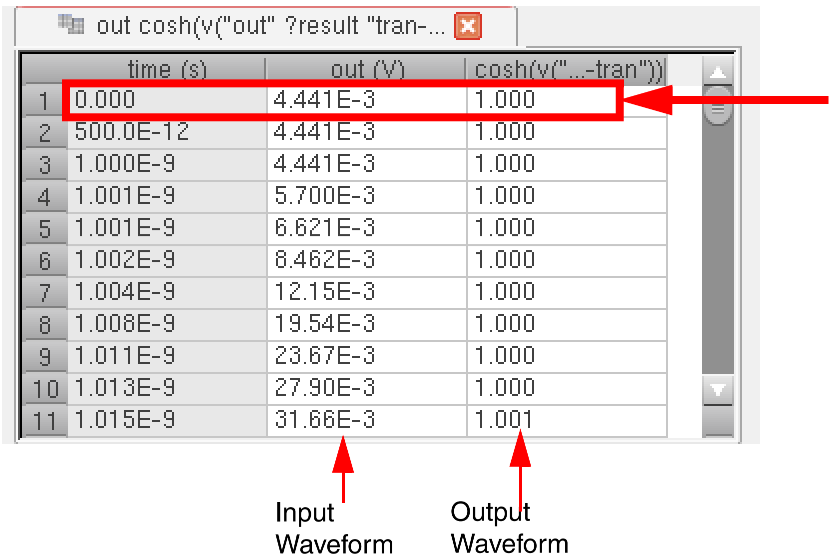
When you evaluate this expression, the function returns the hyperbolic cosine values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`1.0V` when placed at time=`0.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	cosh(v("...-tran"))
1	0.000	4.441E-3	1.000
2	500.0E-12	4.441E-3	1.000
3	1.000E-9	4.441E-3	1.000
4	1.001E-9	5.700E-3	1.000
5	1.001E-9	6.621E-3	1.000
6	1.002E-9	8.462E-3	1.000
7	1.004E-9	12.15E-3	1.000
8	1.008E-9	19.54E-3	1.000
9	1.011E-9	23.67E-3	1.000
10	1.013E-9	27.90E-3	1.000
11	1.015E-9	31.66E-3	1.001

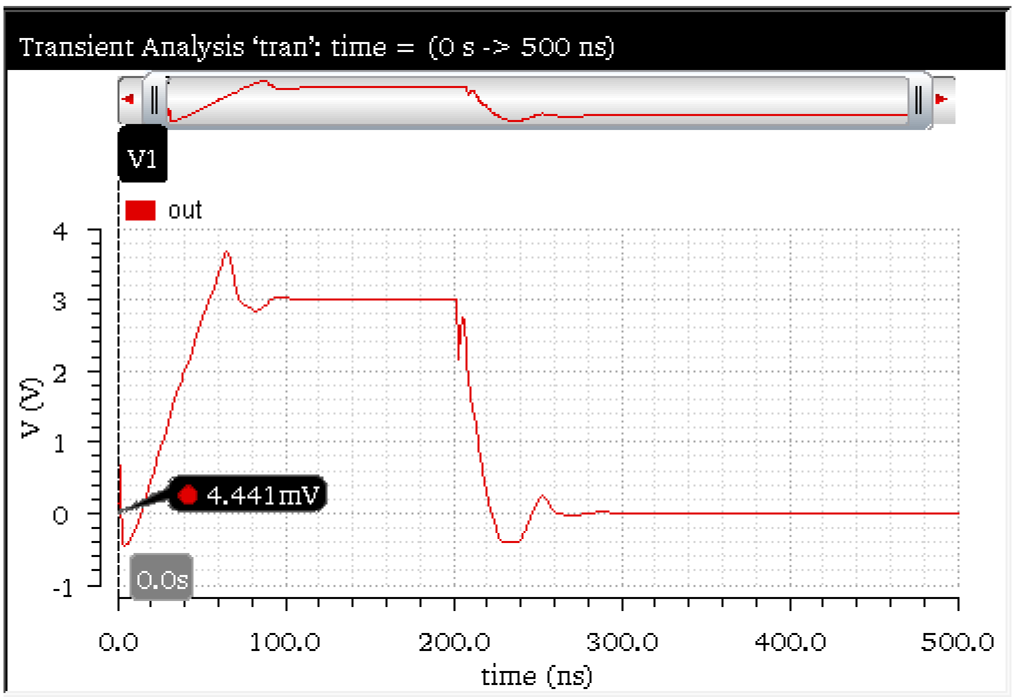
At time=0.0ns, the value of voltage in the input signal, out(V), is 4.441E-3. After the cosh function is applied, the resultant voltage value at time=0.0ns in output signal is 1.0V. This is the hyperbolic cosine value of 4.441E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

acosh

Returns the inverse hyperbolic cosine or hyperbolic arc-cosine of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0s` and voltage=`4.441mV`.

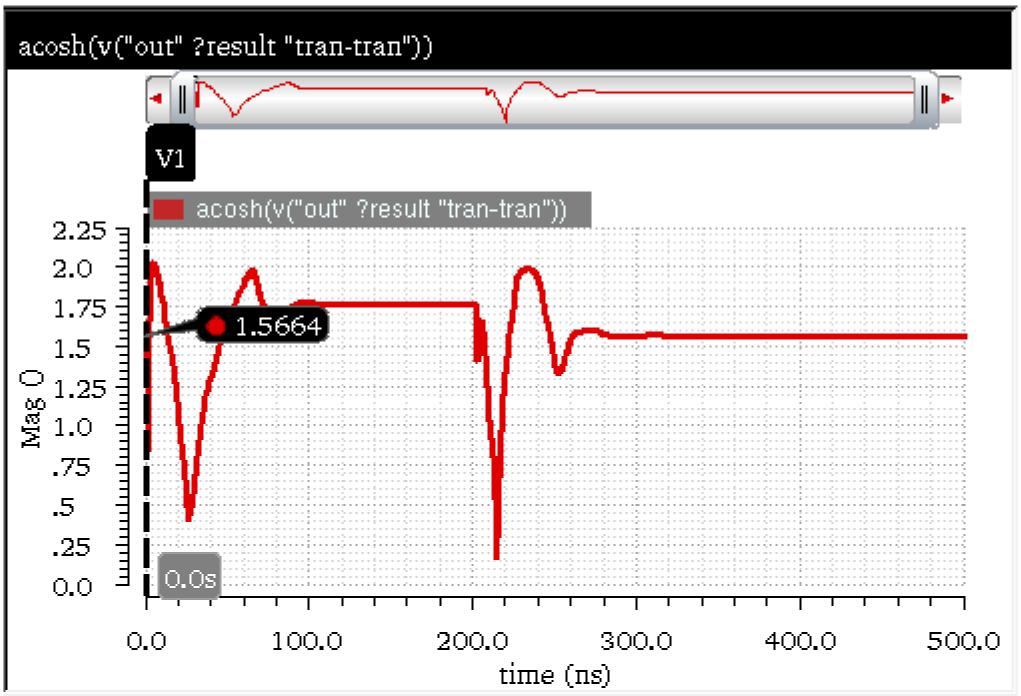
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `acosh` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
acosh(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

When you evaluate this expression, the function returns the hyperbolic arc-cosine values of the input waveform. The following output waveform is displayed in a new window:

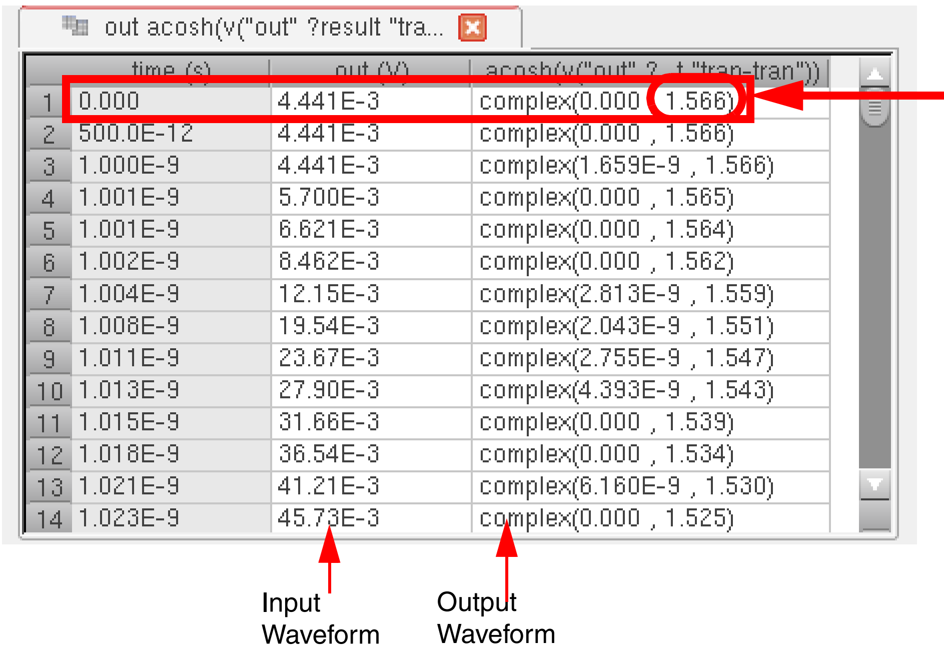


In the output waveform, note that the vertical maker `V1` shows the voltage=`1.5664V` when placed at time=`0.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

To view both the real and imaginary values in the table, right-click the table row and choose *Display Complex As – Real and Imaginary*. The following table contents are displayed when you send the input and output signals described in this example to the table and display the values in the complex number format.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	acosh(v("out" ?...t "tran-tran"))
1	0.000	4.441E-3	complex(0.000 , 1.566)
2	500.0E-12	4.441E-3	complex(0.000 , 1.566)
3	1.000E-9	4.441E-3	complex(1.659E-9 , 1.566)
4	1.001E-9	5.700E-3	complex(0.000 , 1.565)
5	1.001E-9	6.621E-3	complex(0.000 , 1.564)
6	1.002E-9	8.462E-3	complex(0.000 , 1.562)
7	1.004E-9	12.15E-3	complex(2.813E-9 , 1.559)
8	1.008E-9	19.54E-3	complex(2.043E-9 , 1.551)
9	1.011E-9	23.67E-3	complex(2.755E-9 , 1.547)
10	1.013E-9	27.90E-3	complex(4.393E-9 , 1.543)
11	1.015E-9	31.66E-3	complex(0.000 , 1.539)
12	1.018E-9	36.54E-3	complex(0.000 , 1.534)
13	1.021E-9	41.21E-3	complex(6.160E-9 , 1.530)
14	1.023E-9	45.73E-3	complex(0.000 , 1.525)

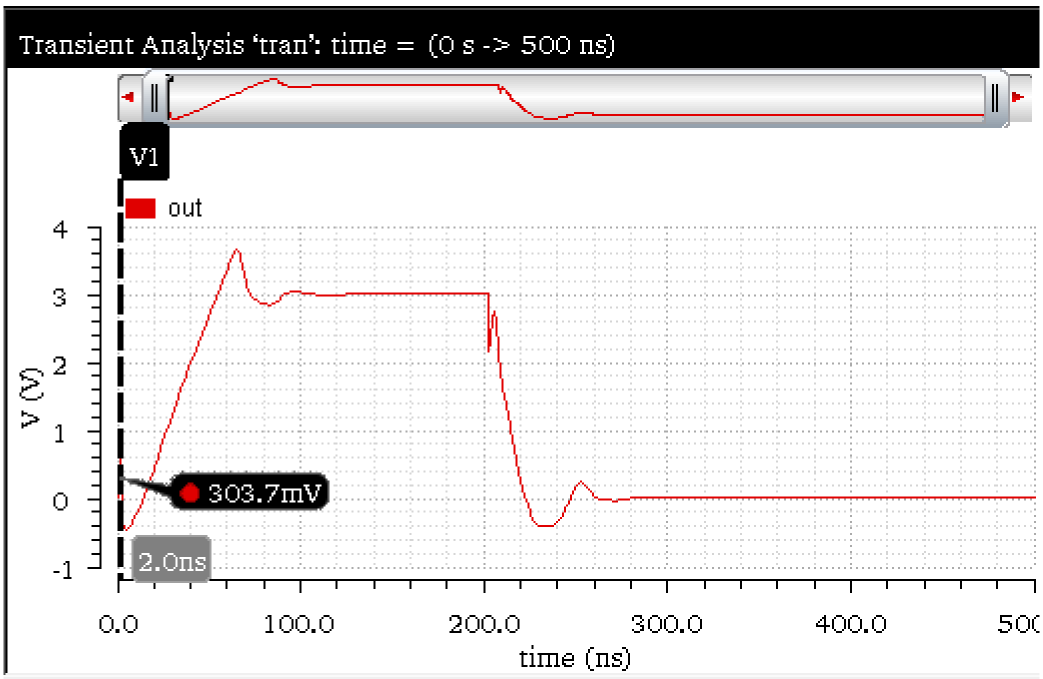
At time=0.0ns, the value of voltage in the input signal, out(V), is 4.441E-3. After the acosh function is applied, the resultant voltage value at time=0.0ns in output signal is a complex number with real part=1.566V. This is the hyperbolic arc cosine value of 4.441E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

sin

Returns the sine of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`2.0ns` and voltage=`303.7mV`.

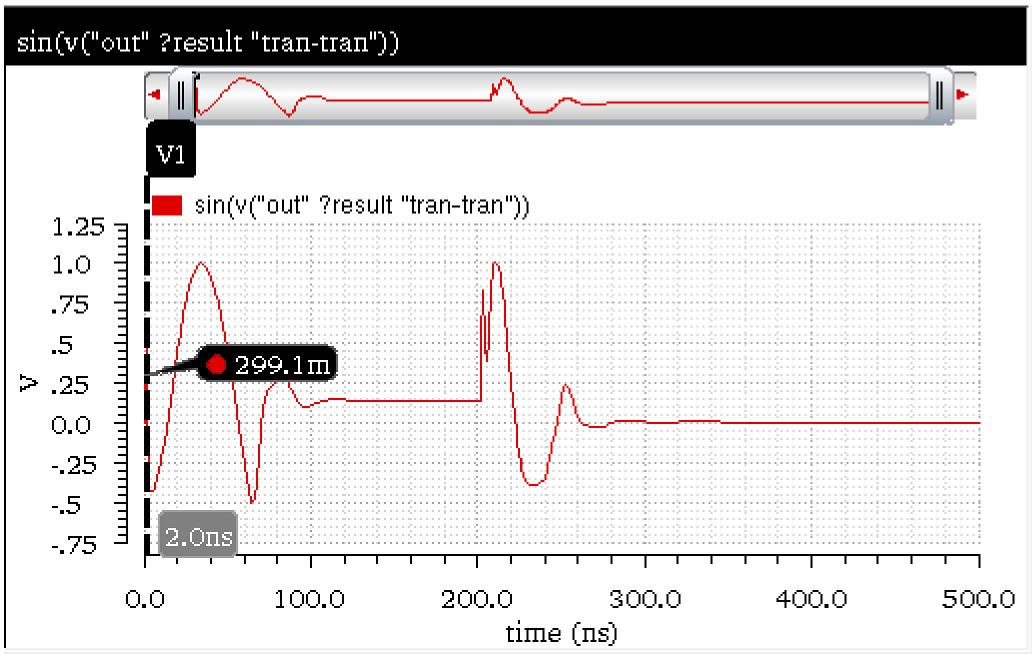
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `sin` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in the Buffer is as follows:

```
sin(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

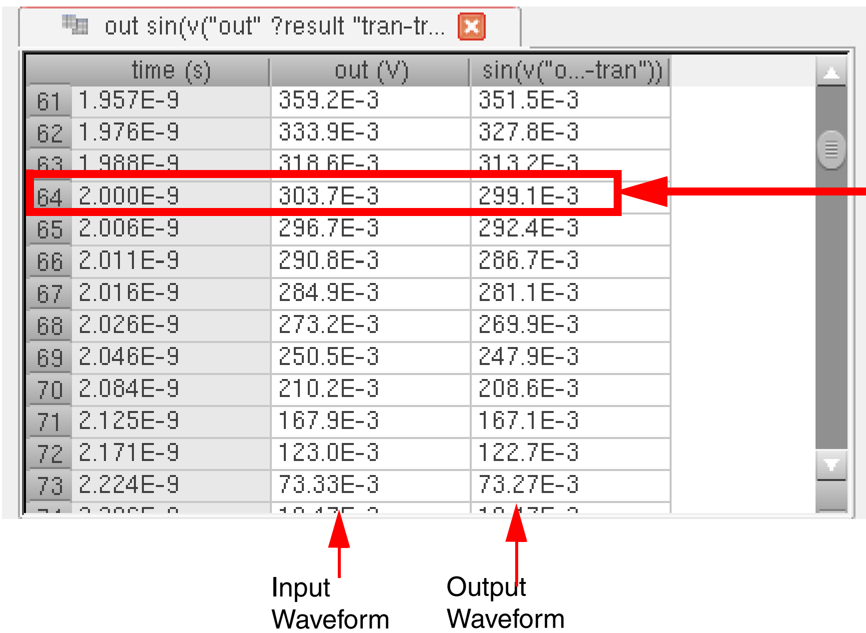
When you evaluate this expression, the function returns the sine values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`299.1mV` when placed at time=`2.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	sin(v("o...-tran"))
61	1.957E-9	359.2E-3	351.5E-3
62	1.976E-9	333.9E-3	327.8E-3
63	1.988E-9	318.6E-3	313.2E-3
64	2.000E-9	303.7E-3	299.1E-3
65	2.006E-9	296.7E-3	292.4E-3
66	2.011E-9	290.8E-3	286.7E-3
67	2.016E-9	284.9E-3	281.1E-3
68	2.026E-9	273.2E-3	269.9E-3
69	2.046E-9	250.5E-3	247.9E-3
70	2.084E-9	210.2E-3	208.6E-3
71	2.125E-9	167.9E-3	167.1E-3
72	2.171E-9	123.0E-3	122.7E-3
73	2.224E-9	73.33E-3	73.27E-3

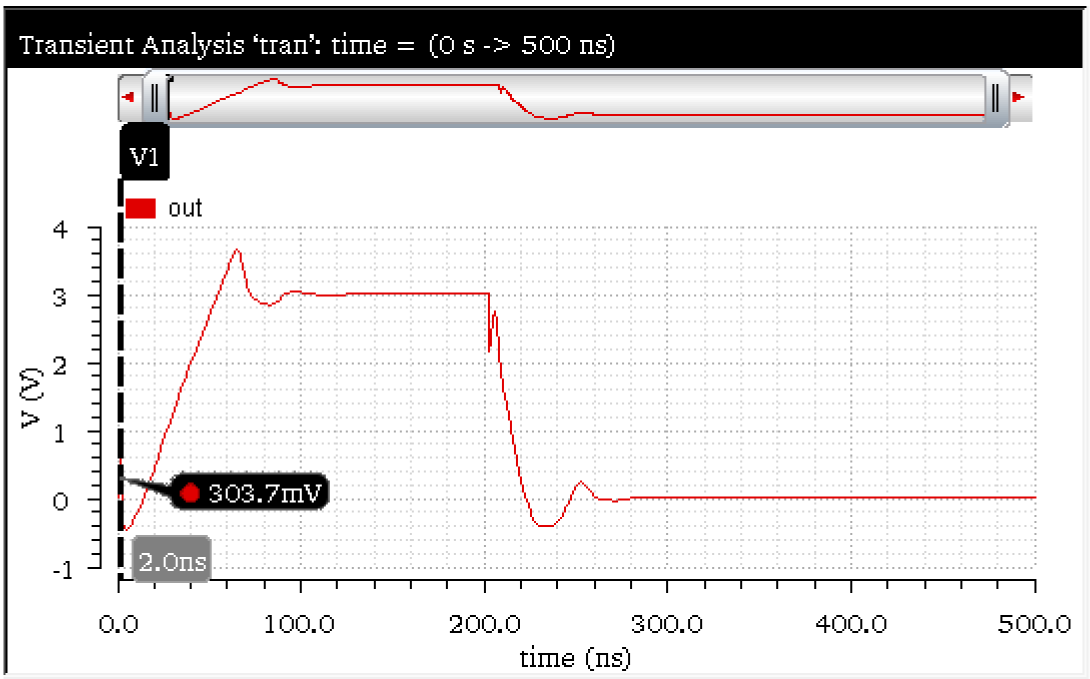
At time=2.00E-9s, the voltage value in the input signal1, out(V), is 303.7E-3V After the sin function is applied, the resulted voltage value at time=2.00E-9s in output signal is 299.1E-3V, which is the sine value of 303.7E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

asin

Returns the inverse sine of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`2.0ns` and voltage=`303.7mV`.

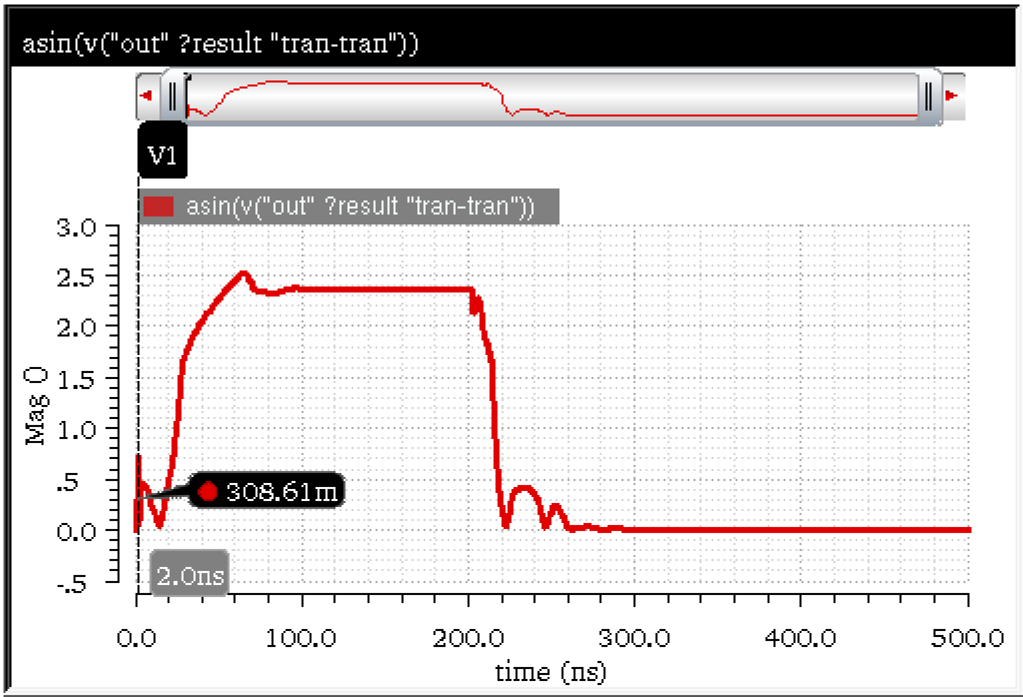
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `asin` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in the Buffer is as follows:

```
asin(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

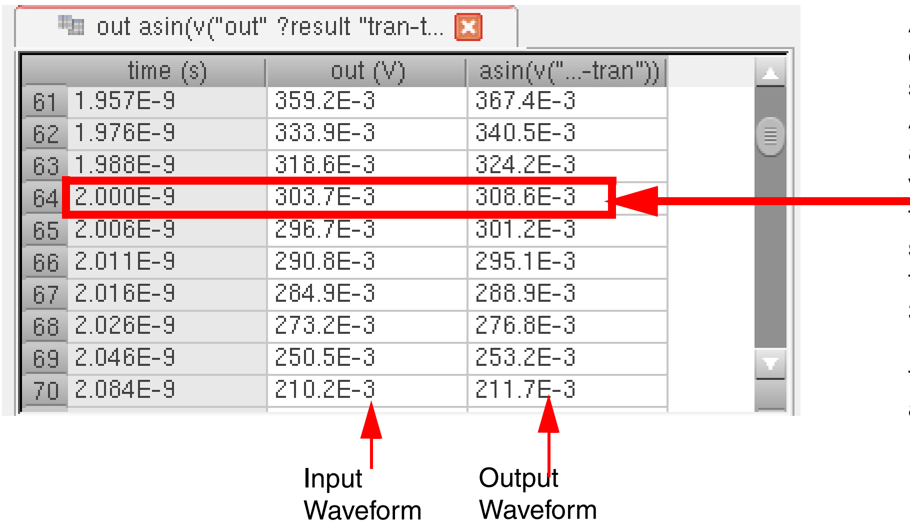
When you evaluate this expression, the function returns the inverse sine values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`308.61mV` when placed at time=`2.0ns`

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out asin(v("out" ?result "tran-t...

	time (s)	out (V)	asin(v("...-tran"))
61	1.957E-9	359.2E-3	367.4E-3
62	1.976E-9	333.9E-3	340.5E-3
63	1.988E-9	318.6E-3	324.2E-3
64	2.000E-9	303.7E-3	308.6E-3
65	2.006E-9	296.7E-3	301.2E-3
66	2.011E-9	290.8E-3	295.1E-3
67	2.016E-9	284.9E-3	288.9E-3
68	2.026E-9	273.2E-3	276.8E-3
69	2.046E-9	250.5E-3	253.2E-3
70	2.084E-9	210.2E-3	211.7E-3

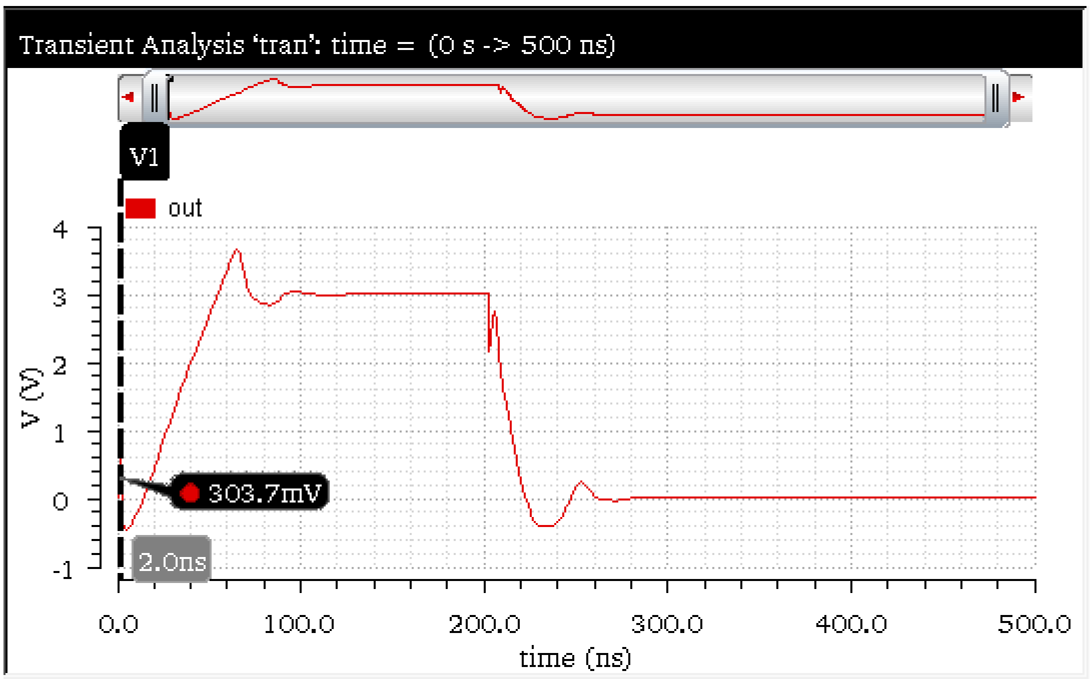
At time=2.00E-9s, the value of voltage in the input signal, out(V), is 303.7E-3V After the asin function is applied, the resultant voltage value at time=2.00E-9s in output signal is 308.6E-3V. This is the inverse sine value of 303.7E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

sinh

Returns the hyperbolic sine of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`2.0ns` and voltage=`303.7mV`.

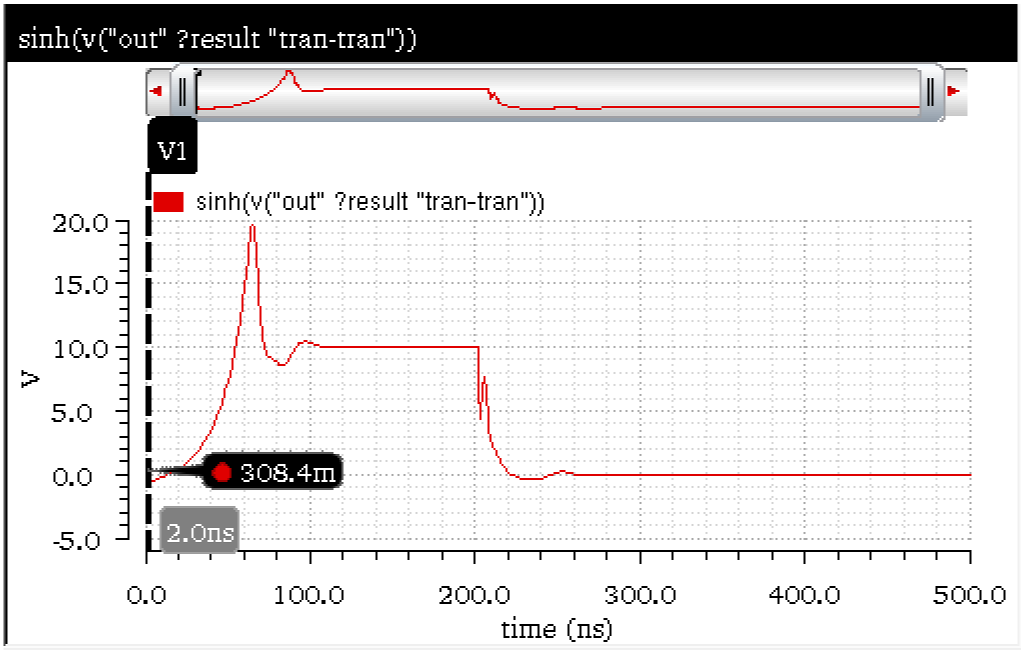
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `sinh` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in the Buffer is as follows:

```
sinh(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

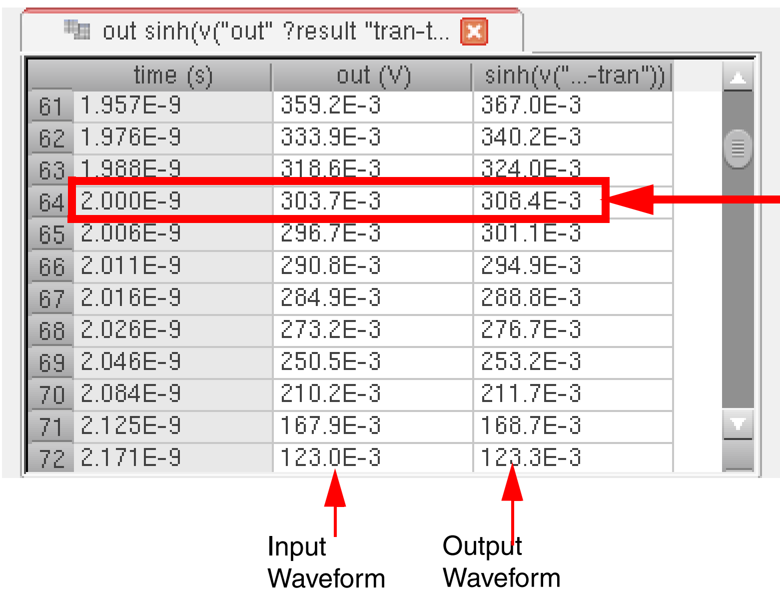
When you evaluate this expression, the function returns the hyperbolic sine values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`308.4mV` when placed at time=`2.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out sinh(v("out" ?result "tran-t...

	time (s)	out (V)	sinh(v("...-tran"))
61	1.957E-9	359.2E-3	367.0E-3
62	1.976E-9	333.9E-3	340.2E-3
63	1.988E-9	318.6E-3	324.0E-3
64	2.000E-9	303.7E-3	308.4E-3
65	2.006E-9	296.7E-3	301.1E-3
66	2.011E-9	290.8E-3	294.9E-3
67	2.016E-9	284.9E-3	288.8E-3
68	2.026E-9	273.2E-3	276.7E-3
69	2.046E-9	250.5E-3	253.2E-3
70	2.084E-9	210.2E-3	211.7E-3
71	2.125E-9	167.9E-3	168.7E-3
72	2.171E-9	123.0E-3	123.3E-3

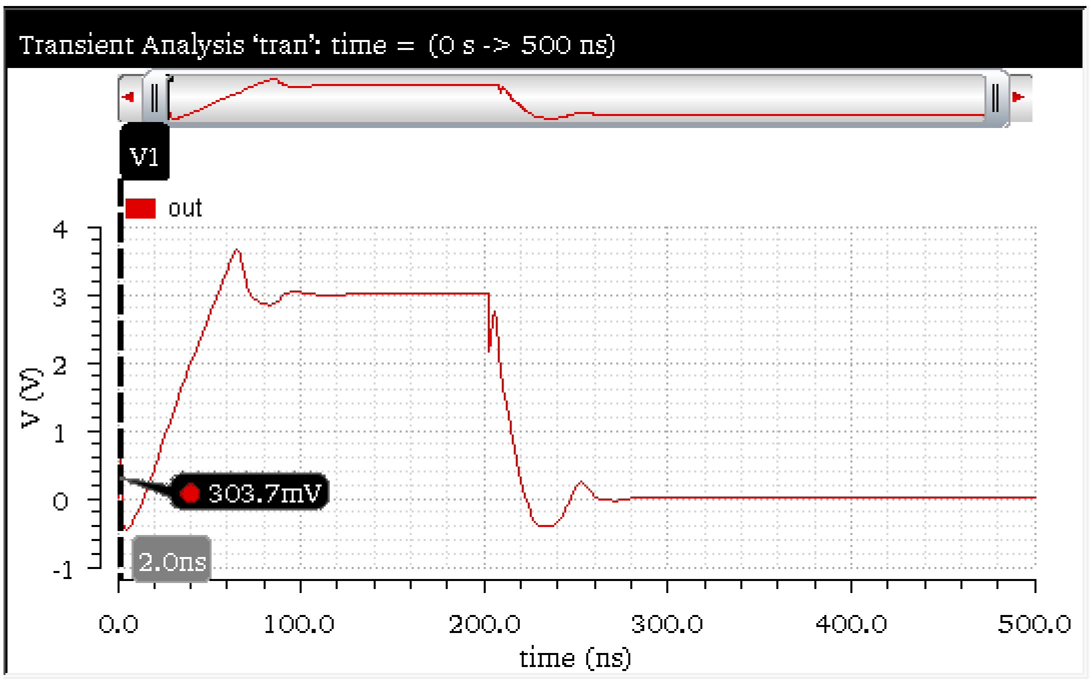
At time=2.00E-9s, the voltage value in the input signal1, out(V), is 303.7E-3V After the sinh function is applied, the resulted voltage value at time=2.00E-9s in output signal is 308.4E-3V, which is the hyperbolic sine value of 303.7E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

asinh

Returns the inverse hyperbolic sine or hyperbolic arc-sine of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`2.0ns` and voltage=`303.7mV`.

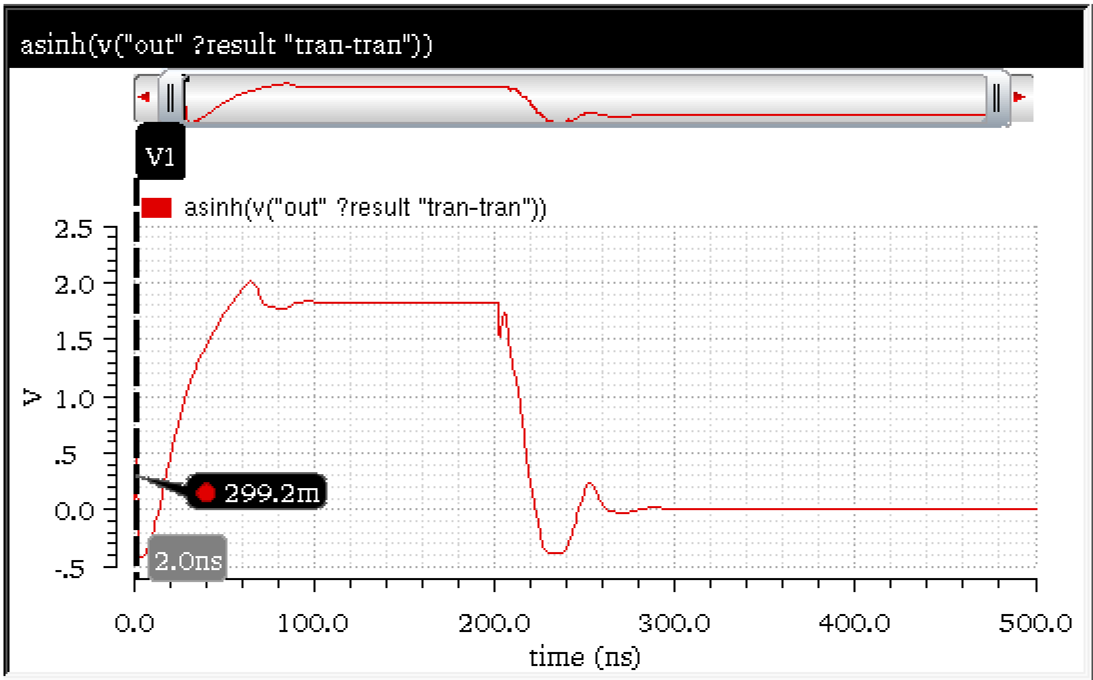
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `asinh` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in the Buffer is as follows:

```
asinh(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

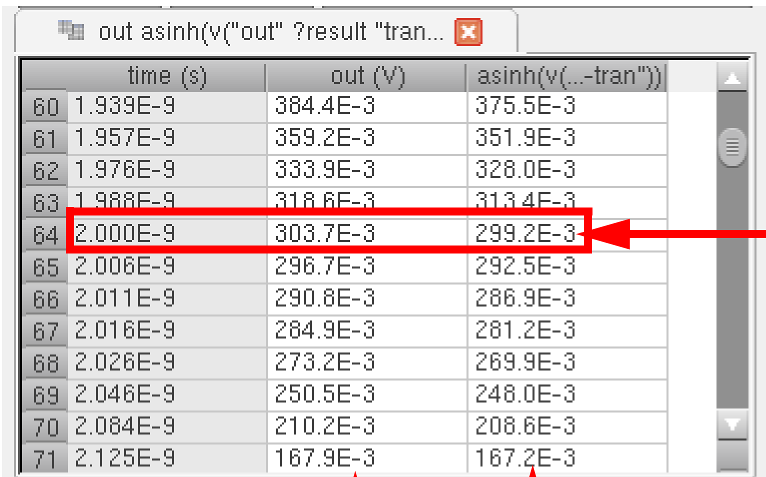
When you evaluate this expression, the function returns the hyperbolic arc-sine values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`299.2mV` when placed at time=`2.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out asinh(v("out" ?result "tran...

	time (s)	out (V)	asinh(v(...-tran"))
60	1.939E-9	384.4E-3	375.5E-3
61	1.957E-9	359.2E-3	351.9E-3
62	1.976E-9	333.9E-3	328.0E-3
63	1.988E-9	318.6E-3	313.4E-3
64	2.000E-9	303.7E-3	299.2E-3
65	2.006E-9	296.7E-3	292.5E-3
66	2.011E-9	290.8E-3	286.9E-3
67	2.016E-9	284.9E-3	281.2E-3
68	2.026E-9	273.2E-3	269.9E-3
69	2.046E-9	250.5E-3	248.0E-3
70	2.084E-9	210.2E-3	208.6E-3
71	2.125E-9	167.9E-3	167.2E-3

Input Waveform

Output Waveform

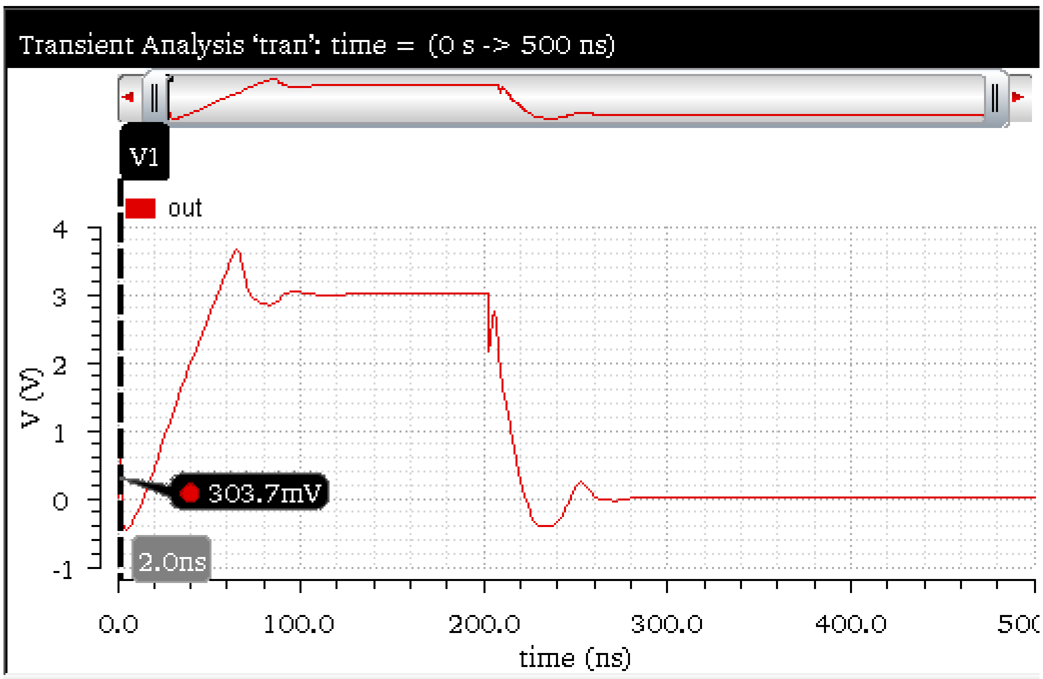
At time=2.00E-9s, the value of voltage in the input signal, out(V), is 303.7E-3V After the asinh function is applied, the resultant voltage value at time=2.00E-9s in output signal is 299.2E-3V. This is the arc hyperbolic sine value of 303.7E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

tan

Returns the tangent of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`2.0ns` and voltage=`303.7mV`.

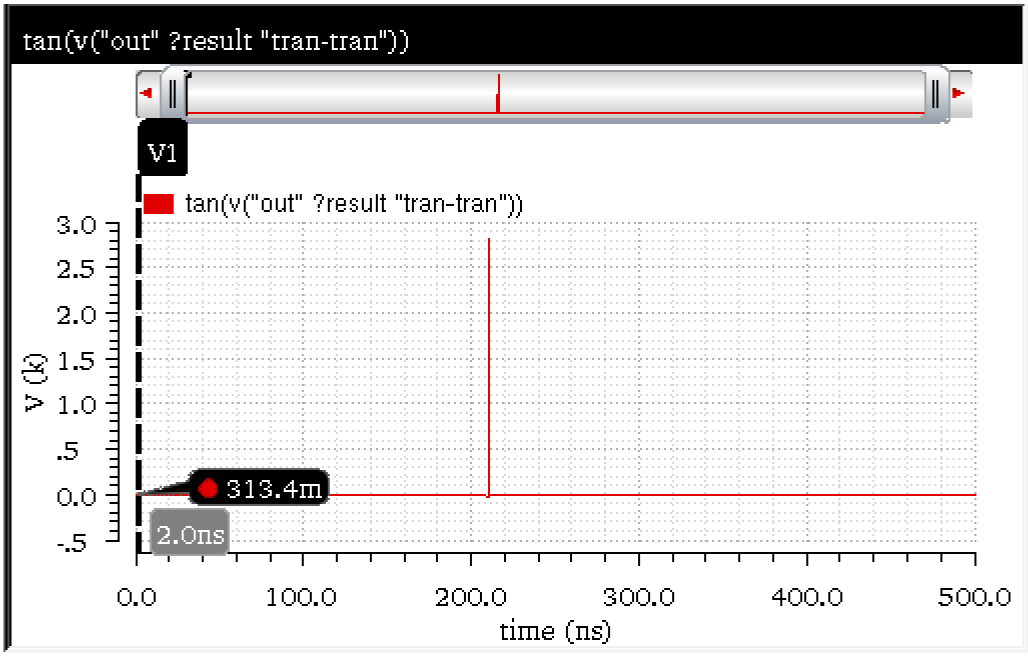
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `tan` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in the Buffer is as follows:

```
tan(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

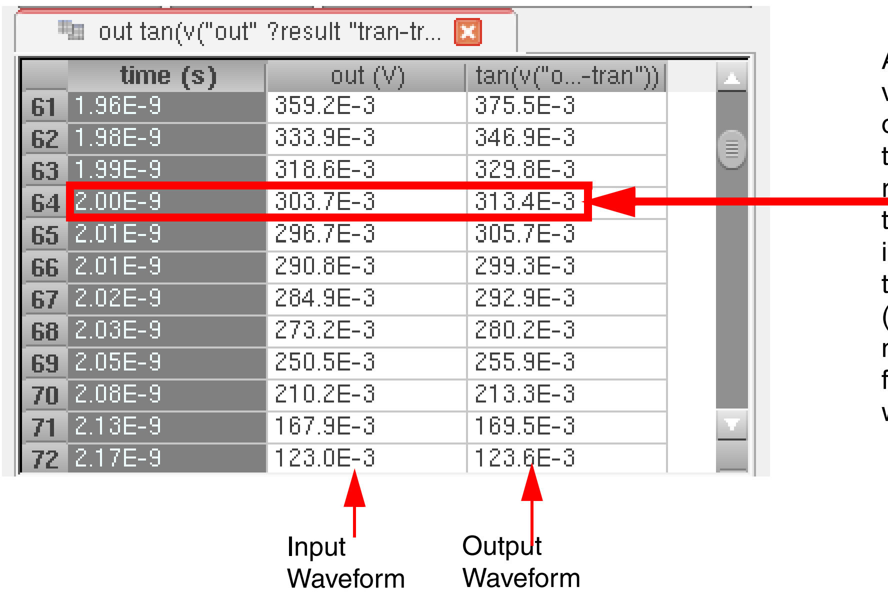
When you evaluate this expression, the function returns the tangent values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`313.4mV` when placed at time=`2.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out tan(v("out" ?result "tran-tr...

	time (s)	out (V)	tan(v("o...-tran"))
61	1.96E-9	359.2E-3	375.5E-3
62	1.98E-9	333.9E-3	346.9E-3
63	1.99E-9	318.6E-3	329.8E-3
64	2.00E-9	303.7E-3	313.4E-3
65	2.01E-9	296.7E-3	305.7E-3
66	2.01E-9	290.8E-3	299.3E-3
67	2.02E-9	284.9E-3	292.9E-3
68	2.03E-9	273.2E-3	280.2E-3
69	2.05E-9	250.5E-3	255.9E-3
70	2.08E-9	210.2E-3	213.3E-3
71	2.13E-9	167.9E-3	169.5E-3
72	2.17E-9	123.0E-3	123.6E-3

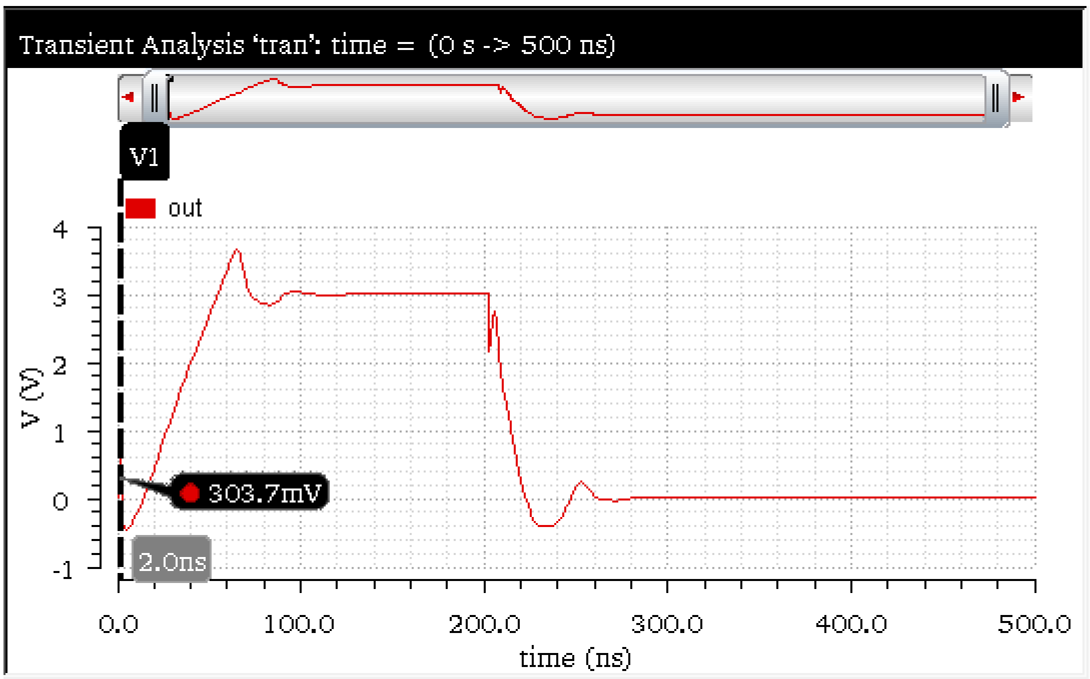
At time=2.00E-9s, the value of voltage in the input signal, out(V), is 303.7E-3V After the tan function is applied, the resultant voltage value at time=2.00E-9s in output signal is 313.4E-3. This is the tangent value of 303.7E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

atan

Returns the nverse tangent of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`2.0ns` and voltage=`303.7mV`.

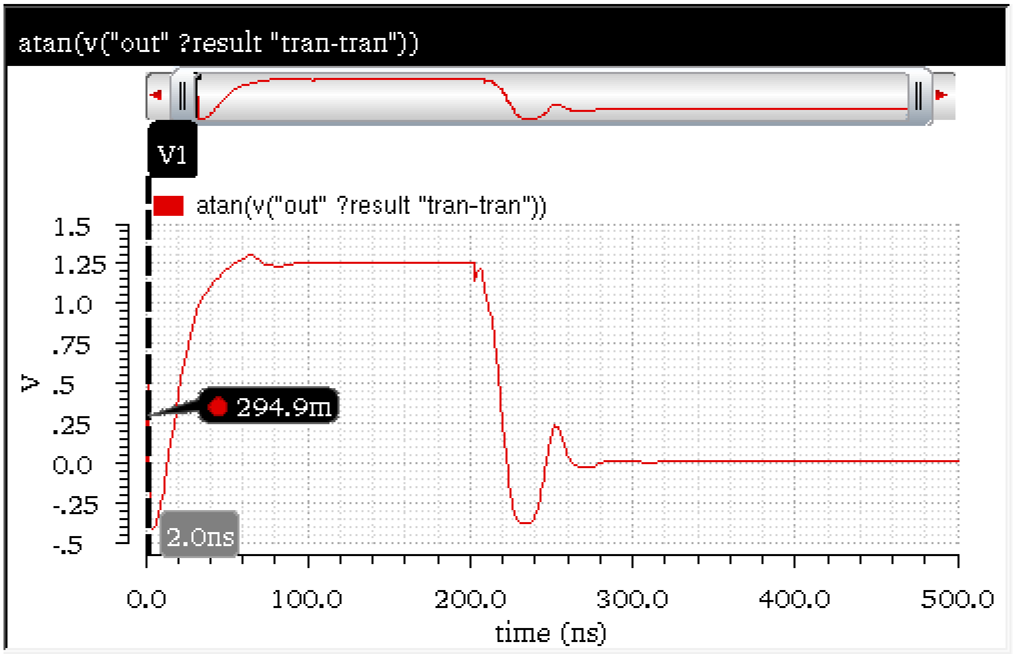
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `atan` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in the Buffer is as follows:

```
atan(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

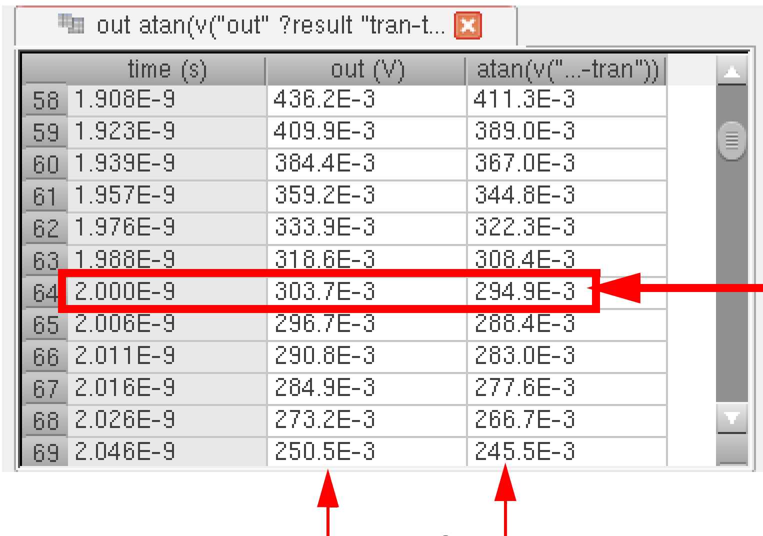
When you evaluate this expression, the function returns the inverse tangent values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`294.9mV` when placed at time=`2.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out atan(v("out" ?result "tran-t...

	time (s)	out (V)	atan(v("...-tran"))
58	1.908E-9	436.2E-3	411.3E-3
59	1.923E-9	409.9E-3	389.0E-3
60	1.939E-9	384.4E-3	367.0E-3
61	1.957E-9	359.2E-3	344.8E-3
62	1.976E-9	333.9E-3	322.3E-3
63	1.988E-9	318.6E-3	308.4E-3
64	2.000E-9	303.7E-3	294.9E-3
65	2.006E-9	296.7E-3	288.4E-3
66	2.011E-9	290.8E-3	283.0E-3
67	2.016E-9	284.9E-3	277.6E-3
68	2.026E-9	273.2E-3	266.7E-3
69	2.046E-9	250.5E-3	245.5E-3

At time=2.00E-9s, the value of voltage in the input signal, out(V), is 303.7E-3V. After the atan function is applied, the resultant voltage value at time=2.00E-9s in output signal is 294.9E-3. This is the inverse tangent value of 303.7E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

Input Waveform

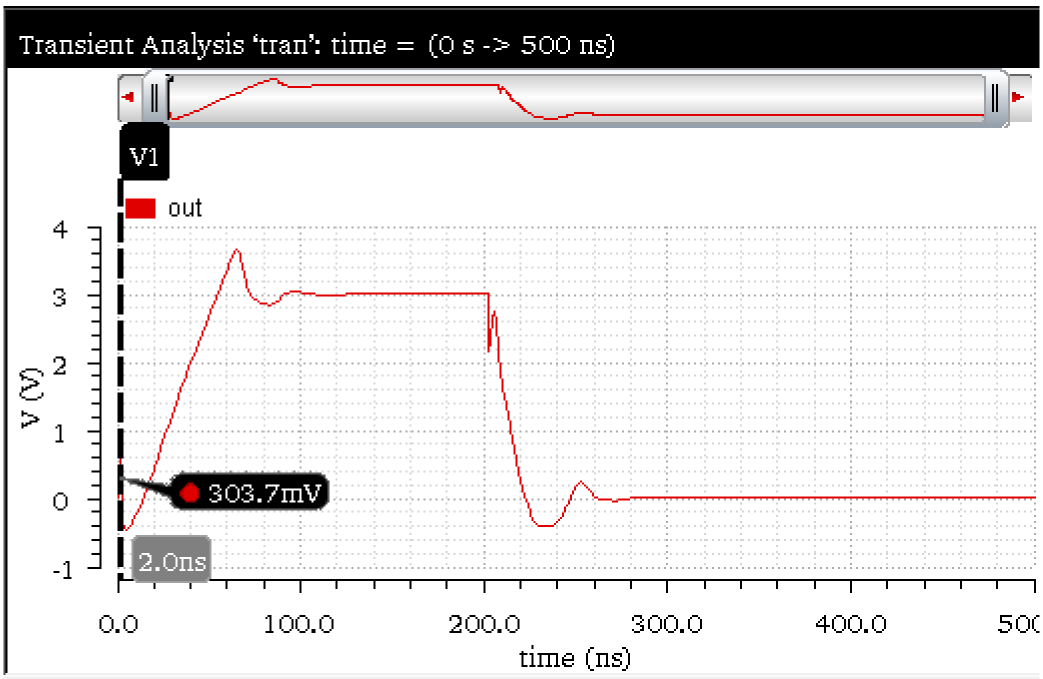
Output Waveform

tanh

Returns the hyperbolic tangent of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`2.0ns` and voltage=`303.7mV`.

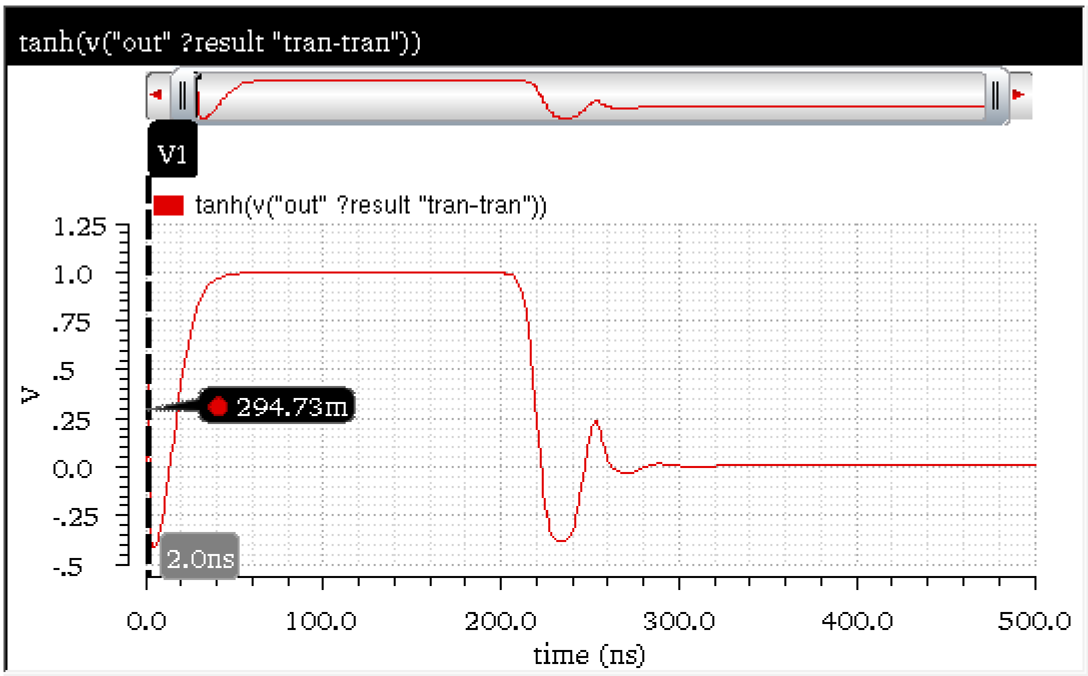
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `tanh` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in the Buffer is as follows:

```
tanh(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

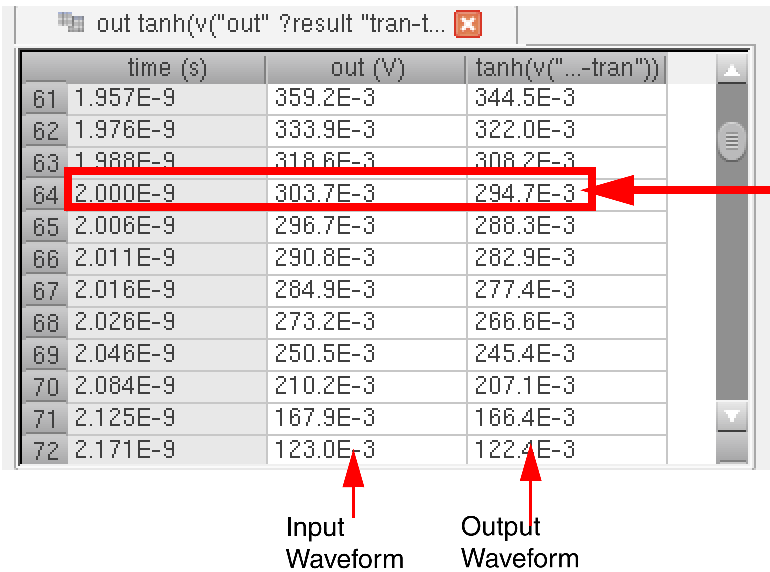
When you evaluate this expression, the function returns the hyperbolic tangent values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`294.93mV` when placed at time=`2.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out tanh(v("out" ?result "tran-t...

	time (s)	out (V)	tanh(v("...-tran"))
61	1.957E-9	359.2E-3	344.5E-3
62	1.976E-9	333.9E-3	322.0E-3
63	1.988E-9	318.6E-3	308.2E-3
64	2.000E-9	303.7E-3	294.7E-3
65	2.006E-9	296.7E-3	288.3E-3
66	2.011E-9	290.8E-3	282.9E-3
67	2.016E-9	284.9E-3	277.4E-3
68	2.026E-9	273.2E-3	266.6E-3
69	2.046E-9	250.5E-3	245.4E-3
70	2.084E-9	210.2E-3	207.1E-3
71	2.125E-9	167.9E-3	166.4E-3
72	2.171E-9	123.0E-3	122.4E-3

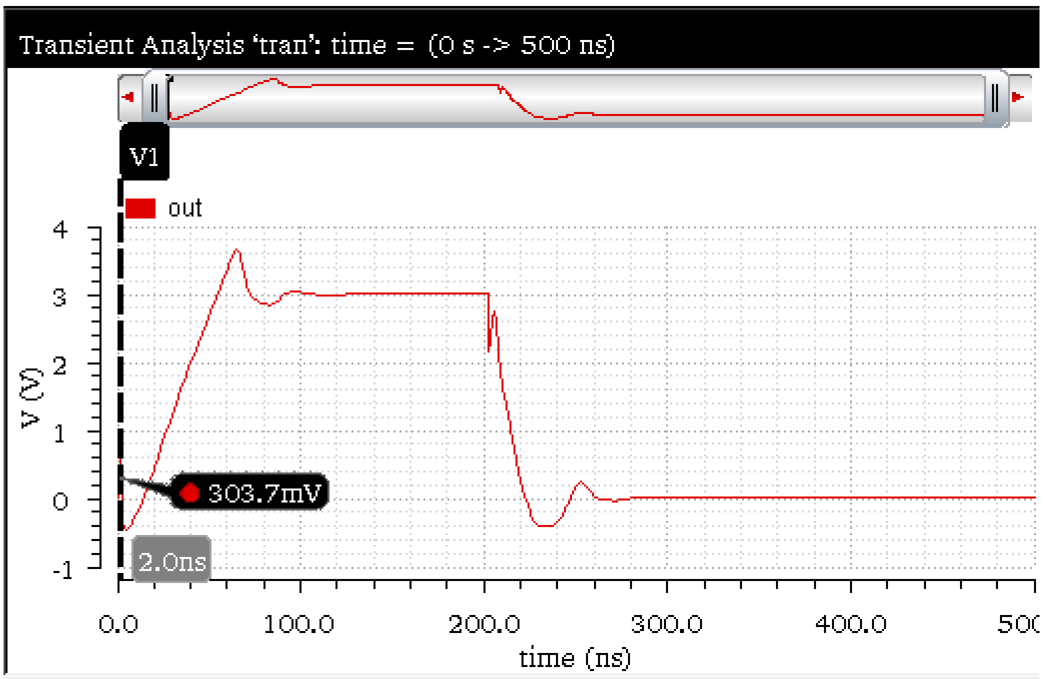
At time=2.00E-9s, the value of voltage in the input signal, out(V), is 303.7E-3V .After the tanh function is applied, the resultant voltage value at time=2.00E-9s in output signal is 294.7E-3. This is the hyperbolic tangent value of 303.7E-3. (You can also refer to vertical markers in the above figures for input and

atanh

Returns the inverse hyperbolic tangent or hyperbolic arc tangent of a signal.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`2.0ns` and voltage=`303.7mV`.

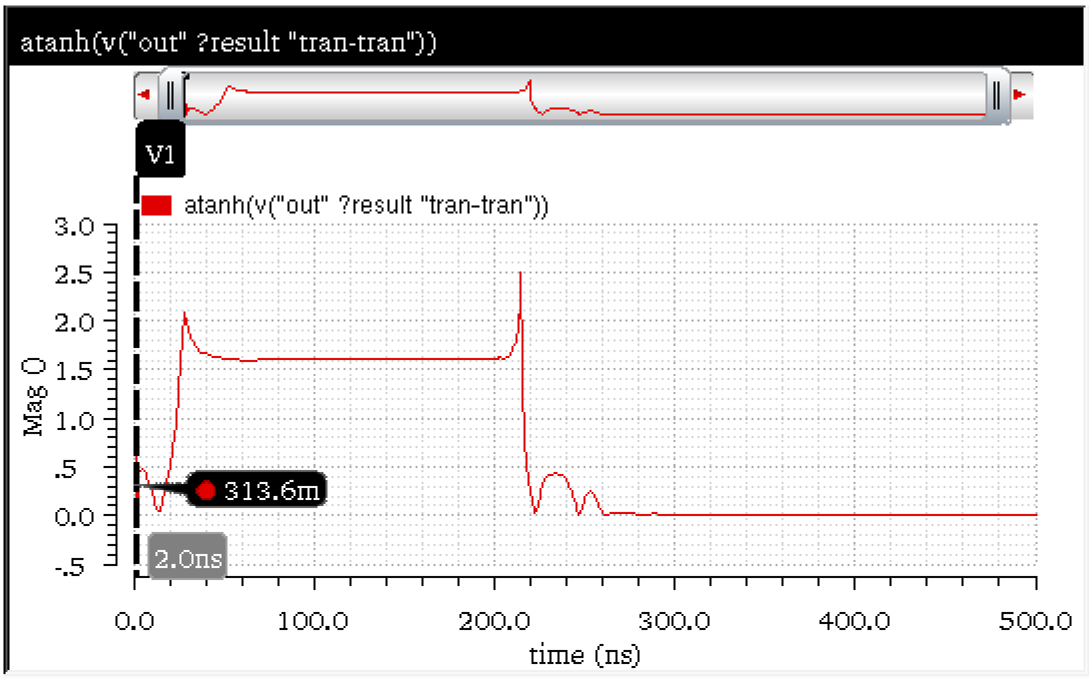
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `atanh` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in the Buffer is as follows:

```
atanh(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

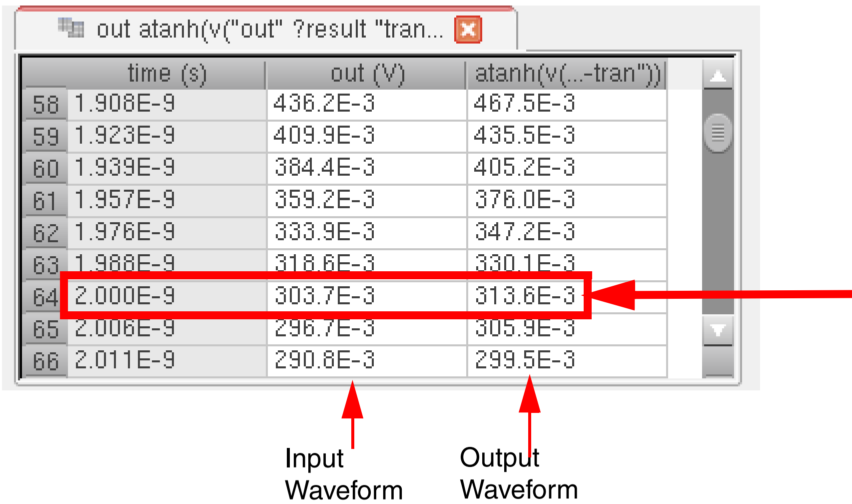
When you evaluate this expression, the function returns the arc hyperbolic tangent values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`313.6mV` when placed at time=`2.0ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	atanh(v(...-tran"))
58	1.908E-9	436.2E-3	467.5E-3
59	1.923E-9	409.9E-3	435.5E-3
60	1.939E-9	384.4E-3	405.2E-3
61	1.957E-9	359.2E-3	376.0E-3
62	1.976E-9	333.9E-3	347.2E-3
63	1.988E-9	318.6E-3	330.1E-3
64	2.000E-9	303.7E-3	313.6E-3
65	2.006E-9	296.7E-3	305.9E-3
66	2.011E-9	290.8E-3	299.5E-3

At time=2.00E-9s, the value of voltage in the input signal, out(V), is 303.7E-3V. After the tanh function is applied, the resultant voltage value at time=2.00E-9s in output signal is 313.6E-3. This is the arc hyperbolic tangent value of 303.7E-3. (You can also refer to vertical markers in the above figures for input and output waveforms)

Math Functions

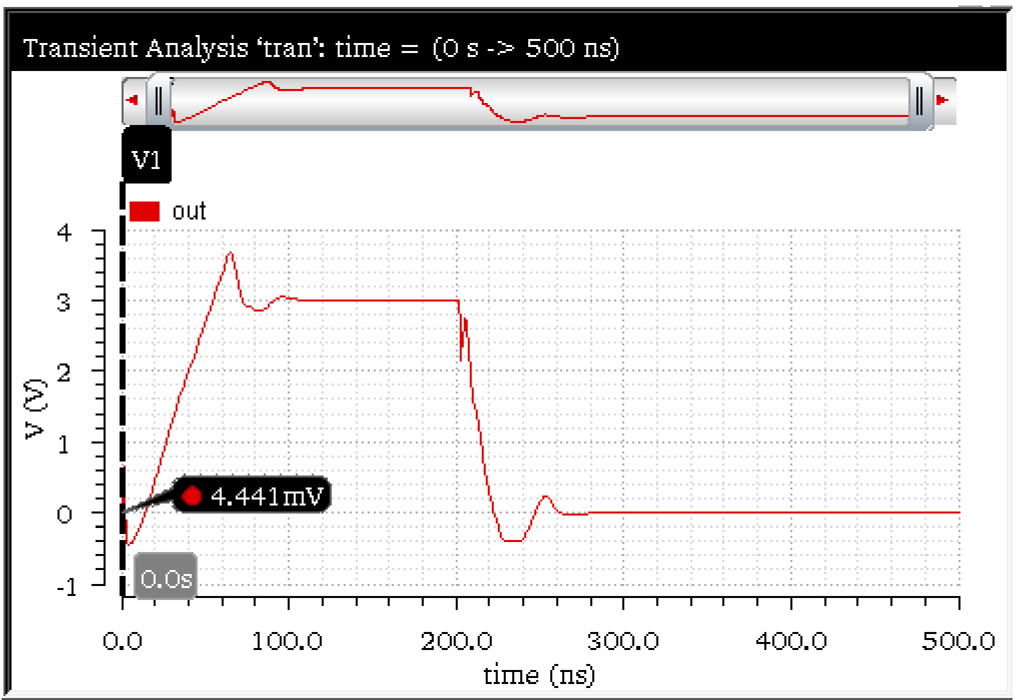
This section covers the following Math functions in Calculator:

1/x

Returns the inverse value. This function is available only in the SKILL mode.

Example

Consider the following signal from the transient analysis:



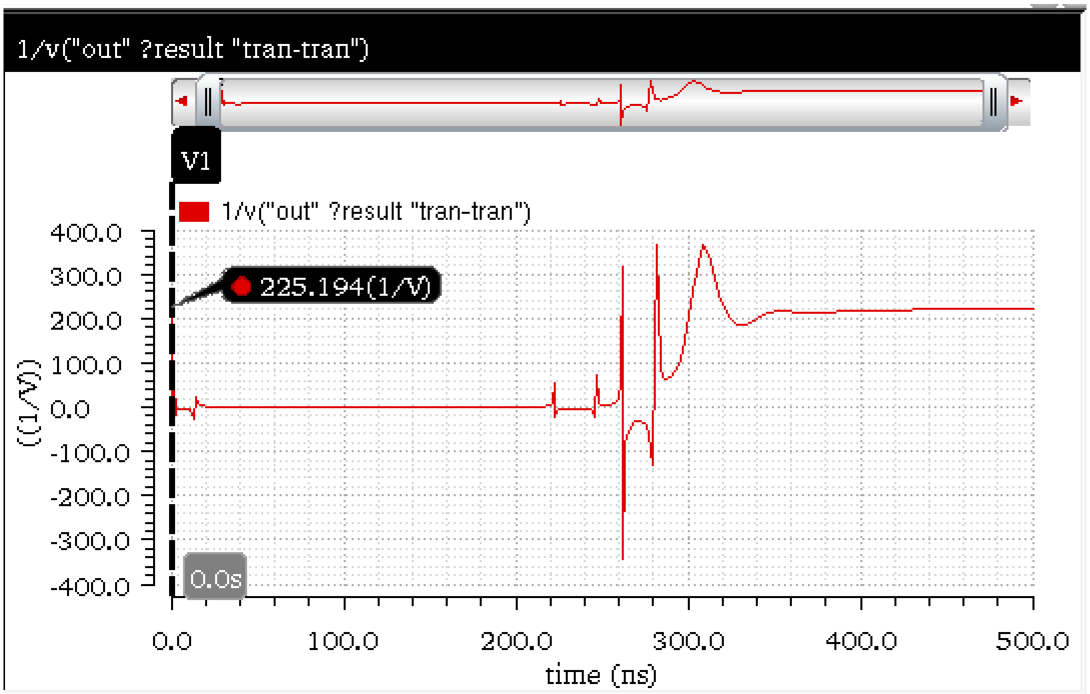
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `1/x` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
1/v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

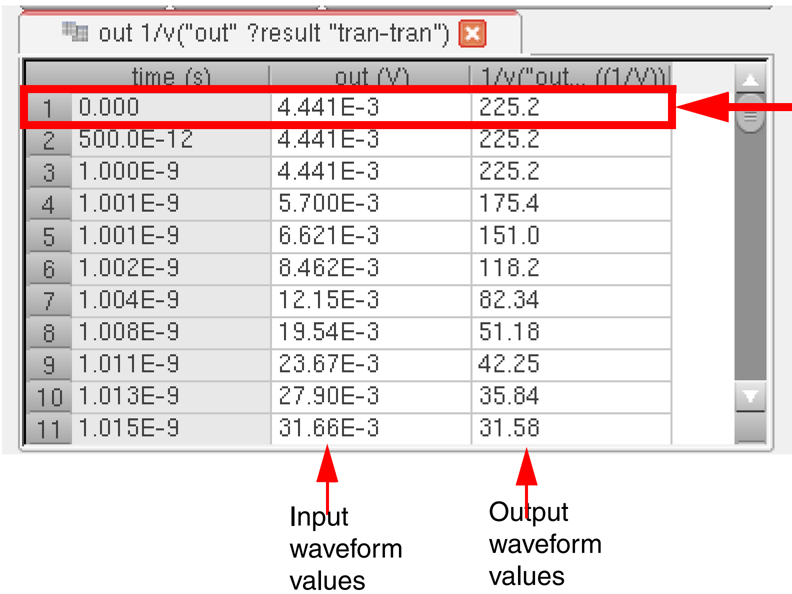
When you evaluate this expression, the inverse of the signal is calculated and the following output signal is displayed in a new graph window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`225.194V` when placed at time=`0.0ns`, which is the inverse of the voltage value (`1/4.441mV`) shown in the input waveform.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	1/v("out... ((1/V))
1	0.000	4.441E-3	225.2
2	500.0E-12	4.441E-3	225.2
3	1.000E-9	4.441E-3	225.2
4	1.001E-9	5.700E-3	175.4
5	1.001E-9	6.621E-3	151.0
6	1.002E-9	8.462E-3	118.2
7	1.004E-9	12.15E-3	82.34
8	1.008E-9	19.54E-3	51.18
9	1.011E-9	23.67E-3	42.25
10	1.013E-9	27.90E-3	35.84
11	1.015E-9	31.66E-3	31.58

At time=0.00ns, the voltage value in the input signal, out(V), is 4.441E-3. After the 1/x function is applied, the inverse voltage value at time=0.00ns in output signal is calculated as 225.2V. (Also, see vertical markers in the above figures for input and output waveforms)

Related OCEAN Function

The equivalent OCEAN command for `1/x` is:

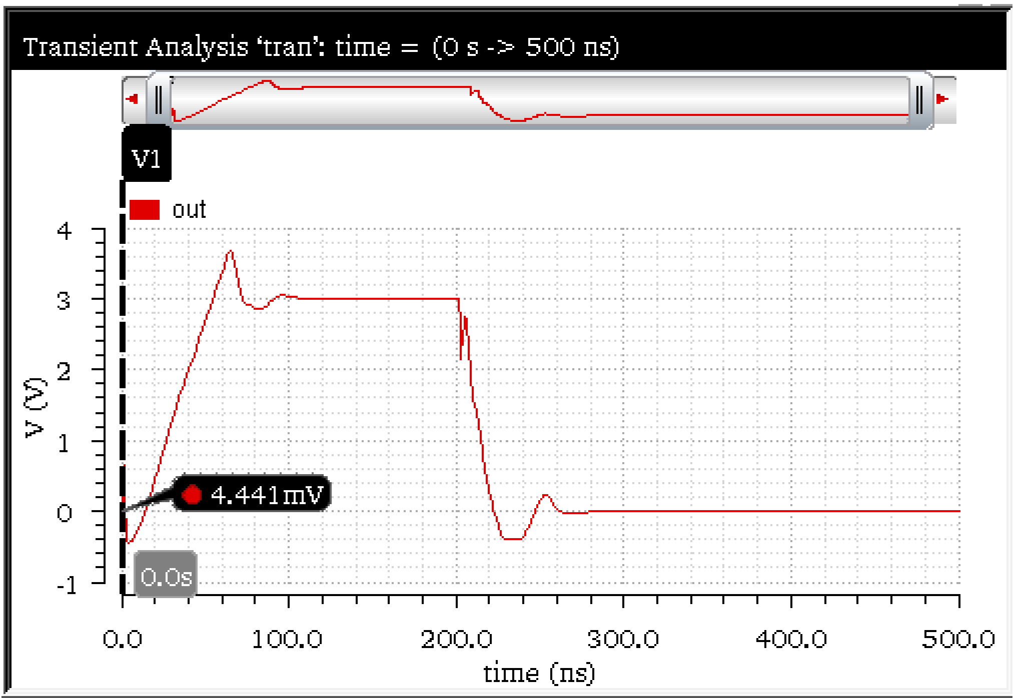
For more information, see 1/x in *OCEAN Reference*.

10**x

Returns the 10^x value of the input signal. This function is available only in the SKILL mode.

Example

Consider the following input signal form a transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0ns` and voltage=`4.441mv`.

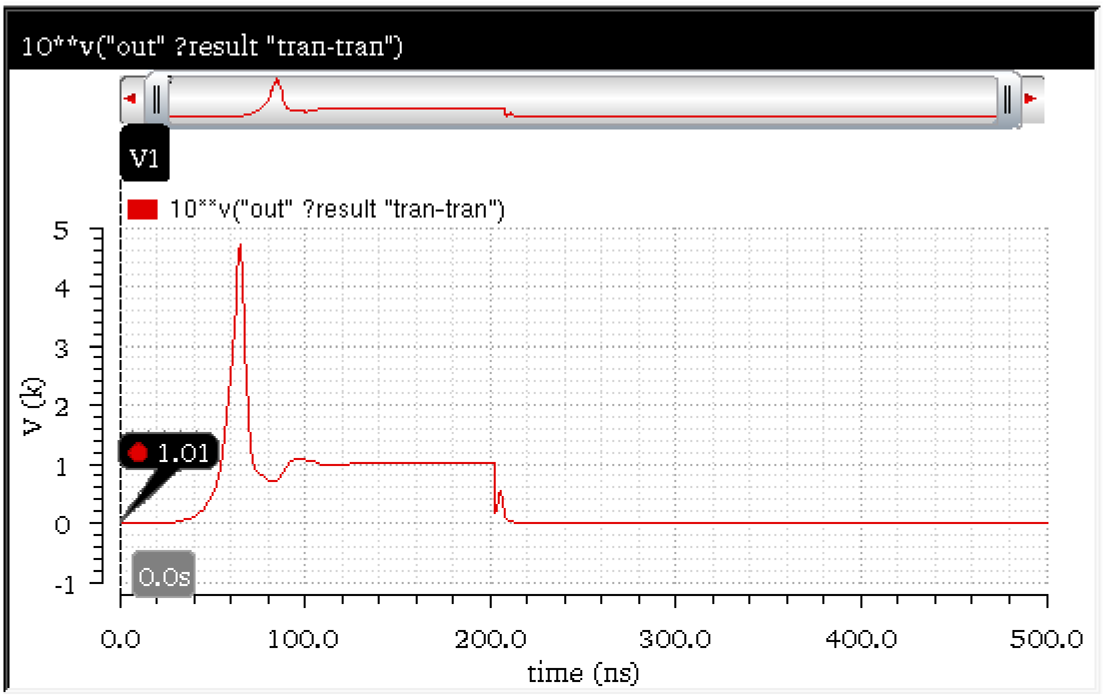
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `10**x` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
10**v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

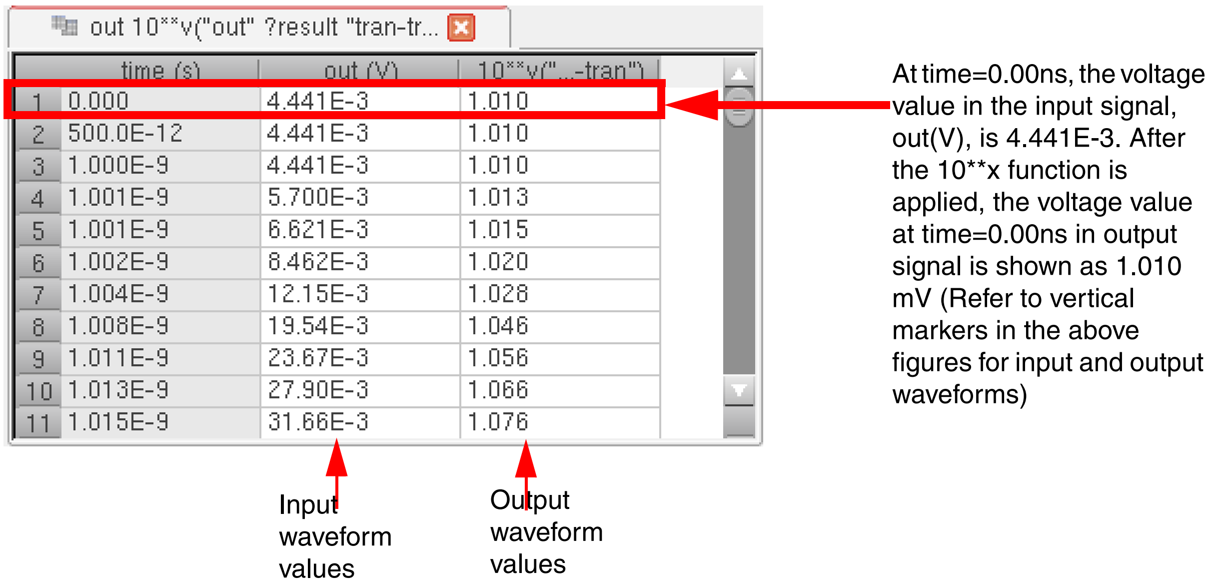
When you evaluate this expression, the 10^x of the signal is calculated and the following output is displayed in a new graph window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`1.01mV` when placed at time=`0.0ns`, which is the 10^x of the voltage value (`4.441mV`) shown in the input waveform.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out 10**v("out" ?result "tran-tr...

	time (s)	out (V)	10**v("...-tran")
1	0.000	4.441E-3	1.010
2	500.0E-12	4.441E-3	1.010
3	1.000E-9	4.441E-3	1.010
4	1.001E-9	5.700E-3	1.013
5	1.001E-9	6.621E-3	1.015
6	1.002E-9	8.462E-3	1.020
7	1.004E-9	12.15E-3	1.028
8	1.008E-9	19.54E-3	1.046
9	1.011E-9	23.67E-3	1.056
10	1.013E-9	27.90E-3	1.066
11	1.015E-9	31.66E-3	1.076

Related OCEAN Function

The equivalent OCEAN command for `10**x` is:

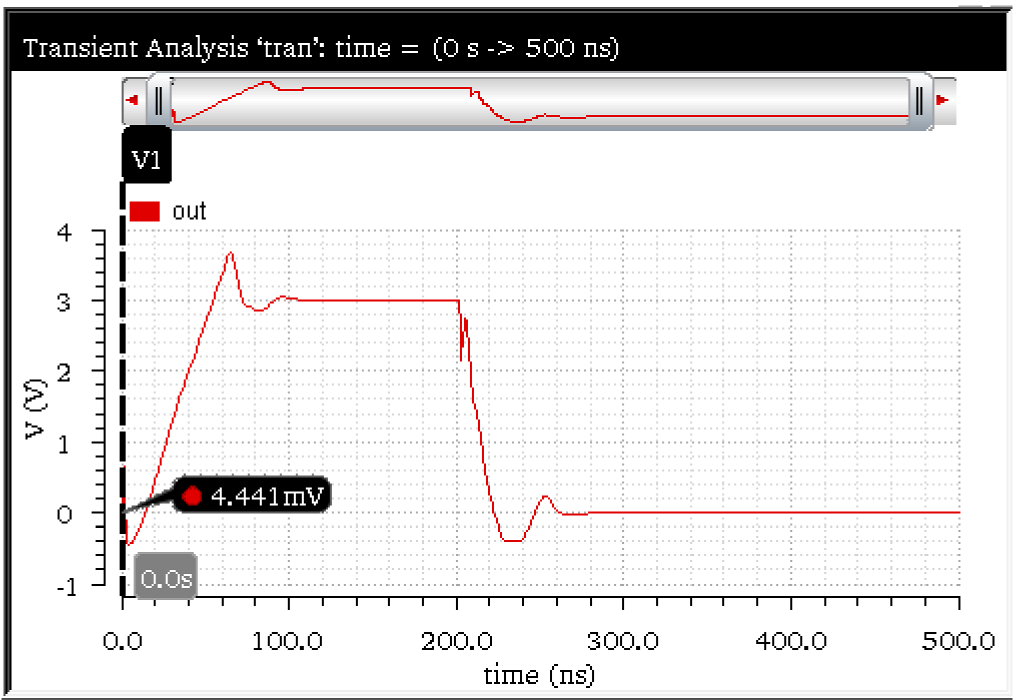
For more information, see `10**x` in *OCEAN Reference*.

exp

Returns the exponential value (e^x) of the signal.

Example

Consider the following input signal from transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0ns` and voltage=`4.441mv`.

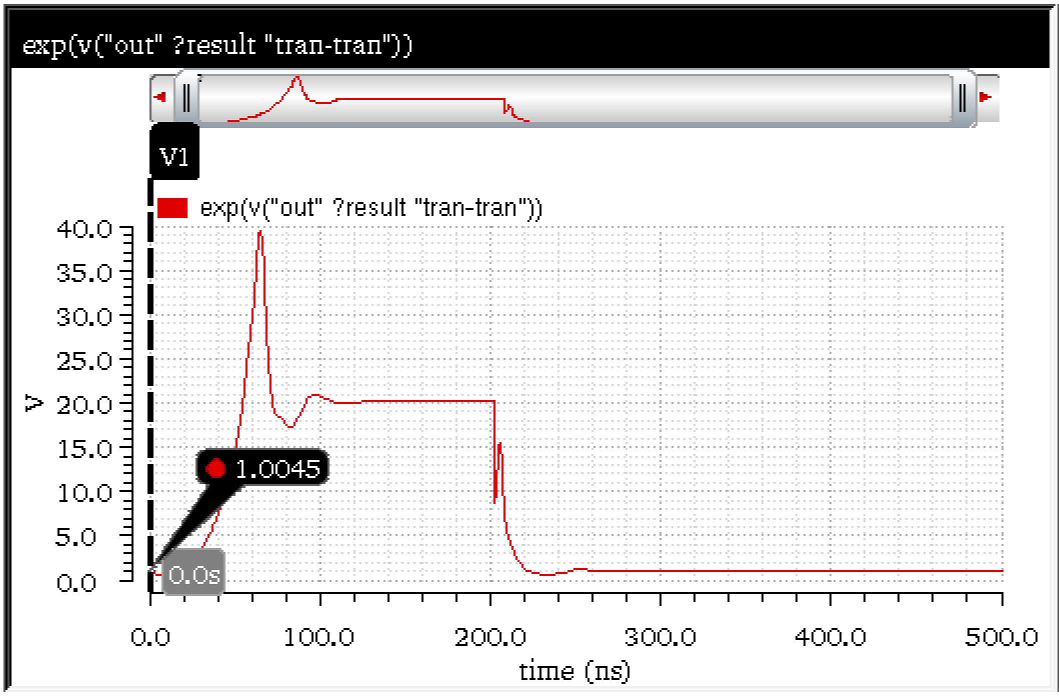
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `exp` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
exp(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

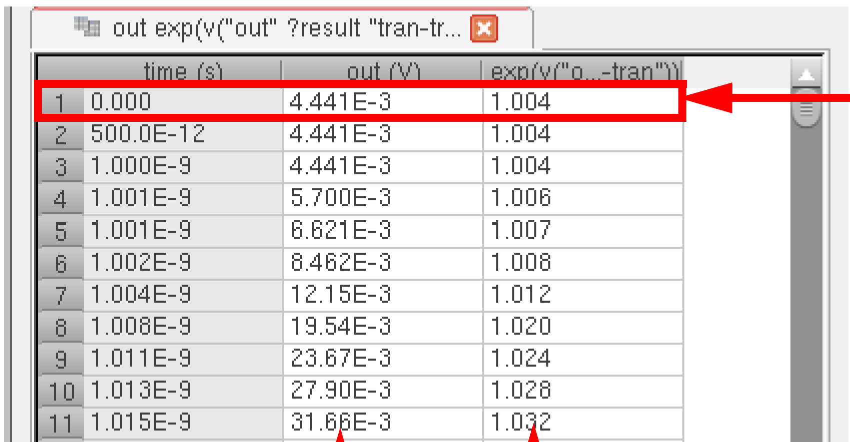
When you evaluate this expression, the exponential value of the input signal is calculated and the following output waveform is plotted in a new graph window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`1.0045mV` when placed at time=`0.0ns`, which is the exponential of the voltage value (`4.441mV`) shown in the input waveform.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	exp(v("o...-tran"))
1	0.000	4.441E-3	1.004
2	500.0E-12	4.441E-3	1.004
3	1.000E-9	4.441E-3	1.004
4	1.001E-9	5.700E-3	1.006
5	1.001E-9	6.621E-3	1.007
6	1.002E-9	8.462E-3	1.008
7	1.004E-9	12.15E-3	1.012
8	1.008E-9	19.54E-3	1.020
9	1.011E-9	23.67E-3	1.024
10	1.013E-9	27.90E-3	1.028
11	1.015E-9	31.66E-3	1.032

Input waveform values

Output waveform values

At time=0.00ns, the voltage value in the input signal, out(V), is 4.441E-3. After the exp function is applied, the voltage value at time=0.00ns in output signal is shown as 1.004. (Also, see vertical markers in the above figures for input and output waveforms)

Related OCEAN Function

The equivalent OCEAN command for `exp` is:

```
exp( n_number )
     => f_result
```

For more information, see `exp` in *OCEAN Reference*.

int

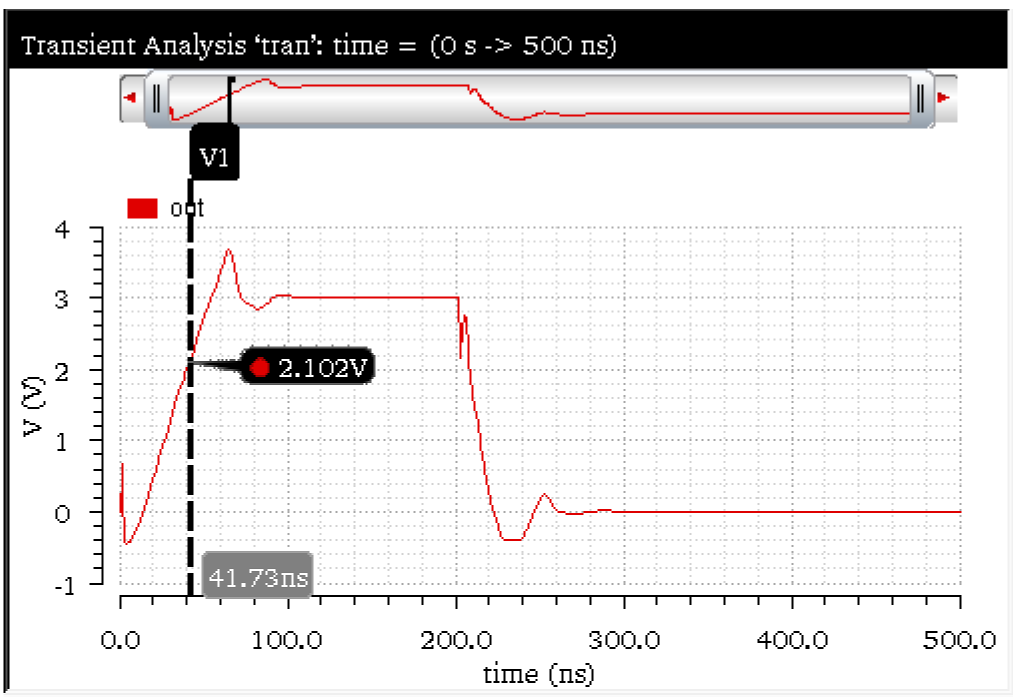
Returns the integer portion of a real value.

Note: While performing the division of two integer or floating point values, if denominator is greater than numerator, the division results are truncated. For example, if you perform the division of `1/2`, the `int` function displays the result as `0` instead of `0.5`.

For more information about how the integer and floating point division is performed using SKILL, see Integer vs. Floating-Point Division in the in the Arithmetic and Logical Expressions chapter of *Cadence SKILL Language User Guide*.

Example

Consider the following input signal from the transient analysis:



Note that a vertical marker `V1` is placed at time=`41.73ns` and voltage=`2.102V`.

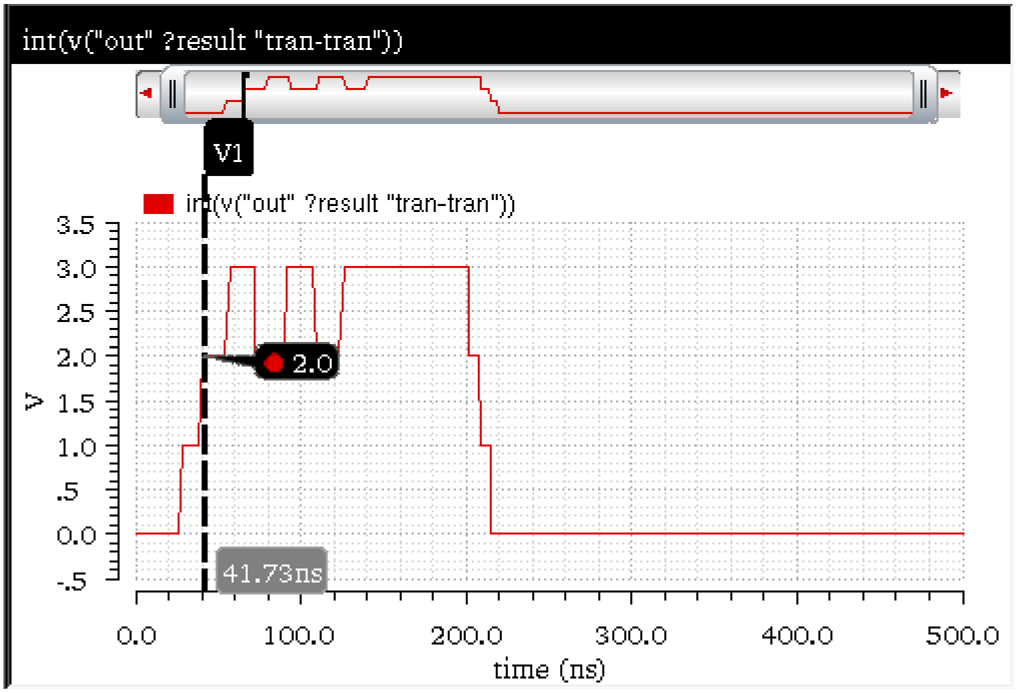
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `int` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
int(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

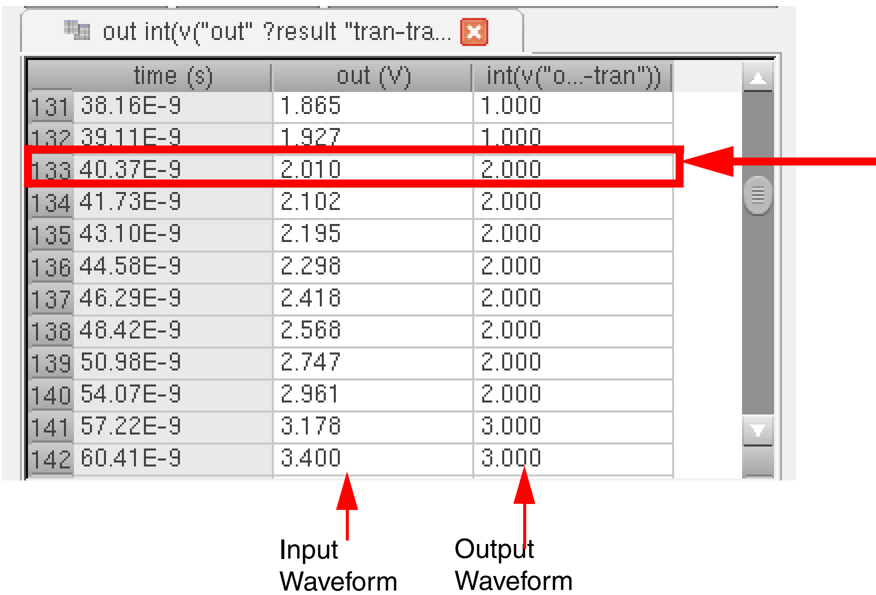
When you evaluate this expression, the function returns the integer voltage values of the input waveform. The following output waveform is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`2.0V` when placed at time=`41.73ns`.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	int(v("o...-tran"))
131	38.16E-9	1.865	1.000
132	39.11E-9	1.927	1.000
133	40.37E-9	2.010	2.000
134	41.73E-9	2.102	2.000
135	43.10E-9	2.195	2.000
136	44.58E-9	2.298	2.000
137	46.29E-9	2.418	2.000
138	48.42E-9	2.568	2.000
139	50.98E-9	2.747	2.000
140	54.07E-9	2.961	2.000
141	57.22E-9	3.178	3.000
142	60.41E-9	3.400	3.000

At time=41.73ns, the voltage value in the input signal1, out(V), is 2.102. After the int function is applied, the resulted voltage value at time=41.73ns in output signal is shown as 2.000, which shows the integer value from 2.102. (Refer to vertical markers in the above figures for input and output waveforms)

Related OCEAN Function

The equivalent OCEAN command for `int` is:

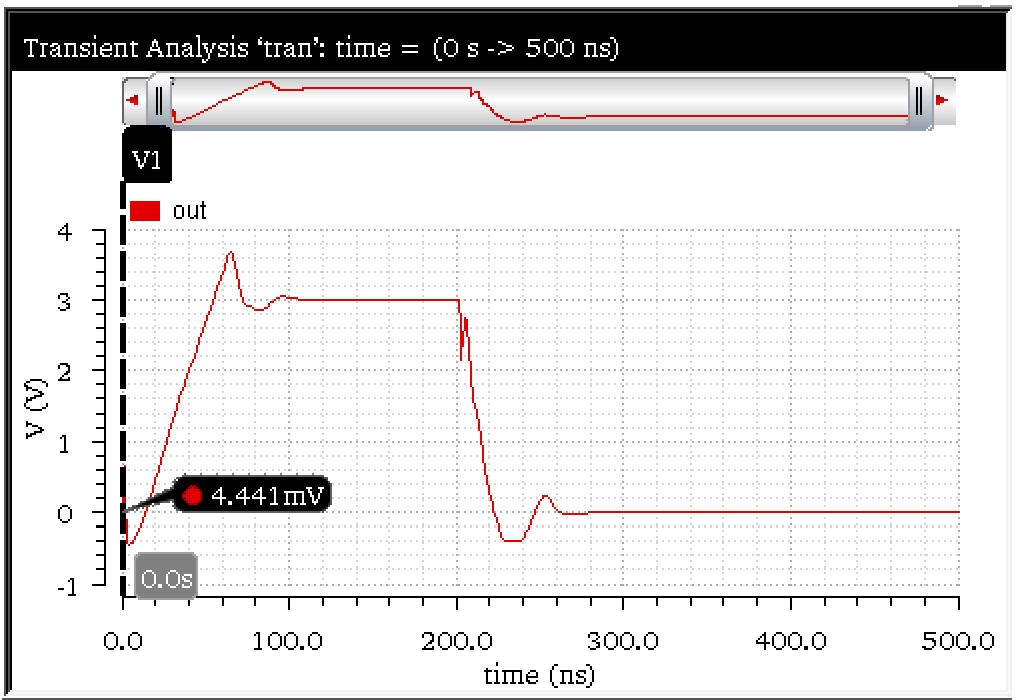
For more information, see `int` in *OCEAN Reference*.

ln

Returns the natural logarithm value of the given signal.

Example

Consider the following input signal from transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0ns` and voltage=`4.441mv`.

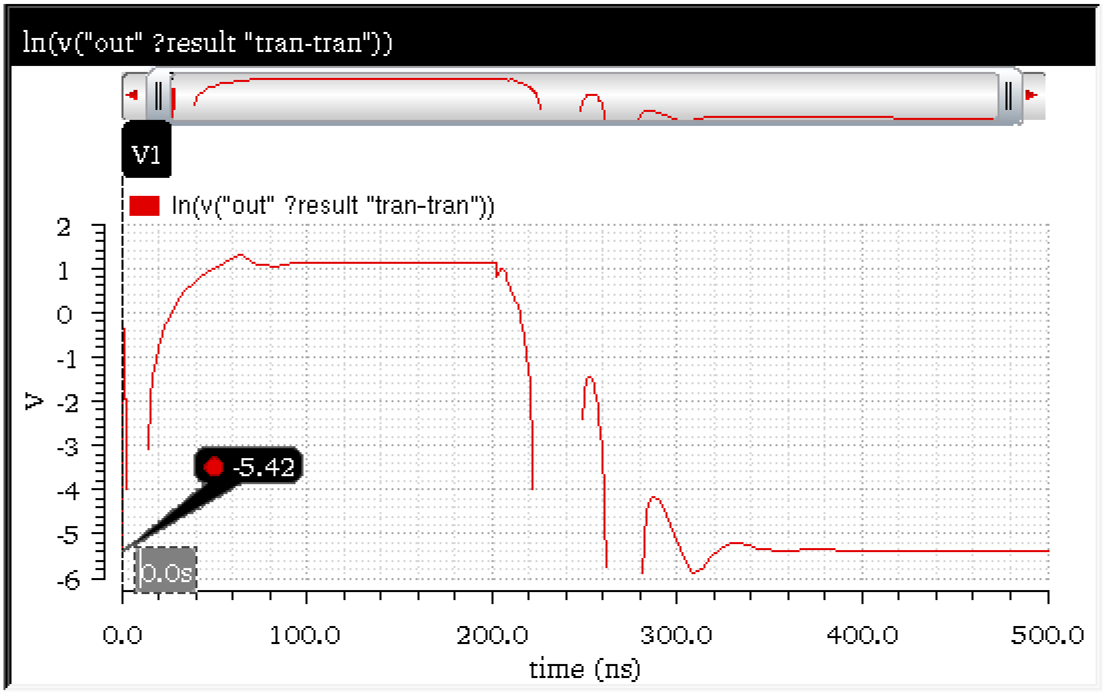
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `ln` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
ln(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

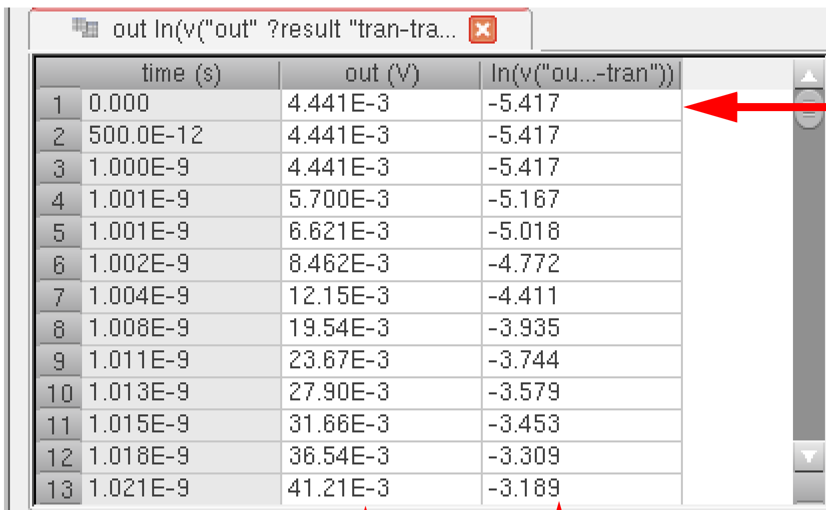
When you evaluate this expression, the exponential value of the input signal is calculated and the following output waveform is plotted in a new graph window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`-5.42mV` when placed at time=`0.0ns`, which is the natural logarithmic value of the voltage value (`4.441E-3mV`) shown in the input waveform.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out ln(v("out" ?result "tran-tra...

	time (s)	out (V)	ln(v("ou...-tran"))
1	0.000	4.441E-3	-5.417
2	500.0E-12	4.441E-3	-5.417
3	1.000E-9	4.441E-3	-5.417
4	1.001E-9	5.700E-3	-5.167
5	1.001E-9	6.621E-3	-5.018
6	1.002E-9	8.462E-3	-4.772
7	1.004E-9	12.15E-3	-4.411
8	1.008E-9	19.54E-3	-3.935
9	1.011E-9	23.67E-3	-3.744
10	1.013E-9	27.90E-3	-3.579
11	1.015E-9	31.66E-3	-3.453
12	1.018E-9	36.54E-3	-3.309
13	1.021E-9	41.21E-3	-3.189

At time=0.00ns, the voltage value in the input signal, out(V), is 4.441E-3. After the ln function is applied, the voltage value at time=0.00ns in output signal is shown as -5.417. (Refer to see vertical markers in the above figures for input and output waveforms)

Input waveform values

Output waveform values

Related OCEAN Function

The equivalent OCEAN command for `ln` is:

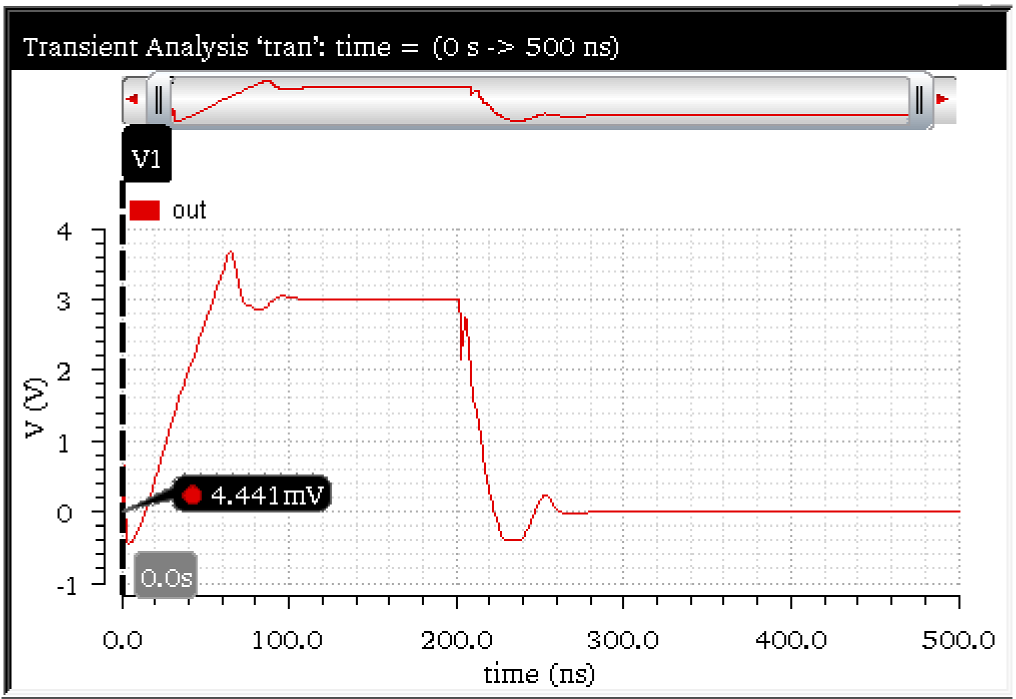
For more information, see `ln` in *OCEAN Reference*.

log10

Returns the base 10 logarithm of a signal.

Example

Consider the following input signal from transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0ns` and voltage=`4.441mv`.

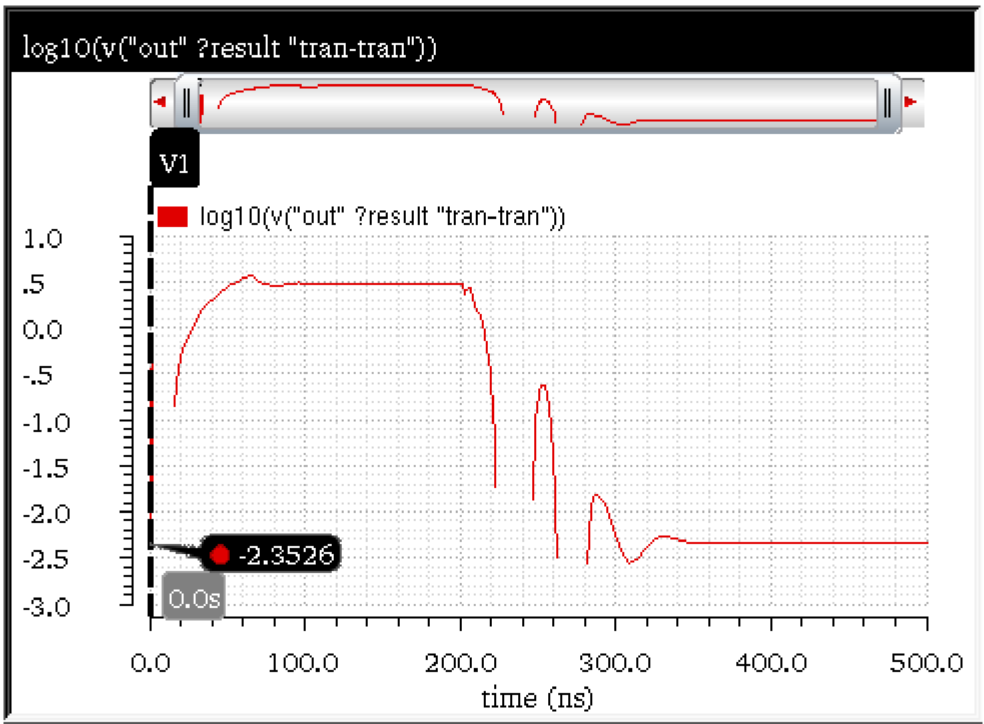
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `log10` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
log10(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

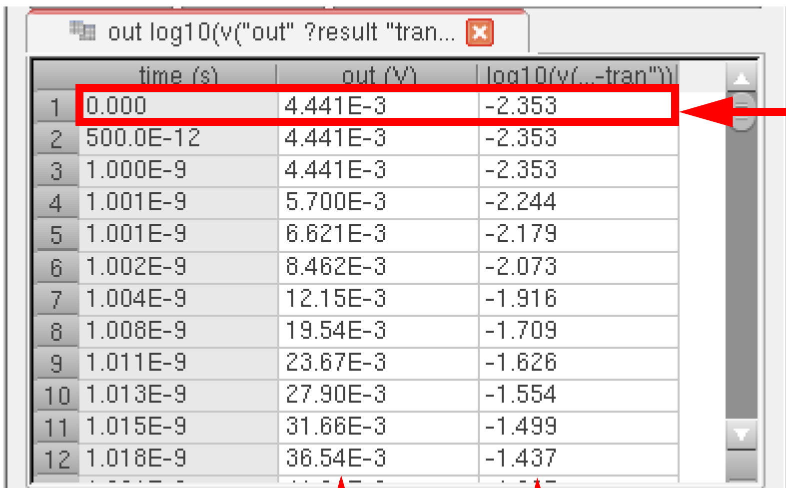
When you evaluate this expression, the log10 value of the input waveform is calculated and the following output signal is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`-2.3526mV` when placed at time=`0.0ns`, which is the logarithmic to the base 10 value of the voltage value (`4.441E-3mV`) shown in the input waveform.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	log10(v(...-tran"))
1	0.000	4.441E-3	-2.353
2	500.0E-12	4.441E-3	-2.353
3	1.000E-9	4.441E-3	-2.353
4	1.001E-9	5.700E-3	-2.244
5	1.001E-9	6.621E-3	-2.179
6	1.002E-9	8.462E-3	-2.073
7	1.004E-9	12.15E-3	-1.916
8	1.008E-9	19.54E-3	-1.709
9	1.011E-9	23.67E-3	-1.626
10	1.013E-9	27.90E-3	-1.554
11	1.015E-9	31.66E-3	-1.499
12	1.018E-9	36.54E-3	-1.437

At time=0.00ns, the voltage value in the input signal, out(V), is 4.441E-3. After the log10 function is applied, the voltage value at time=0.00ns in output signal is shown as -2.353. (Refer to vertical markers in the above figures for input and output waveforms)

Input Waveform Values

Output Waveform Values

Related OCEAN Function

The equivalent OCEAN command for `log10` is:

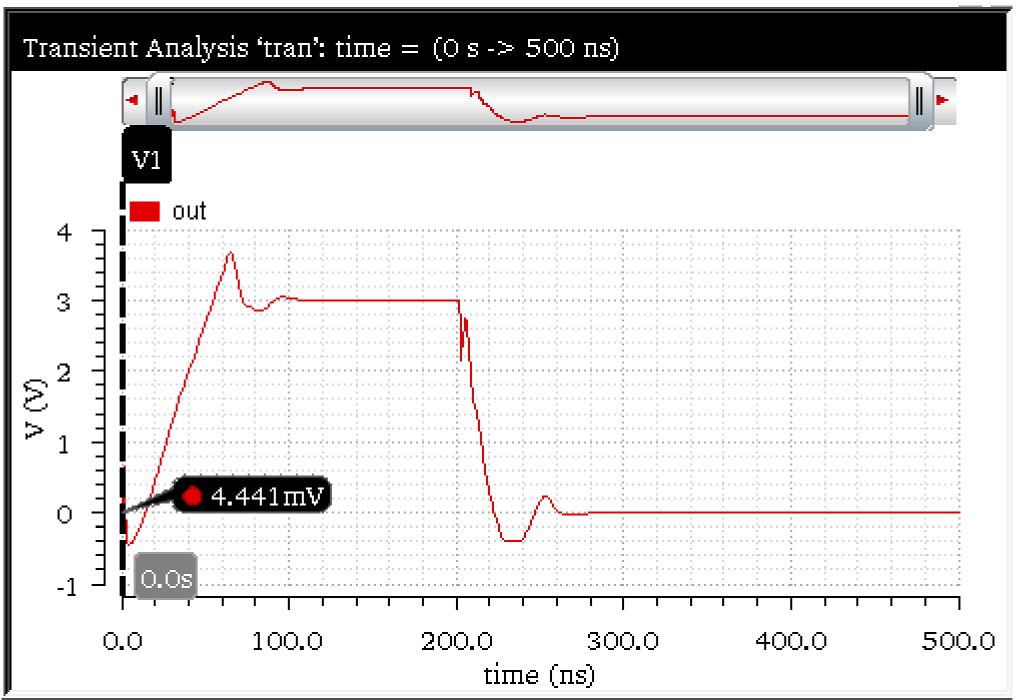
For more information, see `log10` in *OCEAN Reference*.

sqrt

Returns the square root of the given input waveform.

Example

Consider the following input signal from transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0ns` and voltage=`4.441mv`.

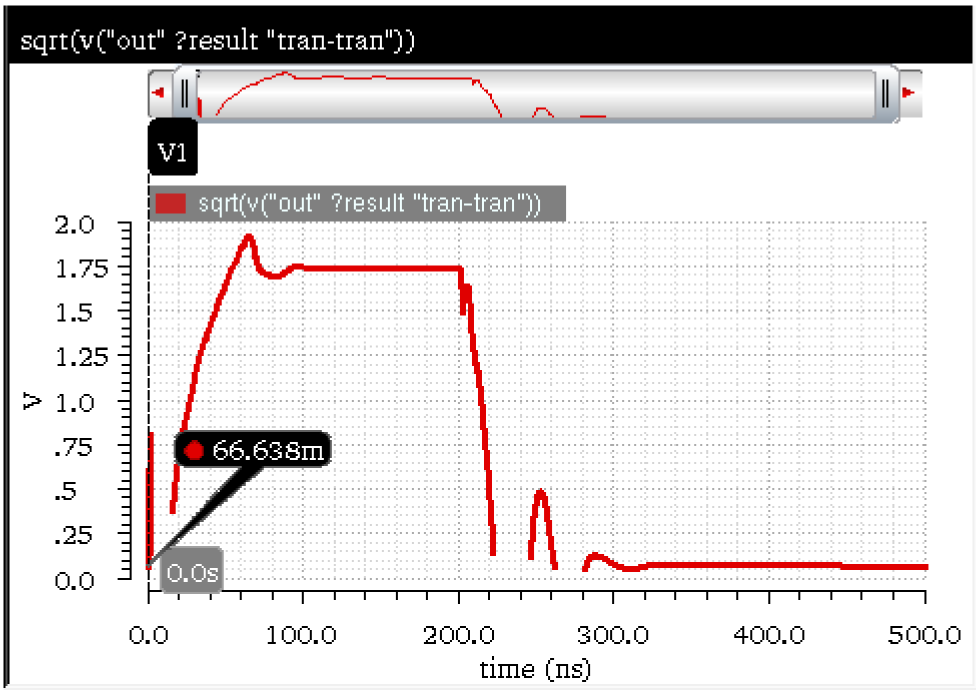
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `sqrt` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
sqrt(v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw"))
```

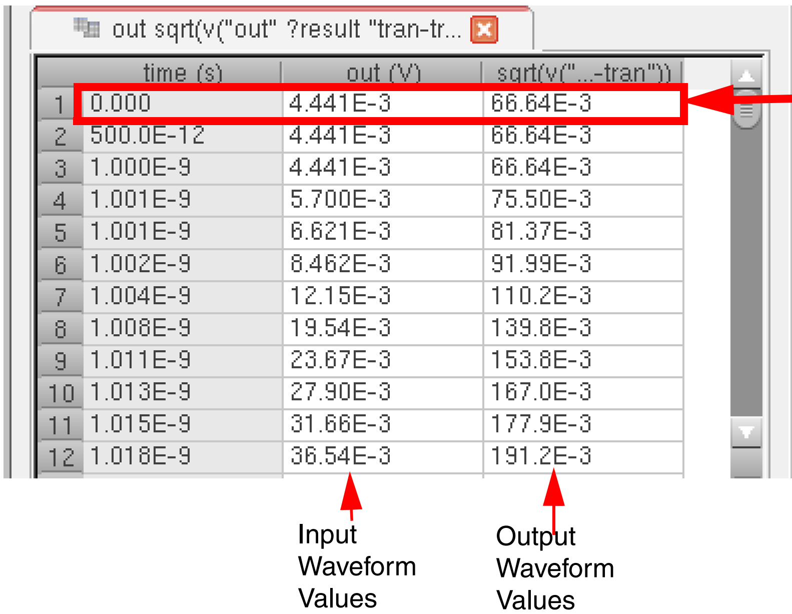
When you evaluate this expression, the square root of the input waveform is calculated and the following output signal is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`-2.3526mV` when placed at time=`0.0ns`, which is the square root of the voltage value (`4.441E-3mV`) shown in the input waveform.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	sqrt(v("...-tran"))
1	0.000	4.441E-3	66.64E-3
2	500.0E-12	4.441E-3	66.64E-3
3	1.000E-9	4.441E-3	66.64E-3
4	1.001E-9	5.700E-3	75.50E-3
5	1.001E-9	6.621E-3	81.37E-3
6	1.002E-9	8.462E-3	91.99E-3
7	1.004E-9	12.15E-3	110.2E-3
8	1.008E-9	19.54E-3	139.8E-3
9	1.011E-9	23.67E-3	153.8E-3
10	1.013E-9	27.90E-3	167.0E-3
11	1.015E-9	31.66E-3	177.9E-3
12	1.018E-9	36.54E-3	191.2E-3

At time=0.00ns, the voltage value in the input signal, out(V), is 4.441E-3. After the sqrt function is applied, the voltage value at time=0.00ns in output signal is shown as 66.64E-3. (Refer to vertical markers in the above figures for input and output waveforms)

Related OCEAN Function

The equivalent OCEAN command for `sqrt` is:

```
sqrt( n_number )
     => f_result
```

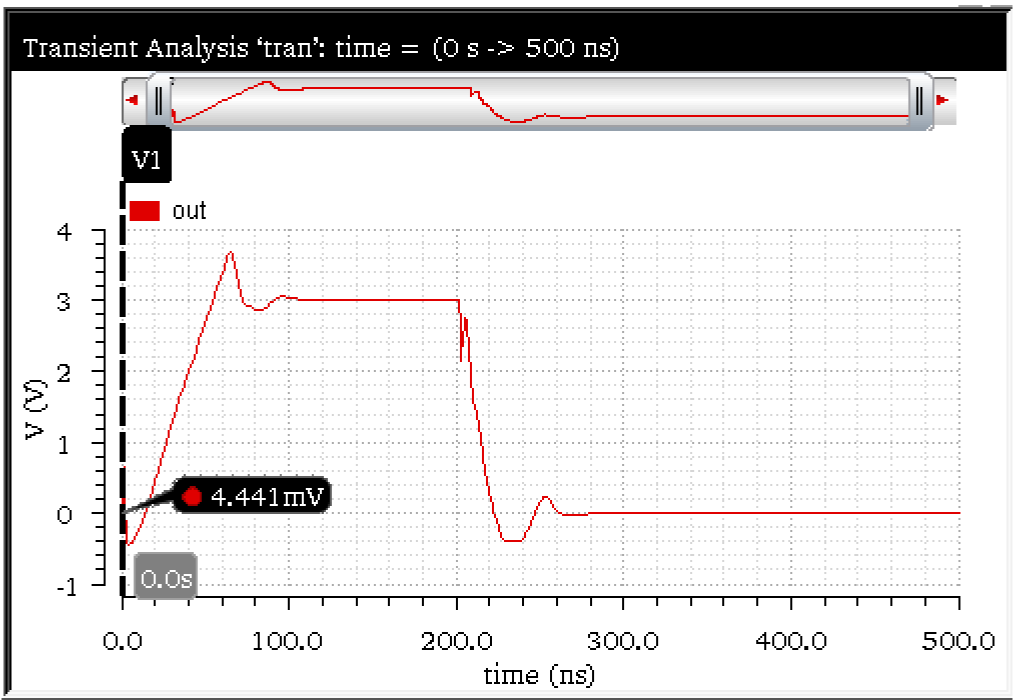
For more information, see `sqrt` in *OCEAN Reference*.

x**2

Returns the x^2 (second power) value of the input signal. This function is available only in the SKILL mode.

Example

Consider the following input signal from transient analysis:



Note that a vertical marker `V1` is placed at time=`0.0ns` and voltage=`4.441mv`.

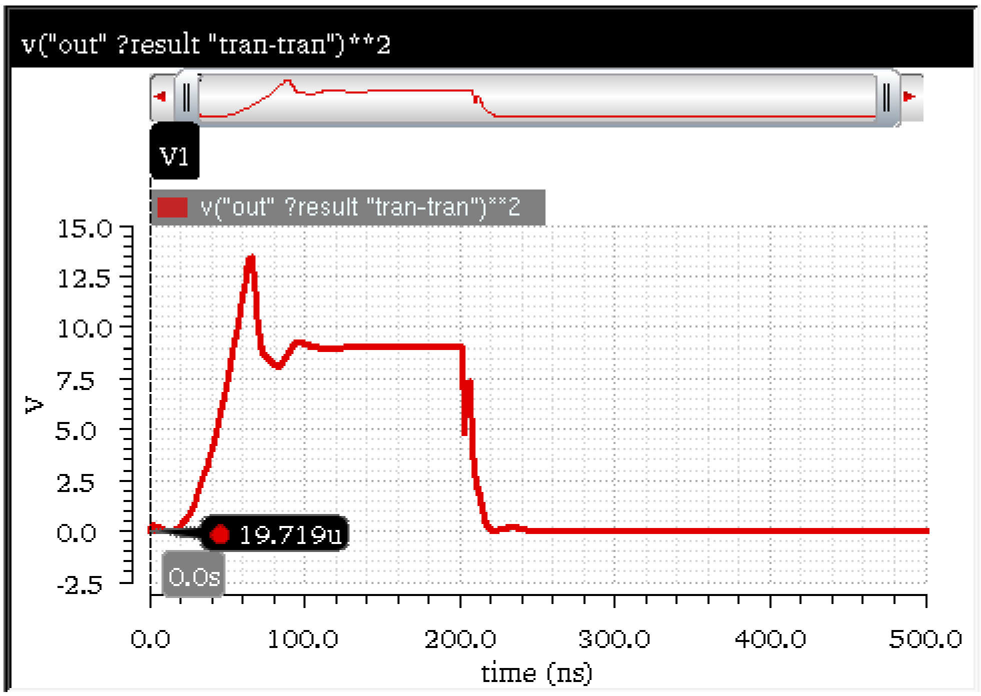
When you send this signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `x**2` function on this input signal, the function is directly applied on the signal expression in Buffer. The expression created in Buffer is as follows:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")**2
```

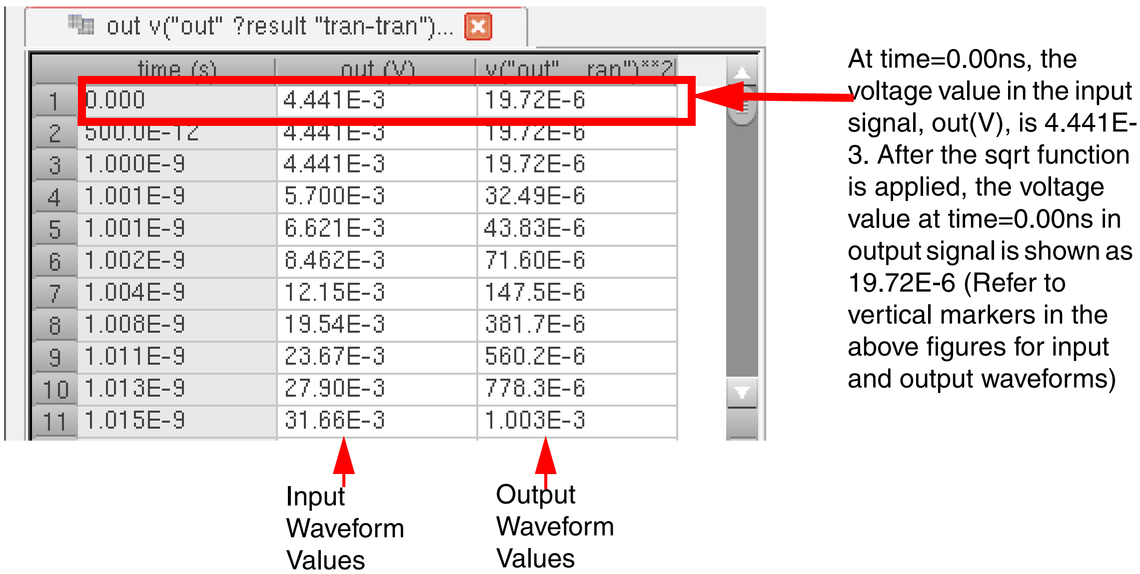
When you evaluate this expression, the second power of the input waveform is calculated and the following output signal is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`19.71mV` when placed at time=`0.0ns`, which is the square root of the voltage value (`4.441E-3mV`) shown in the input waveform.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



	time (s)	out (V)	v("out" ran")**2
1	0.000	4.441E-3	19.72E-6
2	500.0E-12	4.441E-3	19.72E-6
3	1.000E-9	4.441E-3	19.72E-6
4	1.001E-9	5.700E-3	32.49E-6
5	1.001E-9	6.621E-3	43.83E-6
6	1.002E-9	8.462E-3	71.60E-6
7	1.004E-9	12.15E-3	147.5E-6
8	1.008E-9	19.54E-3	381.7E-6
9	1.011E-9	23.67E-3	560.2E-6
10	1.013E-9	27.90E-3	778.3E-6
11	1.015E-9	31.66E-3	1.003E-3

Related OCEAN Function

The equivalent OCEAN command for `x**2` is:

For more information, see `x**2` in *OCEAN Reference*.

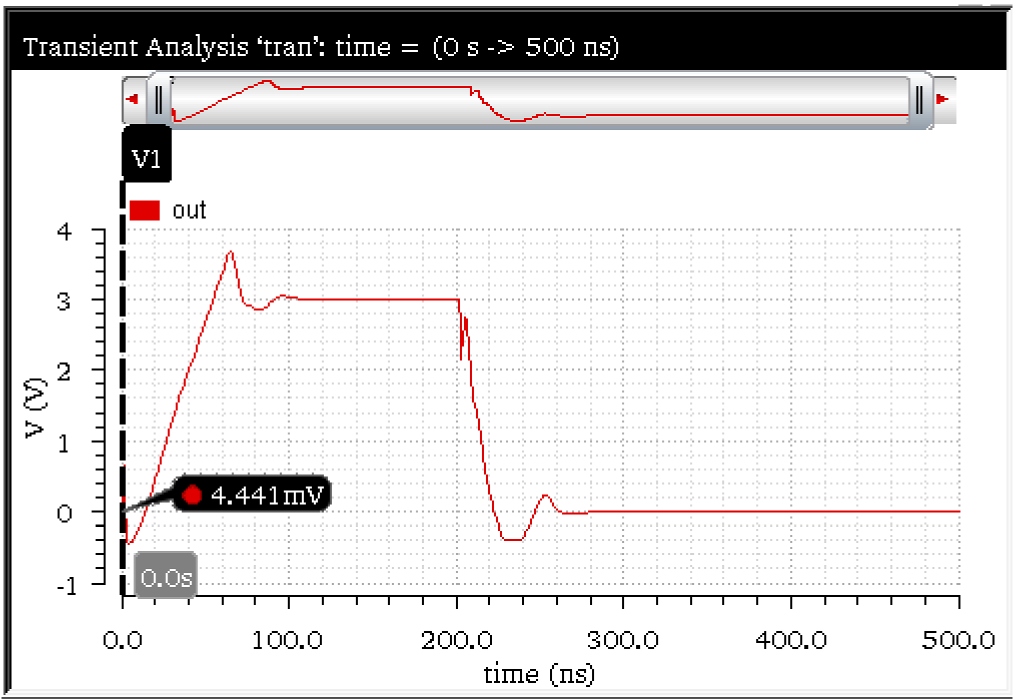
y**x

Returns the y^x (y to the power x) value. This function is available only in the SKILL mode.

Example

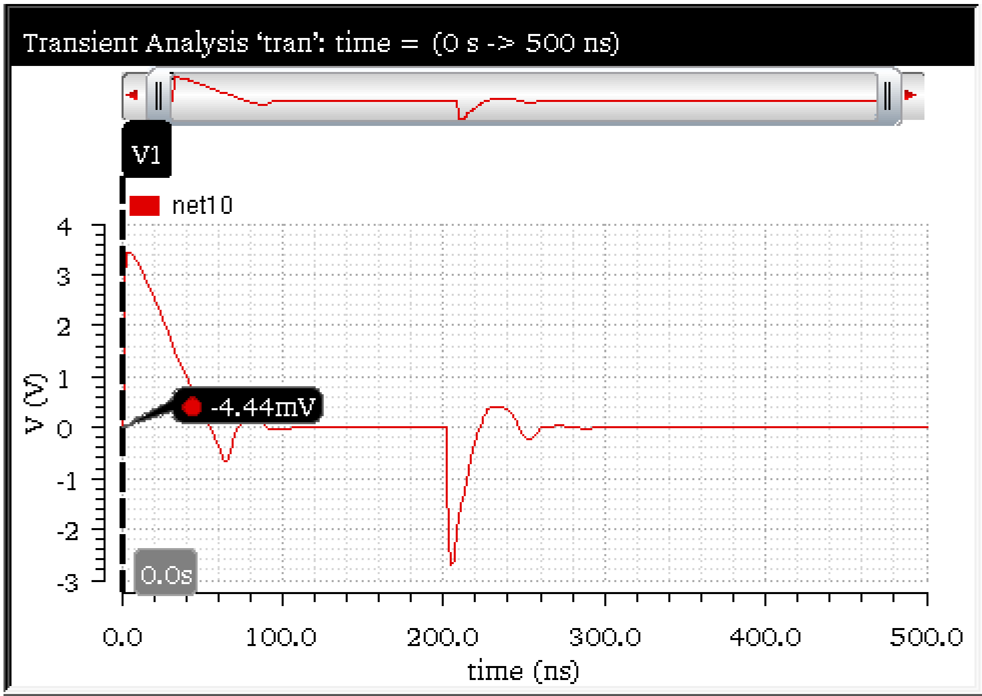
Consider the following two input signals from transient analysis. The first signal `v("out" ?result "tran-tran")` denotes the y value whose power is to be calculated and second signal `v("net10" ?result "tran-tran")` denotes the power (x value in y^x).

First Input Signal: `v("out" ?result "tran-tran")`



Note that a vertical marker `V1` is placed on the first signal at time=`0.0ns` and voltage=`4.441mv`.

Second Input Signal: `v("net10" ?result "tran-tran")`



Note that a vertical marker `V1` is placed on the second input signal at time=`0.0ns` and voltage=`-4.44mv`.

When you send the first input signal to Calculator, the following expression is created in the Buffer:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

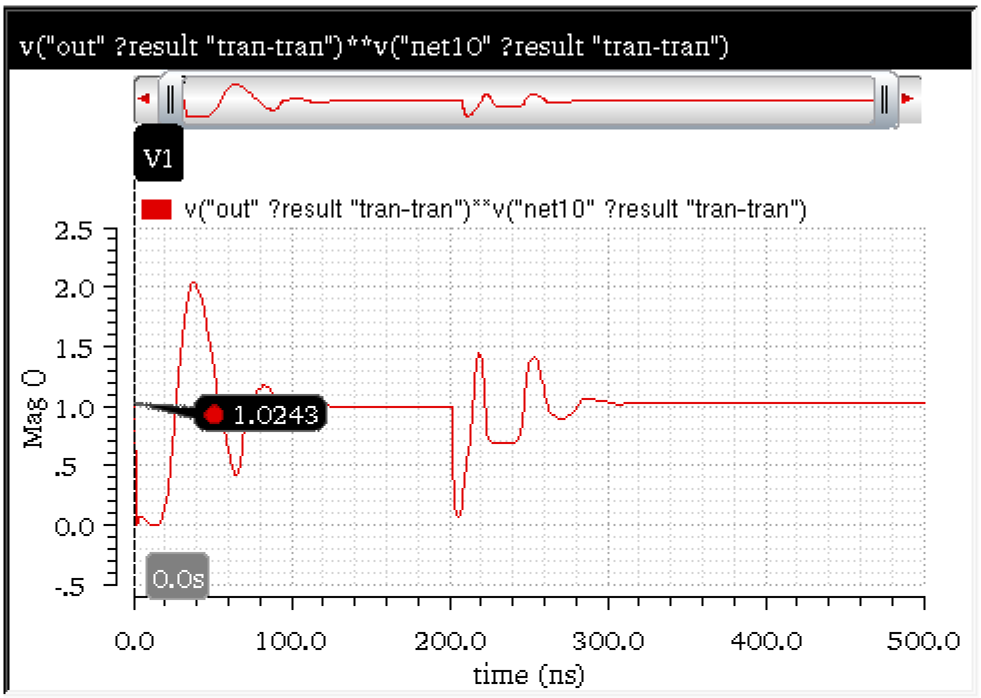
To add second signal to the Calculator, you need to send the expression for the first input signal in the Stack. Now, when you send the second input signal to Calculator, the following expression is created in the Buffer:

```
v("net10" ?result "tran-tran" ?resultsDir "./ampsim.raw")
```

When you apply the `y**x` function on this input signal, the function pops out the first signal expression from the Stack and uses the second signal expression from the Buffer. The new expression created in Buffer is as follows:

```
v("out" ?result "tran-tran" ?resultsDir "./ampsim.raw")**v("net10"
?result "tran-tran" ?resultsDir "./ampsim.raw")
```

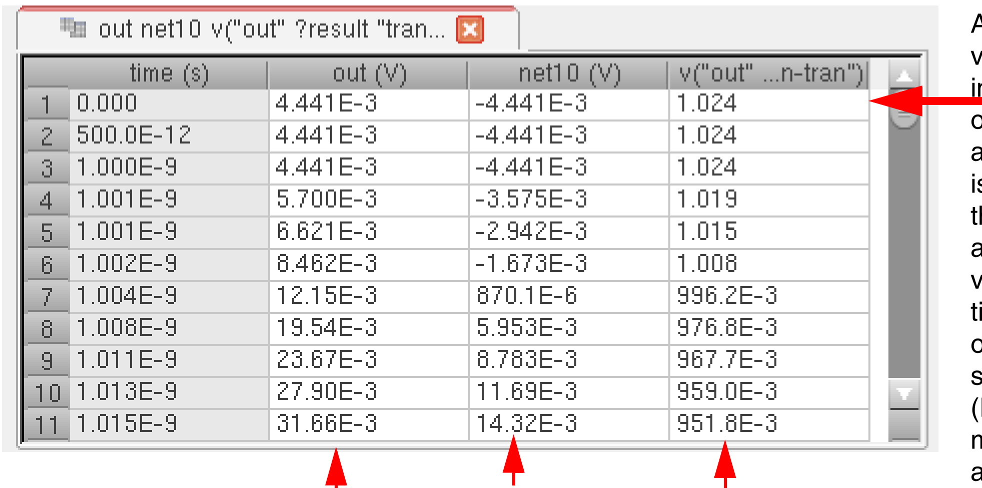
When you evaluate this expression, the second power of the input waveform is calculated and the following output signal is displayed in a new window:



In the output waveform, note that the vertical maker `V1` shows the voltage=`1.0243mV` when placed at time=`0.0ns`, which is the (`4.441E-3`)th power of the voltage value `-4.44E-3` as marked by vertical markers in the input waveforms.

To analyze the generated output values, you can send the input and output signals to the Virtuoso Visualization and Analysis XL Table. To compare the input and output values, it is required to display both the signals in the same table, side by side. Therefore, to send the output signal to the table, right-click the output signal and choose *Send To – Table – Append*. The output signal is appended to the existing table.

The following table contents are displayed when you send the input and output signals described in this example to the table:



out net10 v("out" ?result "tran...

	time (s)	out (V)	net10 (V)	v("out" ...n-tran")
1	0.000	4.441E-3	-4.441E-3	1.024
2	500.0E-12	4.441E-3	-4.441E-3	1.024
3	1.000E-9	4.441E-3	-4.441E-3	1.024
4	1.001E-9	5.700E-3	-3.575E-3	1.019
5	1.001E-9	6.621E-3	-2.942E-3	1.015
6	1.002E-9	8.462E-3	-1.673E-3	1.008
7	1.004E-9	12.15E-3	870.1E-6	996.2E-3
8	1.008E-9	19.54E-3	5.953E-3	976.8E-3
9	1.011E-9	23.67E-3	8.783E-3	967.7E-3
10	1.013E-9	27.90E-3	11.69E-3	959.0E-3
11	1.015E-9	31.66E-3	14.32E-3	951.8E-3

First Input Waveform

Second Input Waveform

Output Waveform

At time=0.00ns, the voltage value in the input signal1(x), out(V), is 4.441E-3 and input signal2 (y) is -4.441E-3. After the y**x function is applied, the resulted voltage value at time=0.00ns in output signal is shown as 1.024 (Refer to vertical markers in the above figures for input and output waveforms)

Related OCEAN Function

The equivalent OCEAN command for `y**x` is:

For more information, see `y**x` in *OCEAN Reference*.

C

Constants

This chapter lists constants and their definitions for the SKILL and MDL modes.

Table C-1 Constants in the SKILL Mode

Constant	Definition
Boltzmann	1.380622e-23
charge	1.6021917e-19
degPerRad	57.2957795130823
epp0	8.854e-12
pi	3.14159265358979323846
sqrt2	1.41421356237309504880
twoPi	6.28318530717958647688

Table C-2 Constants in the MDL Mode

Integer Constants		
yes	Boolean true	1
no	Boolean false	0
Real Mathematical Constants		
pi	π	3.14159265
e	e	2.71828183
inf	∞	infinity

nan	Not a number (result of an invalid operation)	NaN

Real Physical Constants

q	Charge of an electron	$1.6021918 \cdot 10^{-19}$ C
c	Speed of light	$2.99792458 \cdot 10^{8}$ m/s
k	Boltzmann's constant	$1.3806226 \cdot 10^{-23}$ J/K
h	Planck's constant	$6.6260755 \cdot 10^{-34}$ J-s
eps0	Permittivity of a vacuum	$8.854187923944200013968 \cdot 10^{-12}$ F/m
epsrsi	Relative permittivity of silicon	11.7
u0	Permeability of a vacuum	$\pi \times 4.0 \cdot 10^{-7}$ H/m
celsius0	0 celsius	273.15 K
micron		10^{-6} m
angstrom		10^{-10} m
avogadro	Avogadro's number	$6.022169 \cdot 10^{23}$
logic0	The value of logic 0	0
logic1	The value of logic 1	5

D

Defining New SKILL Functions

You can define a function and add it to the *SKILL User Defined Functions* category in the calculator by following these steps:

1. Define the form that prompts for user-defined arguments to the function.
2. Define the syntax of the function in the callback procedure.
3. Register the function.

Defining a Form

The following example shows how to define an input form for a function that takes three arguments. The first argument is the buffer expression. The other two arguments are the boundaries of the range of the expression on which you want to operate.

```
procedure( CreateMyForm()
   let( ( fieldList a b )
    a = ahiCreateStringField(
        ?name 'from
        ?prompt "From"
        ?value ""
    )
    b = ahiCreateStringField(
        ?name 'to
        ?prompt "To"
        ?value ""
    )
    fieldList = list(
        list( a 5:0 120:25 40 )
        list( b 160:0 110:25 30 )
    )
    calCreateSpecialFunctionsForm( 'MyForm
        fieldList )))
```

In this example, the From and To fields are string fields created in a two-dimensional form specification for fieldList. The form is created by the call to `calCreateSpecialFunctionsForm`. This function creates and registers the form with the specified form symbol, MyForm.

Defining a Callback Procedure

You define a callback procedure that is called from the entry on the Calculator User Defined Functions category. Since this example uses a form to prompt for additional information required by the special function, the callback procedure is

```
procedure( MySpecialFunctionCB()
  calCreateSpecialFunction(
        ?formSym 'MyForm
        ?formInitProc 'CreateMyForm
        ?formTitle "Test"
        ?formCallback "calSpecialFunctionInput( 'test
        '(from to) )"
  )
)
```

In this procedure, a call is made to `calCreateSpecialFunction`, which creates and displays the form and then builds the expression in the buffer with the specified form fields.

Using Stack Registers in the Procedure

You can use the special symbol `'STACK` in the list of form fields to get expressions from the stack.

For example, if you want to insert a stack element between the From and To arguments in the special function expression, you can specify the callback line as follows:

```
?formCallback "calSpecialFunctionInput('test '(from STACK to))"
```

If your special function does not require a form to prompt for additional arguments, you can define your callback as follows:

```
procedure( MySpecialFunctionCB()
    calSpecialFunctionInput( 'test nil )
)
```

Registering the Function

You register the function and callback with the `calRegisterSpecialFunction`:

```
calRegisterSpecialFunction(
    list( "test" 'MySpecialFunctionCB )
)
```

The next time you open the calculator, the functions you defined appear in the User Defined Functions category.

Defining a Custom Function

Custom functions need to be supported for both single and multi-dimensional waveform (parametric) data.

A custom function example is shown below:

```
procedure(myFunction(wf)
  cond(
    (drIsWaveform(wf)
      ;; your custom function code
    )
    (famIsFamily(wf)
      famMap('myFunction fam)
    )
    (t
      error("myFunction cannot handle argument %L\n" wf)
    )
)
```

SKILL User Interface Functions for the Calculator

For information on SKILL Functions for the calculator, refer to chapter 22 of the *Virtuoso Analog Design Environment SKILL Language Reference*.

E

Working With Function Templates

This chapter describes the function templates and how you can use them to create new functions in Virtuoso Visualization and Analysis XL Calculator with supporting examples.

Function Templates

Function templates are a mechanism to facilitate easier construction and addition of new functions in the Calculator. They are described in a prescribed format in function template file.

Virtuoso Visualization and Analysis XL uses these template files to perform the following tasks:

- Builds lists of function names, separated into categories and displays it in the GUI panel.
- Generates an expression using rules present in the template file.

Function Template Search Paths

Virtuoso Visualization and Analysis XL searches for function templates using csfsearchpath. The priority of searching a UDF GUI template will be:

1. `<CSF_SEARCH_PATHS>/measures`
2. `<CDS_INST_DIR>/tools/dfII/local/tools/wavescan/measures`
3. `<CDS_INST_DIR>/tools/dfII/etc/tools/wavescan/measures`

Virtuoso Visualization and Analysis XL function templates, shipped by Cadence, are stored at the following location:

`<CDS_INST_DIR>/tools/dfII/etc/tools/wavescan/measures`

Template Catalog Summary File

The template catalog summary file `.skeMeaseruesCatalog` lists the categories and the templates that exists in each category. Virtuoso Visualization and Analysis XL template catalog file is shipped in the following directory.

`<CDS_ROOT>/tools/dfII/etc/tools/wavescan/measures/.skeMeasuresCatalog`

Different categories are specified in-between the lines `skeBeginCategory` and `skeEndCategory`. The below example defines Math, Modifier, RF Functions, Special Function, and Trigonometric as the categories.

```
skeBeginCategory
Math
Modifier
RF Functions
Special Functions
```

```
Trigonometric
skeEndCategory
```

You can specify different functions within a category in between the lines:

```
skeBeginMeasures <Category Name> & skeEndCategory.
```

For example, function “Fourier Evaluation” of category “Special Functions can be defined as follows:

```
skeBeginMeasures.Special Functions
fourEval;"Fourier Evaluation"
skeEndMeasures
```

Creating a template file

To add a new function, write a template file for your function and override `.skeMeasuresCatalog` file.

To create a template file, perform the following steps:

1. Choose one of the two directories listed below where you want to store the new functions:
   - `<CSF_SEARCH_PATHS>/measures`
   - `<CDS_INST_DIR>/tools/dfII/local/tools/wavescan/measures`
2. After creating a directory named "`measures`" which is accessible to the complete project team, copy `<CDS_ROOT>/tools/dfII/etc/tools/wavescan/measures/.skeMeasuresCatalog` to `CSF_SEARCH_PATHS/measures/.skeMeasuresCatalog`.
3. Open `CSF_SEARCH_PATHS/measures/.skeMeasuresCatalog` for editing.
4. Add an entry of the new function in between `skeBeginMeasures &` `skeEndMeasures` tags of the category where the new function is required to belong.

   For example to add a function named 'testfun' in math category, add testfun; in between `skeBeginMeasures Math` & nearest `skeEndMeasures` line.

   Note: Current implementation doesn't merge the entries specified in separate `.skeMeasuresCatalog` files.

Working with Template File

Each Virtuoso Visualization and Analysis XL function template file describes a Calculator function. The template provides information such as:

- The function name
- The function's categories
- Input parameters and default values
- Gui building tips such as adjacent row hints and parameter dependencies
- Tool tip information (currently ignored).
- Rules to build the function expression

The template is divided into following sections:

Header

Header describes the general function information such as the name, display name, description, and category list.

- **function name:** This is the name used to construct the expression
- **name:** This is the display name used as the dialog label in the function panel.
- **Category list:** Each template can belong to multiple categories. For example, the riseTime template could belong to "Special Functions" and "transient". Currently we do not define categories according to analysis type. The current categories are:
    - Math
    - Modifier
    - RF Functions
    - Special Functions
    - Trigonometric

The Analysis Section

The analysis section describes how to generate the expression, the signal, and parameter arguments.

- **args:** Describe how to build the expression
- **signals:** Describe each signal argument:
    - prompt
    - tool tip
    - params. describe each parameter
    - prompt
    - tool tip
    - type
    - default value
    - required

Examples

Example 1: Sample template with single argument: average.ocn

Location:

<CDS_INST_DIR>/tools/dfII/etc/tools/wavescan/measures/average.ocn

```
 1;;; GUI builder information ;;;;;;;;;;;;;;;;;;;;;;;;;;;
 2 ocnmRegGUIBuilder(
 3   '(nil
 4     function average
 5     name average
 6     description "Compute average of a waveform over its entire range."
 7     category ("Special Functions")
 8     analysis (nil
 9               general (nil
10                    args (signal)
11                    signals (nil
12                              signal (nil
13                                      prompt "Signal"
14                                      tooltip "signal to average"
15                                      )
16                    )
17                    inputrange t
18                    )
19     )
20     outputs (result)
21    )
22 )
```

~

- The function name used to build the expression is defined by line 4 (function average).
- Line 5 is the name used in the dialog label if a function panel is displayed. In this case the function signature takes a single argument (signal) and no function panel is required.

- Line 6 is a simple description that will be used at some point for bubble help. (This feature is not yet implemented).
- Line 7 is a set of category names. Typically each function belongs to a single category. Optionally a function can belong to multiple categories.
- Line 8 begins the analysis section.
- Line 10 is the args statement which provides the rule used by Virtuoso Visualization and Analysis XL to generate the function expression. The args statement is a list of ordered signal and parameter names and in our example contains a single item: signal. Therefore the signature looks something like: `average(VT("/net10"))`, it takes a single signal name

  The order in the args statement defines the order the parameter values will be named in the expression that is put into the buffer. This may or may not be the same order that is displayed in the function panel. Some functions require a more complex mechanism to describe the signature. The template mechanism provides a format statement that will be described in a later example.
- Line 11 starts the signal section. A function signature might contain multiple signal parameters (ex: the delay function).
- The single signal description begins at Line 12. Line 13 is the signal prompt used to name the signal parameter in the Function Panel. Line 14 is the signal tool tip. This is currently not used.

Example 2: Sample template with multiple arguments: delay.ocn

Location:

(<CDS_INST_DIR>/tools/dfII/etc/tools/wavescan/measures/delay.ocn)

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;; GUI builder information
;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ocnmRegGUIBuilder(

  '(nil

    function delay

    name delay

    description "delay "

    category ("Special Functions")

    analysis (nil

              tran  (nil
```

```
                    args (signal1 threshold1 edge1 type1 signal2 threshold2 edge2 type2 numberOfOccurences)
                    signals (nil
                            signal1 (nil
                                    prompt "Signal1"
                                    tooltip "signal to measure"
                                    )
                            signal2 (nil
                                    prompt "Signal2"
                                    tooltip "signal to measure"
                                    )
                    )
                    params (nil
                            threshold1 (nil
                                    prompt "Threshold Value 1"
                                    tooltip "Threshold Value 1"
                                    default 2.5
                                    type float
                                    min 0)
…
     periodicity1(nil
     prompt "Periodicity 1”
     tooltip "Periodicity 1"
     default 1
     type float )
 …
                            periodicity2 (nil
                                    prompt "Periodicity 2"
                                    tooltip "Periodicity 2"
                                    default 1
                                    type float
                            )
                            numberOfOccurences (nil
```

```
 72                                      prompt "Number of occurrences"
 73                                      tooltip "Occurrence choice"
 74                                      default single
 75                                   type ((single (0 0 nil nil)) (multiple
 (" %s  %s  t  %s " periodicity1 periodicity2 sweepName)))
 76                            )
 77                           sweepName (nil
 78                                      prompt "Plot/print vs."
79                            tooltip "Independent variable to plot against."
 80                                      required nil
 81                                      default trigger
 82                                      type (trigger target cycle)
 83                                      min 0)
 84                  )
```

The `delay` template is similar to `average.` Some differences:

- Line 11: The args description contains multiple parameters so a Function Panel will be built to describe this function.

  The signals section contains two signals named signal1 and signal2. The prompts (lines 14 and 18) will name the respective fields in the panel.

  This template contains parameters beginning at Line 22.

- Line 77: the parameter sweepName. Line 82 names the type: a simple list of cyclic choices. The default value (line 81) names the default choice to be initially displayed.

- Line 71: a much more complex example of a cyclic type parameter `(numberOfOccurances).`

  The cyclic type (Line 75) is a list of two choices: single and multiple:

  ```
  type ((single (0 0 nil nil)) (multiple (" %s  %s  t  %s " periodicity1
  periodicity2 sweepName)))
  ```

  Each choice has an associated value that describes what must be added to the expression signature.

  If you select “single”, then the value of numberOfOccurances put into the expression string is literally “0 0 nil nil”.

If you select “multiple”, the choice value is described by a formatted string and will look like: “<periodicity1> <periodicty2> t <sweepName>”. If the default values are used, then the choice resolves to: 1 1 t “trigger”.

Example 3: Signature described by a format statement: compression.ocn

Location:

```
(<CDS_INST_DIR>/tools/dfII/etc/tools/wavescan/measures/compress.ocn)
 9     analysis (nil
10              general  (nil
11                     ;; args (signal ...)
12          format ("compression(dB20(harmonic(%s, %s)), ?x %s, ?compress
%s)" signal harmNumber xpoint compressiondb)
```

The compression function is one of those examples where the expression does not conform to a simple rule of function name plus name/value pairs for the parameters. We have to use the format statement (line 12 above).

The format uses %s to substitute in the values of the named parameters (signal, harmNumber, xpoint, and compressiondb).

Example 4: Creating your own template

This example creates a template for calculator function named ‘trap’ that exists in a new category named ‘MyProject’.

Tip

To build a new template, it is advised to pick an existing similar template and modify it accordingly.

1. Define the function signature:
   - ❑ Function name
   - ❑ Signal parameters
   - ❑ Additional parameters

   For example to create a new function named trap with signature:

   ```
   trap(<signal> <from> <to>)
   ```

   The template signal section will contain a single signal named Signal. The parameter section will contain two parameters named From and To, both of type float.

2. Go to the source directory for template files:

   ```
   <CDS_INST_DIR>/tools/dfII/etc/tools/wavescan/measures
   ```

3. Find an existing similar template so that you don’t have to start from scratch. For example, if the new function trap was similar to existing function clip:

   ```
   % cp clip.ocn trap.ocn
   % chmod +w trap.ocn
   ```

4. Define the function information in the header section. Replace the information indicated in boldface below. Redefine the category name as MyProject.

   ```
   ocnmRegGUIBuilder(
   '(nil
   function trap
   name “trap function panel name”
   description "short tool tip description for trap function”
   category ("MyProject")
   ```

5. Define the args (or format) statement so we have a rule for building the expression.

   For the trap example, no change is required in the args statement that comes from the clip template.

```
args (signal From To)
```

6. Define the signal parameter in the signal section.

For the trap example reuse the clip example. Change the tool tip as required.

```
signals (nil
        signal1 (nil
                prompt "Signal"
                tooltip "signal to trap "
                )
```

7. Define the parameters

For the trap example, reuse the parameter definitions from the clip template. Change the tool tip values as required

```
params (nil
        From (nil
                prompt "From"
                tooltip "Trap Start"
                default 0
                type float
                min 0)
         To (nil
                prompt "To"
                tooltip "End Trap Range"
                default 0
                type float
                min 0)
    )
```

8. Install the measures file into the template directory.

Copy trap.ocn to `<USER_HOME>/measures` or `CSF_SEARCH_PATHS/measures`.

For more information, please refer Function Template Search Paths.

9. Add entry in catalog file by doing following tasks:

- ❑ Install your new project catalog file.

```
% cp  <CDS_INST_DIR>/tools/dfII/etc/tools/wavescan/measures/
.skeMeasuresCatalog CSF_SEARCH_PATHS/measures.
```

For more information please refer Template Catalog Summary File.

- Edit your new project catalog file to add the new category and function name:
  - Add category name at line #2 after skeBeginCategory keyword
  - Add new category section before skeBeginMeasures.Math.

  Your `.skeMeasuresCatalog` file will look like as below:

```
skeBeginCategory
MyProject
Math
Modifier
RF Functions
Special Functions
Trigonometric
skeEndCategory
skeBeginMeasures.Project
trap;trap
skeEndMeasures
skeBeginMeasures.Math
exp;exp
dB20;dB20
```

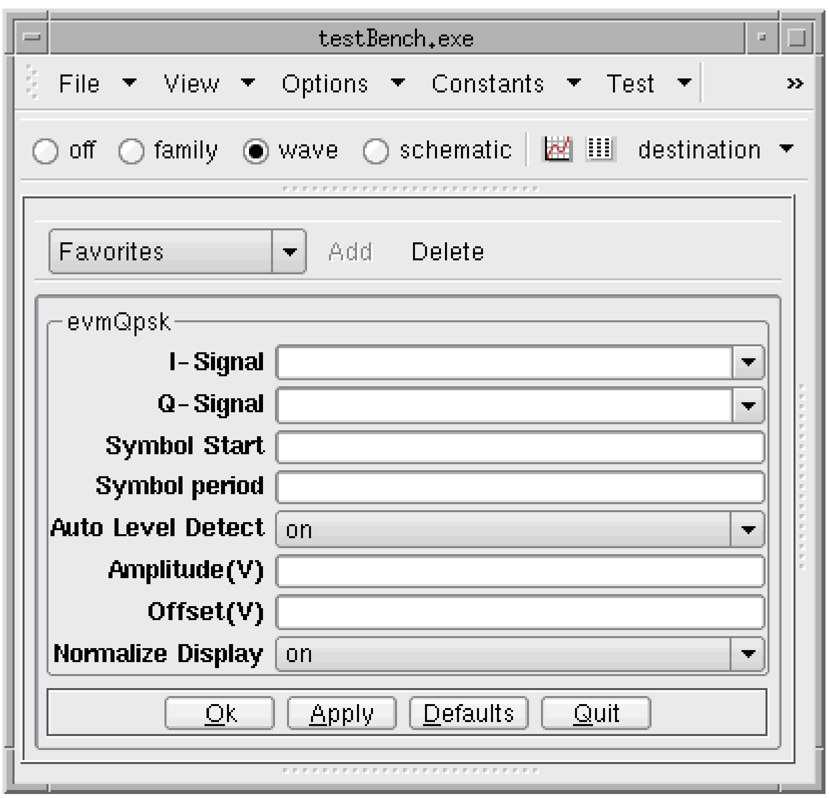
Now run Virtuoso Visualization and Analysis XL and open calculator, you should be able to see trap function added in special function category.

Advanced Features to provide GUI Hints

The template format supports the following advanced features:

- Sometimes it makes sense for two or more parameters to share the same row. An optional parameter property `gridRowHint <num>` directs Virtuoso Visualization and Analysis XL to layout all parameters with the same `gridRowHint value` on the same row.
- Sometimes a parameter controls whether one or more other parameters are enabled or disabled. An optional parameter property `guiEnableHint <target list>` controls whether other parameters are enabled.

Example: evm.ocn template (evmQpsk)



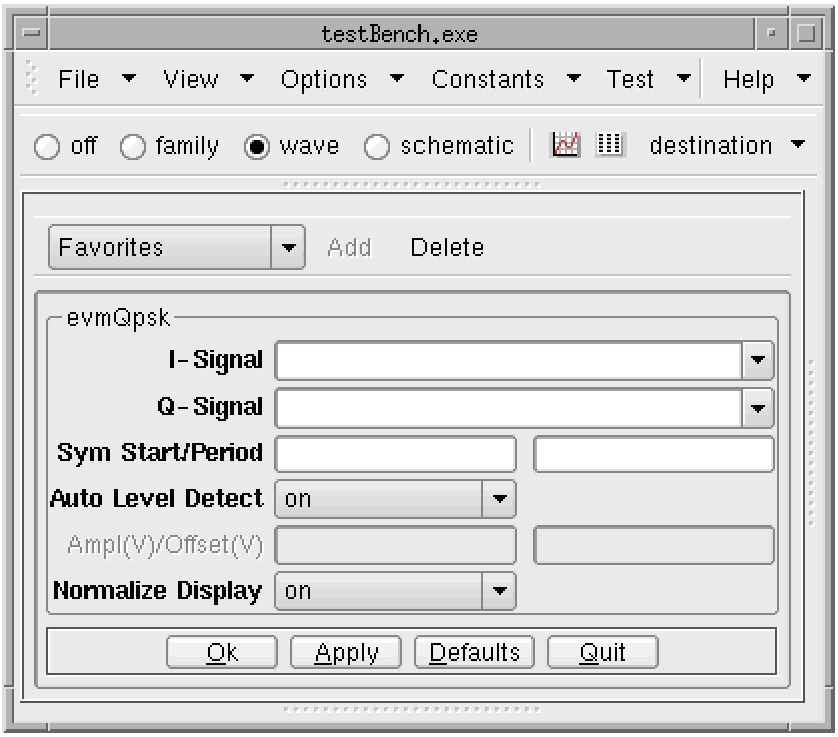
The above example is the GUI form for the original evmQpsk 6.1.0 template. The template contains no `gridRowHints` or `guiEnableHints`.

The next example shows how the two hint properties are used to:

- Place the Symbol Start and Symbol period parameters on the same row.
- Place the Amplitude and Offset parameters on the same row.

- The form has three columns instead of two to make the form more compact.
- The Auto Level Detect parameter will enable/disable the Amplitude and Offset parameters.

Example: evmQpsk with gridRowHint and gridEnableHints:



The following is a snippet of the evmQpsk template with modifications is italized:

```
delay (nil
    prompt "Sym Start/Period"
    tooltip "Symbol Start"
    type float
    required t
    guiRowHint 1
    min 0)
sampling (nil
    prompt ""
    tooltip "Symbol period"
    type float
    required t
    guiRowHint 1
    min 0)
```

Notes on template changes:

- The guiRowHint properties for delay and sampling have the same value (guiRowHint 1), so the prompt/value pairs for both parameters are put on the same row.
- Column 1 contains the prompt for the delay value. Column 3 is supposed to contain the prompt for the sampling value; however, I wanted to simplify the layout and have three columns instead of four. Look at the prompt value for the delay parameter: it actually describes BOTH the delay and sampling value fields. The prompt for the sampling parameter is set to an empty string (“”).

The following is a snippet of the evmQpsk template with modifications is italized:

```
Autoleveldetect(nil
    prompt "Auto Level Detect"
    tooltip "Auto Level Detect"
    type ( ("on" (t, nil, nil)) ("off" ("nil, %s, %s" voltage offset)))
    guiRowHint 2
    guiEnableHint ( (voltage off) (offset off) )
)
```

The autoleveldetect parameter is an enumerated type with values “on” and “off”. When it’s “on”, there is no need to specify two other parameters: voltage and offset because those values will be automatically calculated.

If you want the GUI fields representing these other parameters to be disabled when autoleveldetect is “on”. The guiEnableHint target list gives the name(s) of the other parameters to enable (voltage and offset). Each target is specified as a list: the first name is the target name, followed by the set of autoleveldetect values that will turn that target on.

Examine the first target list for autoleveldetect’s guiEnableHint:

```
guiEnableHint( (voltage off) (offset off))
```

This hint tells the GUI that when the autoleveldetect enumerated value is “off”, the gui for the voltage parameter can be enabled. For any other autoenablehint value, the gui for the voltage parameter will be disabled.

Index

Numerics

A

B

C

D