cādence®

Spectre Interactive Environment User Guide

Product Version IC23.1
June 2023

Contents

Spectre Interactive Environment User Guide

1

Introduction to Spectre Interactive Environment

Spectre Interactive Environment is a debugging environment for Spectre simulations where you can identify the cause of an incorrect output value or an unexpected behavior during a previously run simulation.

If you have an idea about the time point or event, such as a change in the value of a net or terminal, when an issue is occurring, you can use Spectre Interactive Environment to explore the results near that time or event to identify the exact cause. You can further trace back the problem in the design or testbench, and make the required changes to get the desired simulation results.

To facilitate debugging, Spectre Interactive Environment provides various features, such as breakpoints and triggers, that you can use to pause the simulation at any specific point where you need to monitor the value of nets or terminals. You can access circuit instances and connections, and modify their parameters at different events to measure the outputs. These features help you perform a detailed analysis to identify the exact nodes where you can modify the design to improve performance.

In Spectre Interactive Environment, you can perform the following operations before or during a simulation run:

■ Pre-simulation operations

❑ Access and change netlist parameters

❑ Access devices and change their model parameters

❑ Set up probes on element current on nodes

■ Operations while a simulation is running

❑ Iterate over each of the circuit elements and query information about the nodes and instance parameters (can also be performed during pre-simulation).

❑ Iterate over each of the circuit options and query information about the various option parameters (can also be performed during pre-simulation).

❑ Intercept the simulation after major analysis events (such as acceptance of each time point in transient analysis), query the simulator, and then instruct the simulator to stop or continue the simulation from the same point.

❑ Access the values of probes during the simulation.

❑ Register call-back methods or observer objects that can be scheduled to be called/ notified at major analysis events (such as acceptance of each time point in transient analysis).

Video

You can view a video related to this product on Cadence Online support.

In this chapter, you will learn about:

■ Getting Started with Spectre Interactive Environment

■ Introducing the Spectre Interactive Environment User Interface

■ Working with Spectre Interactive Environment

■ Creating Setup for Debugging

■ Configuring Settings in Spectre Interactive Environment

■ Working with States in Spectre Interactive Environment

Tip

To explore the functionality provided by Spectre Interactive Environment, you can use the workshop database and examples available in the Virtuoso installation at following location:

`<install-dir>/tools/dfII/samples/sie/sieWorkshop.tar.gz`

Licensing Requirements

Spectre Interactive Environment requires the **Spectre Interactive for Python and TCL (38530)** license.

If you do not have access to this license, Spectre does not start and exits with an error.

For more information on the license checkout requirements for Spectre Interactive Environment, see _Virtuoso Software Licensing and Configuration Guide_.

Getting Started with Spectre Interactive Environment

Preparing the Setup for Spectre Interactive Environment

Before running Spectre Interactive Environment, ensure that the following are available:

■ Spectre 17.1 ISR4 or a higher version

■ The **Spectre Interactive for Python and TCL (38530)** license

Launching Spectre Interactive Environment

You can launch Spectre Interactive Environment in two ways:

■ From Virtuoso ADE Assembler by using the context-sensitive menu on the *Results* tab

■ From the command prompt by using the `sie` command

Launching Spectre Interactive Environment from ADE Assembler

To start Spectre Interactive Environment from ADE Assembler:

1. In Virtuoso ADE Assembler, run a simulation with transient analysis.

   Note: You cannot open Spectre Interactive Environment from the results of any other analysis.

2. After the simulation is complete, right-click on a result in the *Results* tab and choose *Spectre Interactive Environment*.

   /\ *Important*

   The *Spectre Interactive Environment* command is not available for the results saved by a simulation run using the AMS simulator.

Launching Spectre Interactive Environment from the Command Prompt

To start Spectre Interactive Environment from the command prompt:

➡ Run any of the following commands:

❑ `sie <any-command-line-argument>` to run Spectre Interactive Environment and provide any command-line argument to be sent to Spectre. This is useful when
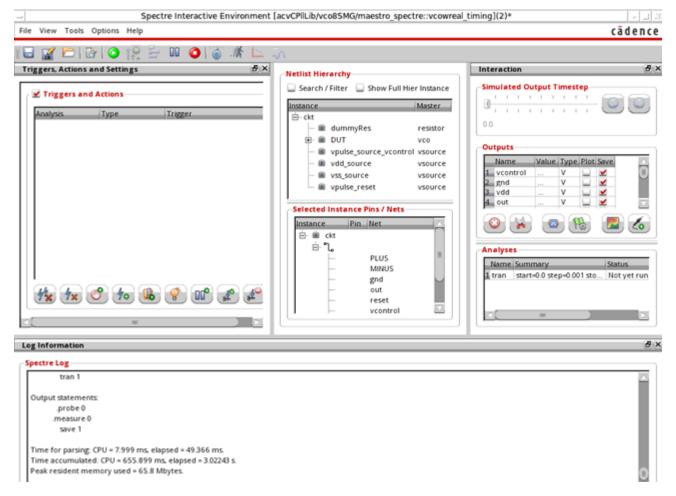
you need to directly send an argument to Spectre that does not have a corresponding command-line argument for the sie command.

❑ `sie -sie_load <sie-setup-file-name>` to run Spectre Interactive Environment and load an existing setup.

Note: For help on the various command-line options, run the `sie -sie_help` command. A list of all command-line options and their use is displayed. You can also refer to <u>Command-Line Options for the `sie` Command</u>.

The Spectre Interactive Environment is displayed, as shown below.

For details about the user interface, refer to <u>Introducing the Spectre Interactive Environment User Interface</u>.

Command-Line Options for the `sie` Command

When launching Spectre Interactive Environment from the command prompt, you can specify any command-line argument for Spectre. In addition, you can specify the additional argument listed below.

Command-Line Option	Description
`-sie_help or -help`	Shows detailed help for the `sie` command.
`-sie_run`	Runs the simulation immediately after starting Spectre Interactive Environment.
`-sie_quiet`	Suppresses the messages that are sent to the terminal
`-sie_plot`	Plots the results immediately after starting Spectre Interactive Environment.
`-sie_signals`	Enables interaction with device signals.
`-sie_params`	Enables interaction with instance parameters.
`-sie_save_restore`	Allows to save or restore the current state of Spectre Interactive Environment.
`-sie_expand_hier`	Expands the full design hierarchy when loaded from Spectre.
`-sie_load <file-name>`	Loads Spectre Interactive Environment and applies the settings saved in the given file. **Note:** This option cannot be used when any other Spectre argument is used.
`-sie_restore <file-name>`	Restores a saved Spectre session back into the current session. This is useful to load a Spectre session that was previously saved while running simulation out of Spectre Interactive Environment.
`-sie_update <milliseconds>`	Opens Spectre Interactive Environment and updates the results after the specified time (in milliseconds). Possible value: Between 10 ms and 1e6 ms Default value: 500ms

The following examples show how you can use the command-line arguments:

```
sie dflip.scs
sie -sie_load t8.sie -sie_run
```
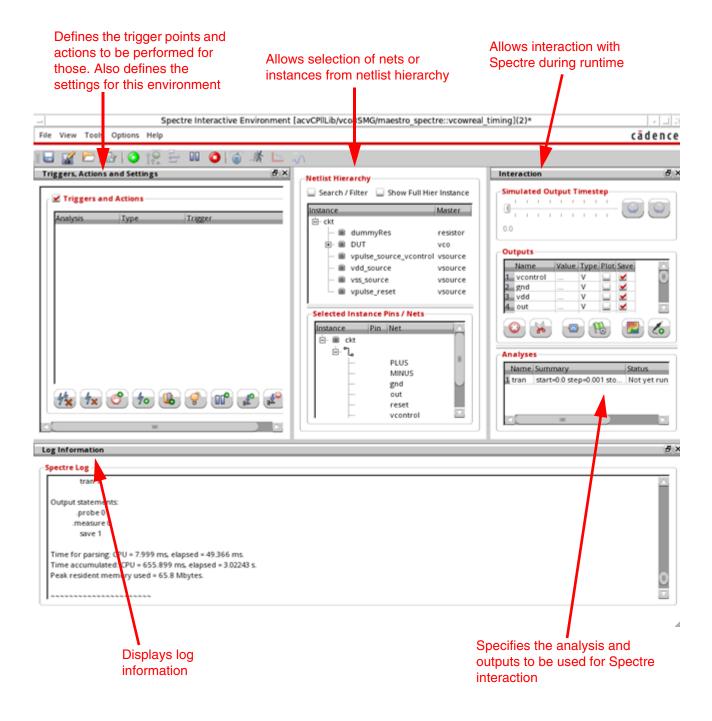
```
sie -sie_load tst1.sie -sie_restore
spectre_saved_state_Tue_Nov_14_2017_11_47_02.srf

sie ball.ckt -sie_quiet -sie_plot -sie_update 1000

sie +aps -64 input.scs
```

Also see: Environment Variables

Introducing the Spectre Interactive Environment User Interface

The following figure describes the various sections in the Spectre Interactive Environment user interface:

Defines the trigger points and actions to be performed for those. Also defines the settings for this environment

Allows selection of nets or instances from netlist hierarchy

Allows interaction with Spectre during runtime

Displays log information

Specifies the analysis and outputs to be used for Spectre interaction

The following sections explain the main components in the Spectre Interactive Environment user interface:
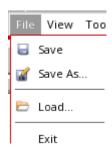
■ Menus

■ Toolbars

Menus

The Spectre Interactive Environment window contains the following menus:
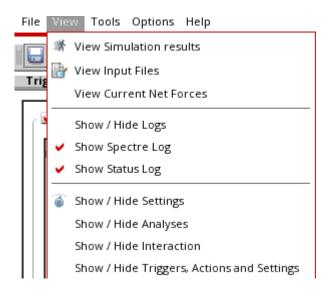
Menu	Description
File	Contains commands to save, load, or exit a session.
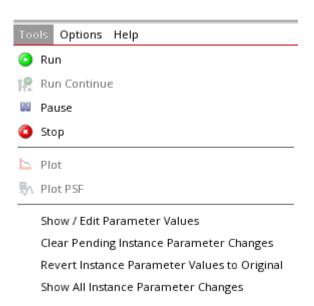
■ *Save*: Saves the settings of the current session in a state file with `.sie` extension

■ *Save As*: Saves the settings of the current session in a different state file that is specified in the sie Save Filename form

■ *Load*: Loads the settings from a saved `.sie` file

■ *Exit*: Closes the Spectre Interactive Environment window

View   Contains commands to view the simulation results and related files, and commands to show or hide different logs and assistants.

- *View Simulation results*: Displays the Simulation Results form for all the simulations run in the current session on Spectre Interactive Environment.

- *View Inputs Files*: Displays the input file to be shared with Spectre for the current setup in Spectre Interactive Environment. It also shows the input files to be passed to Spectre to start the simulation session.

- *View Current Net Forces*: Opens the Forced Nets form in which a list of all the nets and the values forced on them are displayed.

- *Show/Hide Logs*: Controls the display of logs. When disabled, both logs that are saved by Spectre Interactive Environment, or by Spectre, are hidden.

- *Show Spectre Logs*: Controls the display of the simulation run log saved by Spectre.

- *Show Status Logs*: Controls the display of the log saved by Spectre Interactive Environment.

- *Show/Hide Settings*: Displays or hides the *Settings* section in the *Triggers, Action, and Settings* assistant.

- *Show/Hide Analysis*: Displays or hides the *Analyses* assistant.

- *Show/Hide Interaction*: Displays or hides the *Interaction* assistant.

- *Show/Hide Triggers, Actions and Settings*: Displays or hides the *Triggers, Action and Settings* assistant.
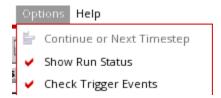
Tools   Contains commands to run simulations and plot graphs. In addition, it also provides commands to view and edit parameter values.

- *Run*: Runs a Spectre simulation by using the current setup.

- *Run Continue*: Continues running the current simulation to the end, to the next the breakpoint, or to the next timestep.

- *Pause*: Pauses the simulation run without waiting for it to complete.

- *Stop*: Stops the Spectre simulation without completing the debug run.

- *Plot*: Plots the results saved by the simulation run from Spectre Interactive Environment.

- *Plot PSF*: Plots the results saved by the simulation run from ADE Assembler

- *Show/Edit Parameter Values*: Displays the Parameter Values form that shows a list of subcircuit parameters with their values in the netlist. If required, you can specify a new value to be used while running the simulations in Spectre Interactive Environment.

- *Clear Pending Instance Parameter Changes*: Clears the pending changes.

- *Revert Instance Parameter Values to Original*: Reverts the pending changes.

- *Show All Instance Parameter Changes*: Prints the changes in the log.

Options   Contains commands to configure options.
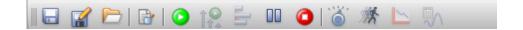
- *Continue or Next Timestep*: Sets the mode for the *Run Continue* command (in the *Tools* menu) to continue to the next breakpoint or to the next timestep.

- *Show Run Status*: Controls the display of the run status of the simulation in progress.

- *Check Trigger Events*: Specifies whether to check that the triggers defined in the Triggers, Events and Actions assistant are saved before running a simulation. If the events are not saved, it stops the simulation run and displays an error.

Help   Contains commands to access help documents and other collateral from Cadence websites.

Toolbars

The Spectre Interactive Environment window contains one toolbar, as shown below.

The following table describes the commands on this toolbar:

Command	Description
	Saves the setup in the current state file.
	Saves the setup in the specified state file.
	Opens the specified state file.
	Shows Spectre input file (input.scs).
	Starts running simulations from the beginning.

	Continues running the current simulation to the end, to the next the breakpoint, or to the next timestep.
	Sets the mode for the *Run Continue* command to continue to the next breakpoint or to the next timestep.
	Pauses a running simulation.
	Stop currently running simulations.
	Shows/hides the *Settings* section in the *Triggers, Action and Settings* assistant.
	Shows the Simulation Results form for all the simulations run in the current session on Spectre Interactive Environment.
	Plots the results saved by the simulation run from Spectre Interactive Environment. These include the plots for the nets selected from the netlist hierarchy.
	Plots the results saved by the simulation run from ADE Assembler. **Note:** This command is available only when Spectre Interactive Environment is run from ADE Assembler.

Working with Spectre Interactive Environment

Working with Spectre Interactive Environment to debug simulation involves the following main tasks:

■ Starting Spectre Interactive Environment with Results of Simulation Run Earlier

■ Creating Setup for Debugging

■ Running Simulations Using Trigger Actions and Breakpoints

■ Configuring Settings in Spectre Interactive Environment

■ Working with States in Spectre Interactive Environment
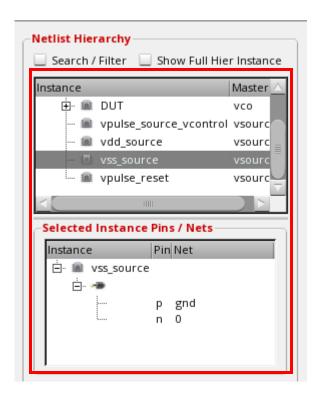
Starting Spectre Interactive Environment with Results of Simulation Run Earlier

Since the objective is to debug the results of a simulation that has already run, you need to begin with loading those results. Depending on how the previous simulation was run, you can load the results in these ways:

■   If the previous simulation was run using Virtuoso ADE Assembler, on the Results tab, select the result of an output of a particular point you want to focus on.

■   If the previous simulation was run using Spectre at the command line, run Spectre Interactive Environment using the sie command and provide the path to the results or a saved state to be loaded as a command-line argument.

Observe the following:

■   Spectre Interactive Environment passes the Spectre command-line arguments to Spectre Interactive to start a new Spectre session. Next, Spectre Interactive Environment queries the Spectre session to access the netlist and other options, and displays the details in different assistants of the Spectre Interactive Environment user interface so that user can interact with it.

■   The tool reads the initial netlist used for that point and loads the instance and pin details from the hierarchy in the *Netlist Hierarchy* assistant.

If the number of instances in the netlist is more than the limit set by the $SIE_INST_LIMIT <num> environment variable, the tool displays a warning message. You can modify the limit to read more instances. However, this would increase the time to load the details.
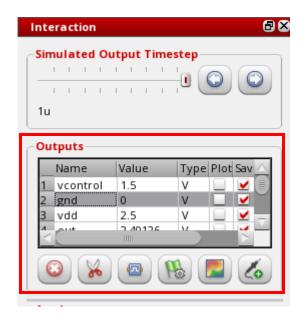
Select the *Show Full Hier Instance* check box to see the full hierarchy path on instances. The table displays an additional column, *Hier Inst*, to show the hierarchical net view.

Tip
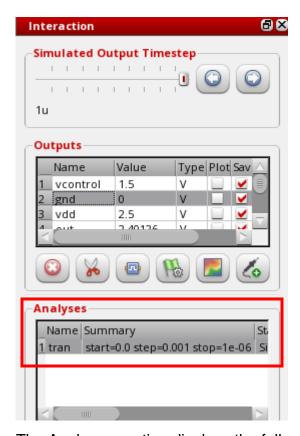
By default, the top level nets are displayed in the *Selected Instance Pins/Nets* section. If you select an instance, the nets and pins of that instance are displayed. To view the top level nets again, select the top level of the netlist in the *Instance* column.

■ The outputs defined in the ADE Assembler setup are displayed in the *Outputs* section of the *Interaction* assistant.

■ The details of the transient analyses extracted from the netlist are also displayed in the *Analyses* section of the *Interaction* assistant.
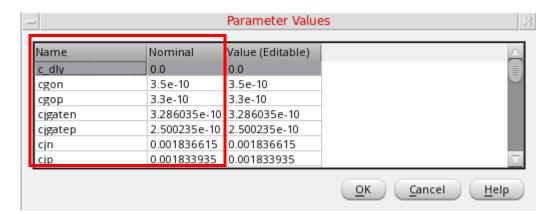
The Analyses section displays the following:

❑ Name of the transient analysis

❑ Summary of transient analyses, which includes start, step and stop time

❑ Status of simulation. Possible values are: `Not yet run`, `Simulation Stopped (breakpoint) @ <time>`, and `Simulation Completed @ <time>`.

Note: If the netlist contains multiple transient analysis, select the row of the analysis for which you want to run the simulation.

■ Choose *Tools – Show/Edit Parameter Values* to view the parameter values saved in the netlist for the selected point. The Parameter Values form is displayed.

The *Name* column shows the names of all parameters saved in the netlist. The *Nominal* column shows the parameter values used in the netlist file.

Note: For all points except the first point, an additional column, *Override* (not shown in the figure), is also displayed on the Parameter Values form. The values of variables in the first point are considered as base values. Any parameter value modified for other points is considered as an overridden value and displayed in the *Override* column. For more details, refer to Editing Subcircuit Parameters.

Run a simulation from Spectre Interactive Environment with the initial setup and view its results.
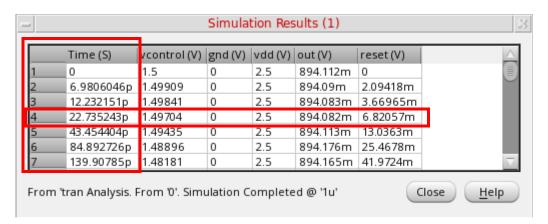
To run a simulation:

➡ Click *Start running simulation from the beginning* 🟢 on the toolbar.

The simulation is run and its progress and other details are displayed in the *Log Information* assistant.

When the simulation run is complete, the results are displayed in the Simulation Results window, as shown below.

You can review the output at different time intervals and decide the scope for debugging, which may involve editing the parameter values, defining trigger events, and actions.

You can also click [icon] on the toolbar and plot the results of the current simulation run in Spectre Interactive Environment in the Virtuoso Visualization and Analysis XL window.

Tip

You can skip running a simulation with the initial setup. In that case, you can directly plot the results from the PSF database saved by ADE Assembler. To do this, click [icon] on the toolbar.

Creating Setup for Debugging

An important task in debugging is to create a setup that helps in identifying the problem area and gives clues for possible changes.

Creation of a setup for debugging involves the following steps:

■   Selecting Outputs for the Debug Simulation Run

■   Editing Parameter Values

■   Forcing Values on Nets

■   Creating Actions and Breakpoints for Events

Selecting Outputs for the Debug Simulation Run

When running Spectre Interactive Environment from ADE Assembler, by default, the outputs from the ADE setup are copied to the *Outputs* section of the Interaction assistant.

When running Spectre Interactive Environment from the command prompt in a new setup, you need to select the outputs to be saved while running a debug simulation.
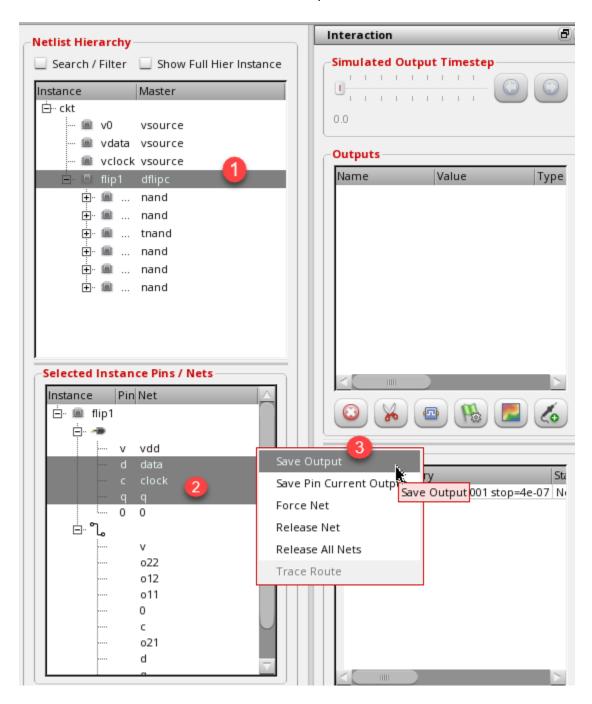
To select outputs, perform the following steps:

1. Select an instance from the list of instances in the upper section of the *Netlist Hierarchy* assistant.
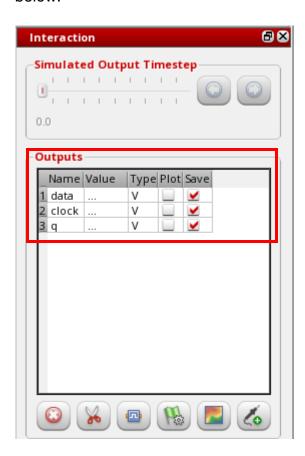
   A list of instance pins, nets connected to pins, and instance parameters, are displayed in the *Selected Instance Pins/Nets* section of the *Netlist Hierarchy* assistant.

2. To save the voltage value for a net, right-click the net name in the *Selected Instance Pins/Nets* section and choose *Save Output*.
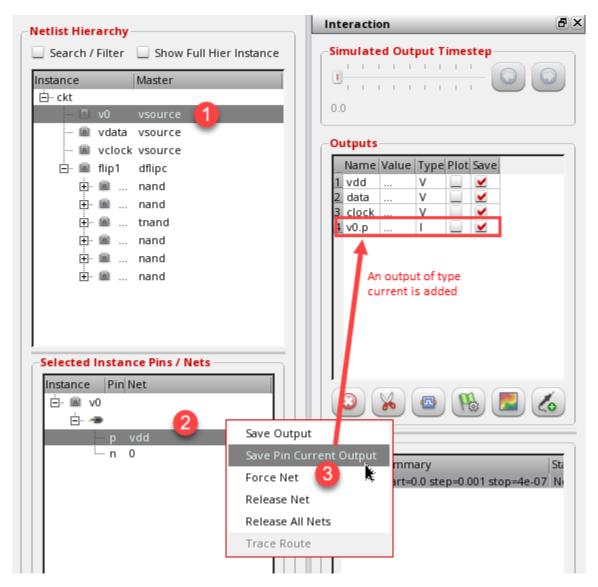
The net name is added to the *Outputs* section of the *Interaction* assistant, as shown below.

3. To save the current value, right-click the net name in the *Selected Instance Pins/Nets* section and choose *Save Pin Current Output*.

An `<instance-name>.<pin-name>` output is added to the *Outputs* section of the *Interaction* assistant.

Note: If you launched Spectre Interactive Environment from ADE Assembler, all outputs from the Assembler setup are copied to the *Outputs* table. The types of outputs supported by Spectre Interactive Environment are 'V' (net voltage), 'I' (terminal current), 'S' (signal), 'M' (model VA var), and 'expr'.
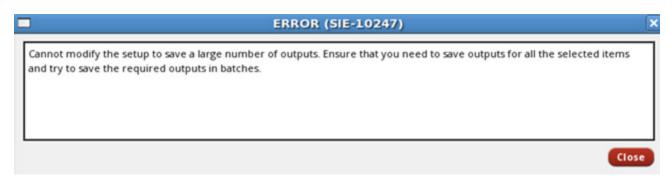
Managing the Selected Outputs

You can manage the outputs selected in the *Outputs* table by using the following commands given at the bottom of Interaction assistant:

Command	Description
	Deletes all outputs from the *Outputs* table.
	Deletes the selected output from the *Outputs* table.
	Selects the *Plot* check box for all outputs. Use this to enable or disable plotting for all outputs.
	Selects the *Save* check box for all outputs. Use this to enable or disable plotting for all outputs.
	Saves the pin current of the instances selected in the *Selected Instance Pins / Nets* section
	Saves the pin net voltage, signal, or Verilog-A variable of the instances selected in the *Selected Instance Pins / Nets* section

⚠️ *Important*

The number of outputs to be saved and plotted affect the performance of a debug simulation. Therefore, select only the relevant outputs to be saved. It is recommended to keep the maximum count of outputs within a few hundreds. If you select a large number of outputs, Spectre Interactive Environment displays the following message:

ERROR (SIE-10247)

Cannot modify the setup to save a large number of outputs. Ensure that you need to save outputs for all the selected items and try to save the required outputs in batches.

Close
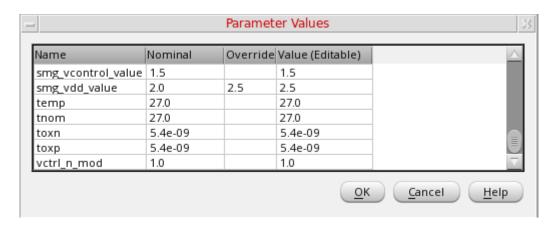
Editing Parameter Values

You can edit the values for subcircuit parameters or instance parameters.

Editing Subcircuit Parameters

To edit the value for any subcircuit parameter in the netlist so that the updated value is used for the simulation run in Spectre Interactive Environment, follow these steps:

1. Choose *Tools – Show/Edit Parameter Values* to open the Parameter Values form.

Name	Nominal	Override	Value (Editable)
smg_vcontrol_value	1.5		1.5
smg_vdd_value	2.0	2.5	2.5
temp	27.0		27.0
tnom	27.0		27.0
toxn	5.4e-09		5.4e-09
toxp	5.4e-09		5.4e-09
vctrl_n_mod	1.0		1.0

Important

> The *Override* column is visible when you open Spectre Interactive Environment from the results of the first simulation point. The values of variables in the first point, or the initial netlist, are considered as base values. Any parameter value modified for other points is considered as an overridden value and displayed in the *Override* column.

2. Edit the value of the required parameter in the *Value (Editable)* column.

   Note: The edited value is used in the next simulation run from Spectre Interactive Environment.

Editing Instance Parameters

To edit the value for any parameter for an instance or pin, perform the following steps:

1. Click an instance in the Instance list shown in the upper section of the *Netlist Hierarchy* assistant.

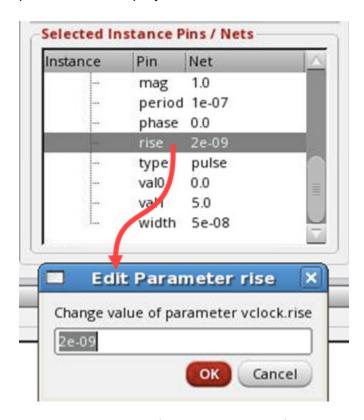A list of instance pins, nets connected to pins, and instance parameters, are displayed in the *Selected Instance Pins/Nets* section of the *Netlist Hierarchy* assistant.

2. Double-click the instance or pin name in the *Selected Instance Pins/Nets* section.

The Edit Parameter <param-name> form is displayed and the current value of the parameter is displayed in a text field.

Selected Instance Pins / Nets

Instance	Pin	Net
	mag	1.0
	period	1e-07
	phase	0.0
	rise	2e-09
	type	pulse
	val0	0.0
	val1	5.0
	width	5e-08

Edit Parameter rise ✕

Change value of parameter vclock.rise

`2e-09`

OK   Cancel

Note: In the above figure, the value of the `rise` parameter of the `vclock` net is shown.

3. Edit the parameter value.

4. Click *OK*.

⚠ *Important*

Currently, you can edit the parameters for an instance of a pin only when you open Spectre Interactive Environment from the command prompt.

Any change in the parameter value takes effect only in the next simulation run from Spectre Interactive Environment. Until then, the changes are considered as `pending` and are not reflected in the Parameter Values form. Before running the next simulation, you can either clear the pending changes to reuse the previous values, or revert all changes in the parameters since the original (initial) run.

Clearing Pending Parameter Changes

To clear the pending parameter value changes that were done after the previous run or before the first run in Spectre Interactive Environment:

➡ Choose *Tools – Clear Pending Instance Parameter Changes*.

When the pending changes are cleared, the parameter values from the previous run are used in the next simulation run.

Note: This command works only when you open Spectre Interactive Environment from the command prompt with `-sie_params`.

Reverting Parameter Changes

To revert the pending parameter value changes that were done after the previous run or before the first run in Spectre Interactive Environment:

➡ Choose *Tools – Revert Instance Parameter Values to Original*.

When all parameter changes are reverted, the parameter values from the initial (original) run are used in the next simulation run.

Note: This command works only when you open Spectre Interactive Environment from the command prompt with `-sie_params`.

Forcing Values on Nets

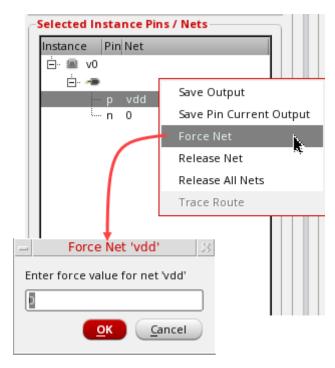If you want to use a specific value for a net, which is different from the value in netlist, and keep it constant while you vary the value of other nets and analyze the result, you can force a value on that net.

To force a value on a net, perform the following steps:

1. Right-click the net in the *Selected Instance Pins/Nets* section of the *Netlist Hierarchy* assistant.

2. Choose *Force Net*.

The Force Net form is displayed.

3. Specify a value to be used for this net.

4. Click *OK*.

Viewing Nets with Forced Values

At any point, you can view all the nets on which values are forced.

To view the nets on which values are forced:

➡ Choose *View – View Current Forced Nets*.

The Forced Nets form is displayed, as shown below.

If required, you can edit the values in the Forced Nets form.

Removing Forced Values from Nets

To remove a forced value from a net, select the net and perform the following steps:

1. Right-click the net in the *Selected Instance Pins/Nets* section of the *Netlist Hierarchy* assistant.

2. Choose *Release Net*.

   Note: You can also remove forced values from all nets by right-clicking a net in the *Selected Instance Pins/Nets* section of the *Netlist Hierarchy* assistant, and choosing *Release All Nets*.

Creating Actions and Breakpoints for Events

While debugging an output, you might be interested in specific events, such as a change in the value of an output and the time when that happens. Accordingly, you might want to analyze the values of other outputs and parameters at that time, or perform a specific action. Using Spectre Interactive Environment, you can specify events to look for and create triggers to perform a specific action when those events happen.

For example, for a `clock` net, you can look for the event when it reaches the rising threshold, and then add a trigger to pause the simulation run.
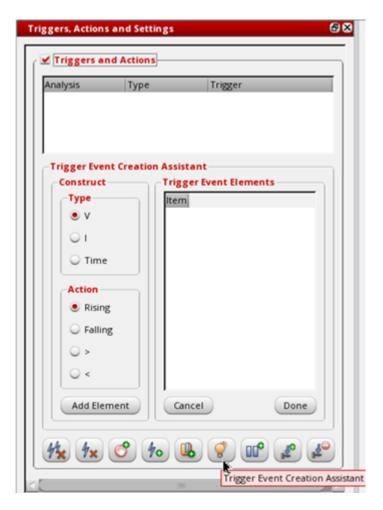
To create an action or breakpoint, you need to first identify the outputs you want to check and create events.

Creating Events

To create an event in Spectre Interactive Environment, perform the following steps:

1. In the *Triggers, Actions and Settings* assistant, click ( ) to open *Trigger Event Creation Assistant*.

2. In the *Interactions* assistant, select an output in the *Outputs* list.

3. In the *Construct* section of *Trigger Event Creation Assistant*, select a type of event and action.

An event can be selected for a change in the following type of values:

❏   *V*: Voltage. This can be used for net voltage, device signals, or Verilog-A variable.

❏   *I*: Current. This can be used for an instance terminal.

❏   *Time*: The current time of a simulation run is checked.

You can look the following type of changes:

❏   *Rising*: The value shows an increasing trend. The event happens when the value of the given type rises above the given threshold.
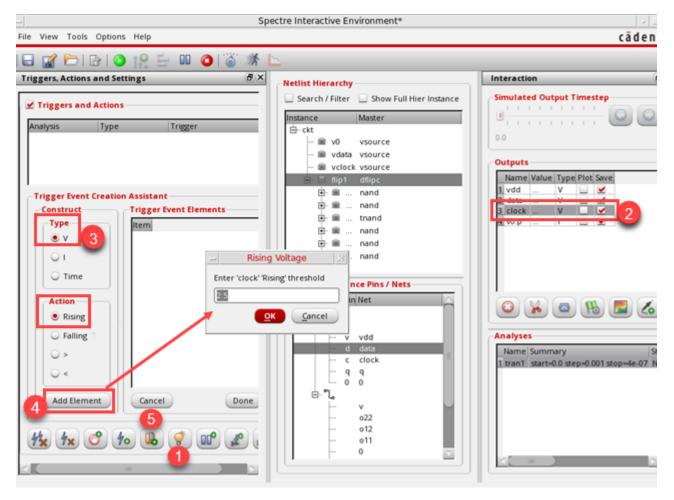
❏ *Falling*: The value shows a decreasing trend. The event happens when the value of the given type falls below the given threshold.

❏ >: The value is greater than the given value. The event happens when the value of the given type becomes greater then the value specified for the event.

❏ <: The value is smaller than the given value. The event happens when the value of the given type becomes less then the value specified for the event.

4. Click *Add Element*.

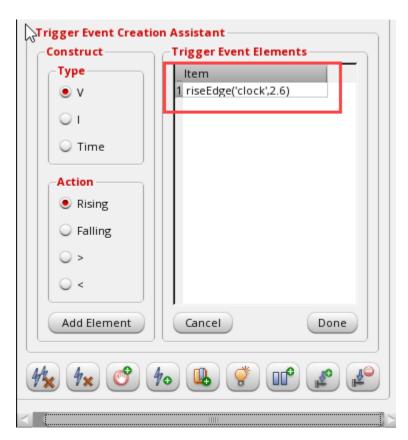A form is displayed where you can enter the threshold value for the element.

5. Click [icon] in the *Triggers, Actions and Settings* assistant.

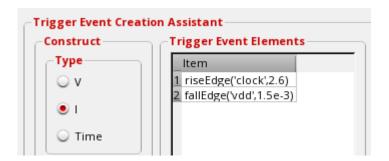The Rising Voltage form is displayed, as shown below.

6. Enter a threshold value and click *OK*.

An event is created with the given details and displayed in the *Trigger Event Elements* list, as shown below.

Similarly, you can create multiple events that you want to look for when a debug simulation is running. An example is shown below.
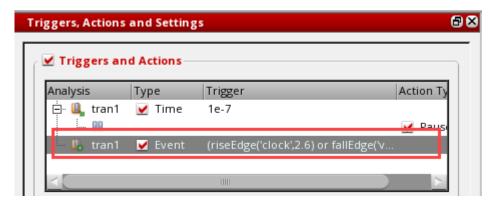

△ *Important*

It is mandatory to include at least one rising or falling action for an event. If you define multiple rising or falling actions, they are joined using OR, which implies that either of these actions can cause this event to happen. However, if you define multiple > or < actions, they are joined using AND.

7. After defining the events, click *Done* in the *Trigger Event Elements* section.

A new row is added to the *Triggers and Actions* table, as shown below.

Next, proceed with creation of triggers and actions as described in Creating Actions and Triggers.

Creating Actions and Triggers

Actions and triggers define the steps to be performed when an event happens while the simulation run for debugging is in progress.

You can define triggers in the following ways:

■ Using the events defined in the *Trigger Event Elements* section of the *Trigger Event Creation Assistant*

■ Using specific time points

■ Using python expressions

■ Using breakpoints for time
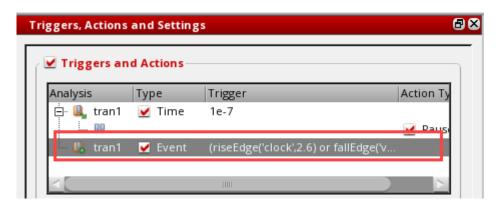
■ Using breakpoints for assertions

Creating Triggers Using Events

To define an action or trigger using the defined events, perform the following steps:

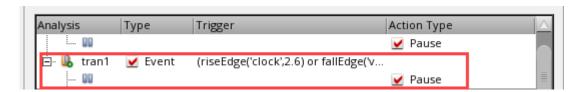**1.** Create an event as described in Creating Events.

For each event a new row is added to the *Triggers and Actions* table, as shown below.

In this row, the left most column shows the name of the transient analysis for which this trigger is applicable. The *Type* column is set to `Event` because the trigger is corresponding to an event. The details of the event are added using an `OR` operator and displayed in the *Trigger* column.

2. Click ⬛ in the *Triggers, Actions and Settings* assistant to add a pause action for the trigger.

A pause action is added for the trigger, as shown below.

Creating Triggers Using Time Points

To create triggers using specific time points, perform the following steps;

1. Click ⬛ in the *Triggers, Actions and Settings* assistant.

The Add Event Time form is displayed.

2. Specify a space- or comma-separated list of time points when you want to trigger an action.

3. Click *OK*.

A new row is added in the *Triggers and Actions* table, as shown below.
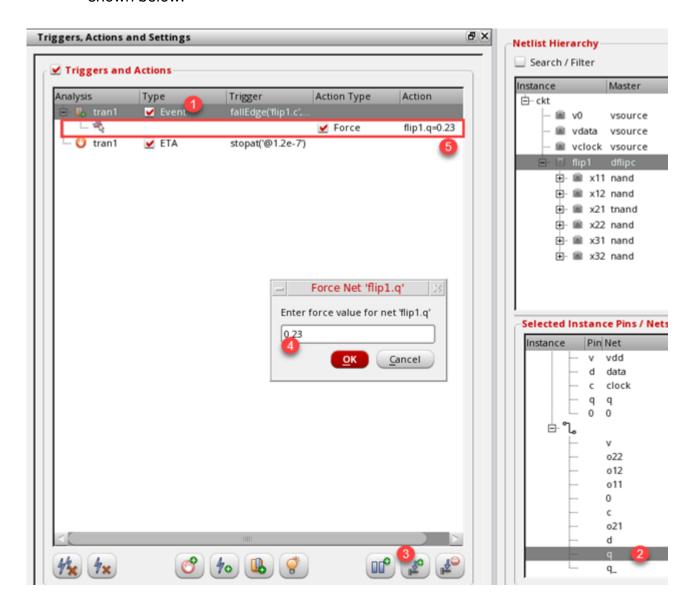
4. Add an appropriate action for the event.

You can add two types of actions:

❑ Pause action: For this, click in the *Triggers, Actions and Settings* assistant to add a pause action for the trigger.

❑ **Force net action:** For this, select a net an click in the *Triggers, Actions and Settings* assistant. The Force Net form is displayed. Add a value that you want to
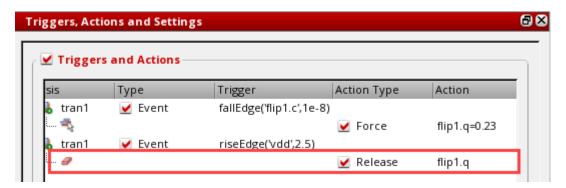
force on the net and click *OK*. A force action is added for the selected event, as shown below.
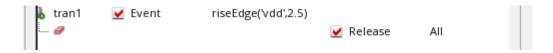
❑ **Release net action:** For this, select a net in the *Selected Instance Pins/Nets* section. Next, click ⬇ in the *Triggers, Actions and Settings* assistant. An action to release net is added for the selected event, as shown below.

Note: If you do not select any net in the *Selected Instance Pins/Nets* section, Spectre Interactive Environment displays a message to confirm whether you want to release the forced values from all nets. If you click *Yes*, an action is created for that, as shown below.
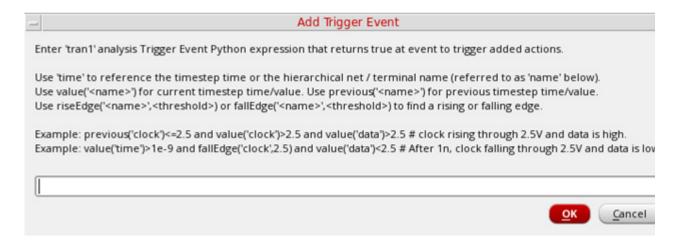
Creating Triggers Using Python Expressions

To create triggers using python expressions, perform the following steps:

1. Click 🔼 in the *Triggers, Actions and Settings* assistant.

The Add Trigger Event form is displayed.

2. Specify an expression for a trigger for the selected analysis.

3. Click *OK*.

A new row is added in the *Triggers and Actions* table.

4. Click in the *Triggers, Actions and Settings* assistant to add a pause action for the trigger.

 Important
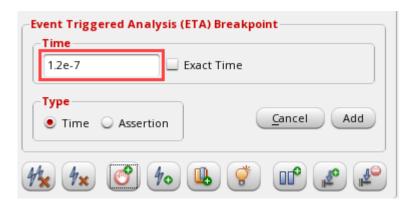
Spectre Interactive Environment does not validate the python expressions. Therefore, ensure the correctness of the code. If Spectre finds an error in the expression, it exits from the simulation.

Using Breakpoints for Specific Time Points

You can add breakpoints to stop a debug simulation at specific time points. To add a breakpoint, perform the following steps:

1. Ensure that *Trigger Event Creation Assistant* is closed. If not, use the toggle button to open or close this assistant.

2. Click in the *Triggers, Actions and Settings* assistant.

3. The *Event Triggered Analysis (ETA) Breakpoint* section is displayed within the assistant.

4. Add a time value in the *Time* field, as shown below.
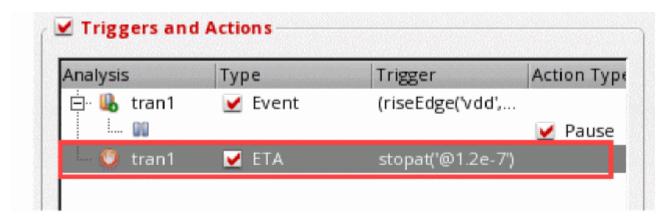
5. Select *Exact Time* if you want to add a breakpoint at exactly the same time. For example, if you specify the stop time as `3.4n` and select *Exact Time*, the simulation will stop exactly at `3.4n`.

   Note: If you do not select *Exact Time*, the simulation will stop at the next time step after the specified stop time. For example, if you do not select *Exact Time*, the simulation can stop either at 3.4n if it happens to be one of the time steps or at any time step after `3.4n`, for example, `3.85n`.

6. Ensure that *Type* is set as *Time*.

7. Click *Add*.

8. A breakpoint trigger with trigger as `stopat` is added in the *Triggers and Actions* table, as shown below.

Note: You do not need to add specific actions for the breakpoints. Therefore, the buttons to add actions for events are disabled when you select a breakpoint trigger in the *Triggers and Actions* table.

Using Breakpoints for Assertions
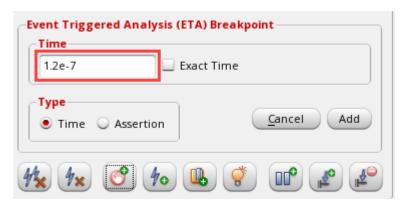
If the Spectre netlist includes assertions, Spectre Interactive Environment can read the assertions from the netlist. You can add breakpoints to pause a debug simulation whenever an assertion defined in the netlist becomes true.

To add a breakpoint for an assertion, perform the following steps:

1. Ensure that *Trigger Event Creation Assistant* closed. If not, you can use the toggle button to open or close this assistant.

2. Click in the *Triggers, Actions and Settings* assistant.
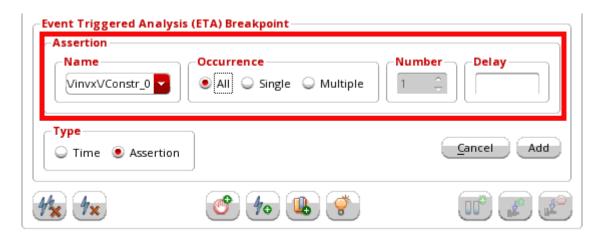
3. The *Event Triggered Analysis (ETA) Breakpoint* section is displayed within the assistant.

4. Select *Assertion* from the *Type* section.

The *Time* section is hidden and the *Assertion* section is displayed, as shown below.
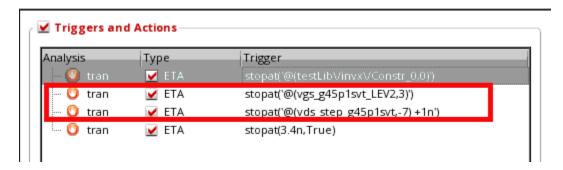
The *Name* drop-down list contains all the assertions saved in the netlist.

5. Select an assertion from the *Name* drop-down list.

6. In the *Occurrence* group box, specify the occurrences you want to check for the selected assertion. The following options are available in this group:

❑ *All*: Adds a breakpoint at every occurrence of the assertion during the simulation run.

❑ *Single*: Adds a breakpoint at the first occurrence of the assertion during the simulation run.

❑ *Multiple*: Adds multiple breakpoints at the first few occurrences of the assertion during the simulation run. You can specify the number of occurrences to check for in the *Number* spin box.

7. Specify the number of occurrences to check for in the *Number* spin box.

8. If you want to add a delay before applying a breakpoint in the simulation run, specify the delay time (in seconds) in the *Delay* field.

9. Click *Add*.

10. A breakpoint trigger with trigger as `stopat` is added in the *Triggers and Actions* table.

When you run the simulation after the setup is complete, a breakpoint is added for these assertion checks for the given number of occurrences.

Running Simulations Using Trigger Actions and Breakpoints

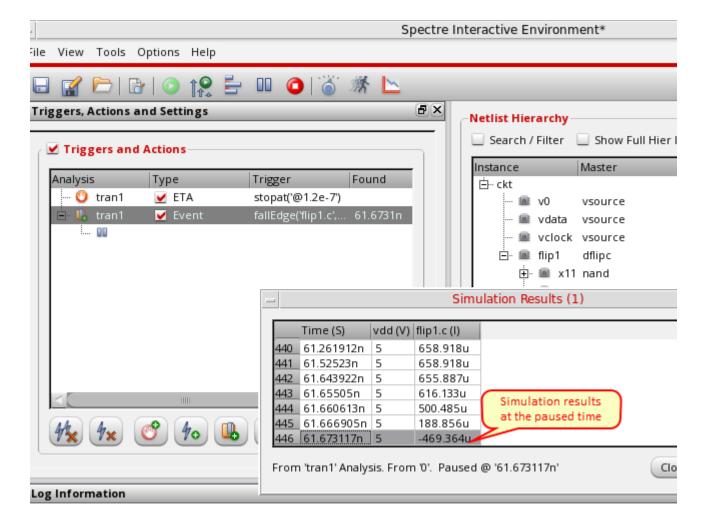When the events and triggers are defined, you can run a debug simulation and use the trigger actions and breakpoints to analyze the outputs values.

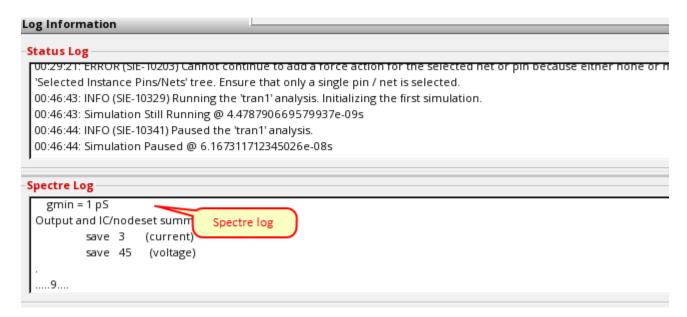To run a debug simulation:

➥ Click on the toolbar.

The simulation starts. Spectre Interactive Environment keeps checking for the defined events. When any event happens, it triggers the action specified for it, as shown in the example given below.
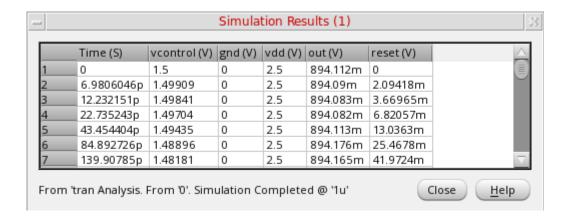
The logs are displayed in the Log Information assistant.

Log Information

Status Log

00:29:21: ERROR (SIE-10203) Cannot continue to add a force action for the selected net or pin because either none or n
'Selected Instance Pins/Nets' tree. Ensure that only a single pin / net is selected.
00:46:43: INFO (SIE-10329) Running the 'tran1' analysis. Initializing the first simulation.
00:46:43: Simulation Still Running @ 4.478790669579937e-09s
00:46:44: INFO (SIE-10341) Paused the 'tran1' analysis.
00:46:44: Simulation Paused @ 6.167311712345026e-08s

Spectre Log

```
   gmin = 1 pS
Output and IC/nodeset summ          Spectre log
        save  3    (current)
        save  45   (voltage)

.....9....
```

The Simulation Results table shows the results of the simulation run at each time step of the transient analysis, as shown below.

Simulation Results (1)

	Time (S)	vcontrol (V)	gnd (V)	vdd (V)	out (V)	reset (V)
1	0	1.5	0	2.5	894.112m	0
2	6.9806046p	1.49909	0	2.5	894.09m	2.09418m
3	12.232151p	1.49841	0	2.5	894.083m	3.66965m
4	22.735243p	1.49704	0	2.5	894.082m	6.82057m
5	43.454404p	1.49435	0	2.5	894.113m	13.0363m
6	84.892726p	1.48896	0	2.5	894.176m	25.4678m
7	139.90785p	1.48181	0	2.5	894.165m	41.9724m

From 'tran Analysis. From '0'. Simulation Completed @ '1u'   Close   Help
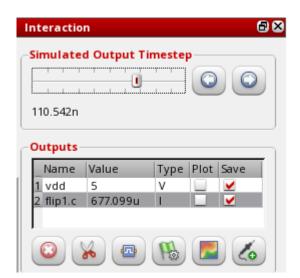
The timesteps are indicated on the *Simulated Output Timestep* progress bar in the *Interaction* assistant. Values of the outputs at a timestep are displayed in the *Value* column of the *Outputs* table.
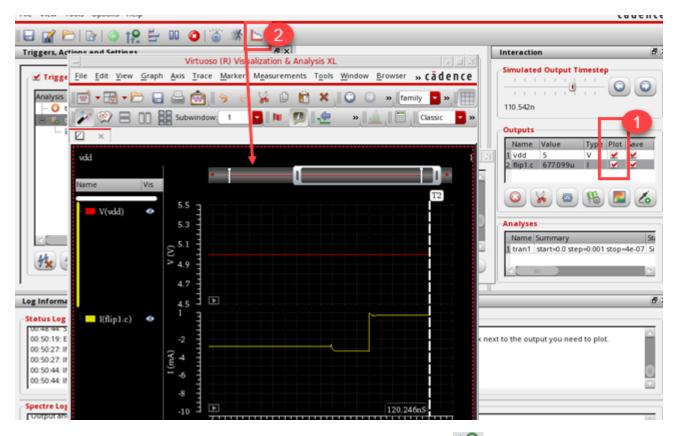

You can move the time steps using the arrows on the *Simulated Output Timestep* progress bar to analyze the outputs at different points.

You can also plot graphs for the outputs. For this, select the Plot check box for the outputs and click ▧ on the toolbar. The ViVA XL window is displayed and the outputs are plotted. A marker is added at the timestep of the breakpoint.

To continue to run the simulation from the paused state, click ↑ on the toolbar. The simulation continues to the end or again breaks at a time, whichever comes first. The results are then displayed in a new Simulation Results table.

Note: The behavior of ↑ depends on the status of ⮂, which can be either to continue to the next time step, or to continue till the end. You can toggle the setting of ⮂ to continue the run from a paused state to the end, or to the next time step when you click ↑.
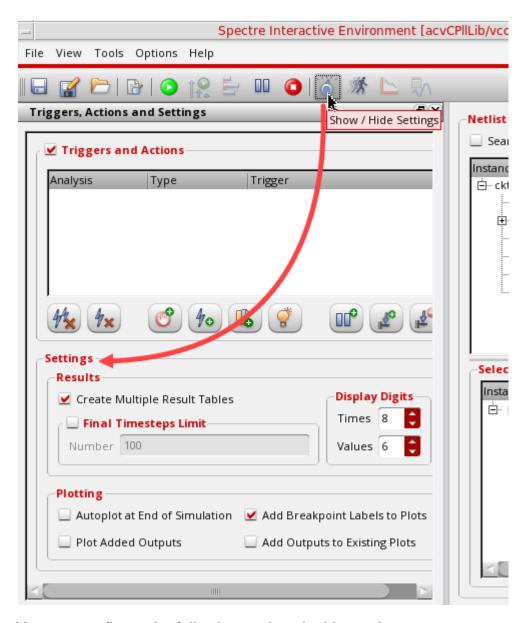
At any point of time during the simulation run, you can click ⏹ to stop the simulation run. If the simulation is already in the paused state and the result table is displayed, the run is stopped. If the simulation is in progress, it is stopped and the displays the results at the current time.

Configuring Settings in Spectre Interactive Environment

To configure various settings in Spectre Interactive Environment:

➡ Click ⚬ on the toolbar.

The *Settings* section is displayed in the Triggers, Actions and Settings assistant.

You can configure the following settings in this section:

■ Results

❑ *Create Multiple Result Tables*: Creates a separate results table after every breakpoint or after the end of simulation. Each result table is saved for a start and end time. When this setting is disabled, all results are appended to a single results table.

❑ *Final Timesteps Limit*: Saves the results for only the final 'n' timepoints before end of simulation or breakpoint instead of saving the results for all the points. Since the results are saved only for the selected time points, the time to run simulations and to save results is reduced a lot.

Note: Set this field if you are interested in only the last few timesteps before the breakpoint.

■ *Display Digits*

❑ *Times*: Controls the number of significant digits to be displayed for time values in the simulation results. When set to 0, the time values are shown with full resolution. The default value is 8.

❑ *Values*: Controls the number of significant digits to be displayed for output values in the simulation results. When set to 0, the output result values are shown with full resolution. The default value is 6.

■ *Plotting*

❑ *Autoplot at End of Simulation*: Update plots with new results at the end of the simulation, or at breakpoints

❑ *Add Breakpoint Labels to Plots*: Shows markers on plots where breakpoints were found

❑ *Plot Added Outputs*: Enables plotting of new outputs to be saved

❑ *Add Outputs to Existing Plots*: Keeps existing plots and adds new plots to the same graphs

⚠ *Important*

You can configure the default values for these settings by using the environment variables documented in Environment Variables.

Working with States in Spectre Interactive Environment

You can save the setup of the Spectre Interactive Environment in a state file and load it again for reuse.

Saving State

To save the state, perform the following steps:

1. Choose *File – Save*.

   Alternatively, click ![save] on the toolbar.

2. In the sie Save FileName form, specify a location and name for the state file.

3. Click *Save*.

   The state is saved with the given file name and `.sie` extension.

Loading State

To load a saved state in the Spectre Interactive Environment interface, perform the following steps:

1. Choose *File – Open*.

   Alternatively, click ![open] on the toolbar.

2. In the sie Load FileName form, select the name for the state file.

3. Click *Open*.

   The state is loaded from the given file.

A

Environment Variables

This appendix describes the environment variables you can use for Spectre Interactive Environment:

■ $SIE_RUN_VIRTUOSO

■ $SIE_EDITOR <command>

■ $SIE_SPECTRE_TIMEOUT <num>

■ $SIE_GET_INSTS_TIMEOUT <num>

■ $SIE_RES_VALUE_DIGITS <num>

■ $SIE_RES_TIME_DIGITS <num>

■ $SIE_INST_LIMIT <num>

$SIE_RUN_VIRTUOSO

When set, runs Virtuoso to plot graphs in ViVA XL. Otherwise, ViVA XL is opened in standalone mode.

Default value: By default, this variable is not set.

$SIE_EDITOR <command>

Specifies the editor to be used to open the files for viewing or editing. The command contains the name of the exe for the text editor to be opened. This variable is used when viewing the input files by using the *View – View Inputs Files* command.

Default value: By default, this variable is not set. This means, the input files are opened in the default dialog as view-only. You cannot edit the files in the default dialog.

Example: `setenv SIE_EDITOR 'nedit'`

$SIE_SPECTRE_TIMEOUT <num>

The maximum time (in seconds) for which Spectre Interactive Environment waits for Spectre to start.

Default value: `600`

$SIE_GET_INSTS_TIMEOUT <num>

The maximum time (in seconds) for which Spectre Interactive Environment waits to get the instance information from the Spectre netlist file.

Default value: `600`

$SIE_RES_VALUE_DIGITS <num>

Sets the number of digits up to which the numeric values in the simulation results are displayed.

Default value: `6`

$SIE_RES_TIME_DIGITS <num>

Sets the number of digits to be used to display the time values in simulation results.

Default value: 8

$SIE_INST_LIMIT <num>

Sets the maximum number of instances to be returned by Spectre from the netlist file. If the number of instances in the netlist is more than the limit set by this variable, an appropriate warning message is displayed.

Default value: 2000