

---

## Getting Started with Spectre

This chapter discusses the following topics:

- [Using the Example and Displaying Results](#) on page 2-2
- [Sample Schematic](#) on page 2-3
- [Sample Netlist](#) on page 2-6
- [Instructions for a Spectre Simulation Run](#) on page 2-11
- [Viewing Your Output](#) on page 2-14

# Using the Example and Displaying Results

In this chapter, you examine a schematic and its Affirma™ Spectre® circuit simulator netlist to get an overview of Spectre syntax. You also follow a sample circuit simulation. The best way to use this chapter depends on your past experience with simulators.

Carefully examine the schematic ([“Sample Schematic”](#) on page 2-3) and netlist ([“Sample Netlist”](#) on page 2-6) and compare Spectre netlist syntax with that of SPICE-like simulators you have used. If you have prepared netlists for SPICE-like simulators before, you can skim [“Elements of a Spectre Netlist”](#) on page 2-7. With this method, you can learn a fair amount about the Spectre simulator in a short time.

Approach this chapter as an overview. You will probably have unanswered questions about some topics when you finish the chapter. Each topic is covered in greater depth in subsequent chapters. Do not worry about learning all the details now.

To give you a complete overview of a Spectre simulation, the example in this chapter includes the display of simulation results with Analog Waveform Display (AWD), a waveform display tool that is included with the Spectre simulator. If you use another display tool, the procedures you follow to display results are different. This user guide does not teach you how to display waveforms with different tools. If you need more information about how to display Spectre results, consult the documentation for your display tool.

The example used in this chapter is a small circuit, an oscillator; you run a transient analysis on the oscillator and then view the results. The following sections contain the schematic and netlist for the oscillator. If you have used

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

---

SPICE-like simulators before, looking at the schematic and netlist can help you compare Spectre syntax with those of other simulators. If you are new to simulation, looking at the schematic and netlist can prepare you to understand the later chapters of this book.

You can also get more information about command options, components, analyses, controls, and other selected topics by using the `spectre -h` command to access the Spectre online help.

## Sample Schematic

A schematic is a drawing of an electronic circuit, showing the components graphically and how they are connected together. The following schematic has several annotations:

- Names of components

Each component is labeled with the name that appears in the instance statement for that component. The names for components are in italics (for example, *Q2*).

- Names of nodes

Each node in the circuit is labeled with its unique name or number. This name can be either a name you create or a number. Names of nodes are in boldface type (for example, **b1**). Ground is node 0.

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

---

#### ■ Sample instance statements

The schematic is annotated with instance statements for some of the components. Arrows connect the components in the schematic with their corresponding instance statements.

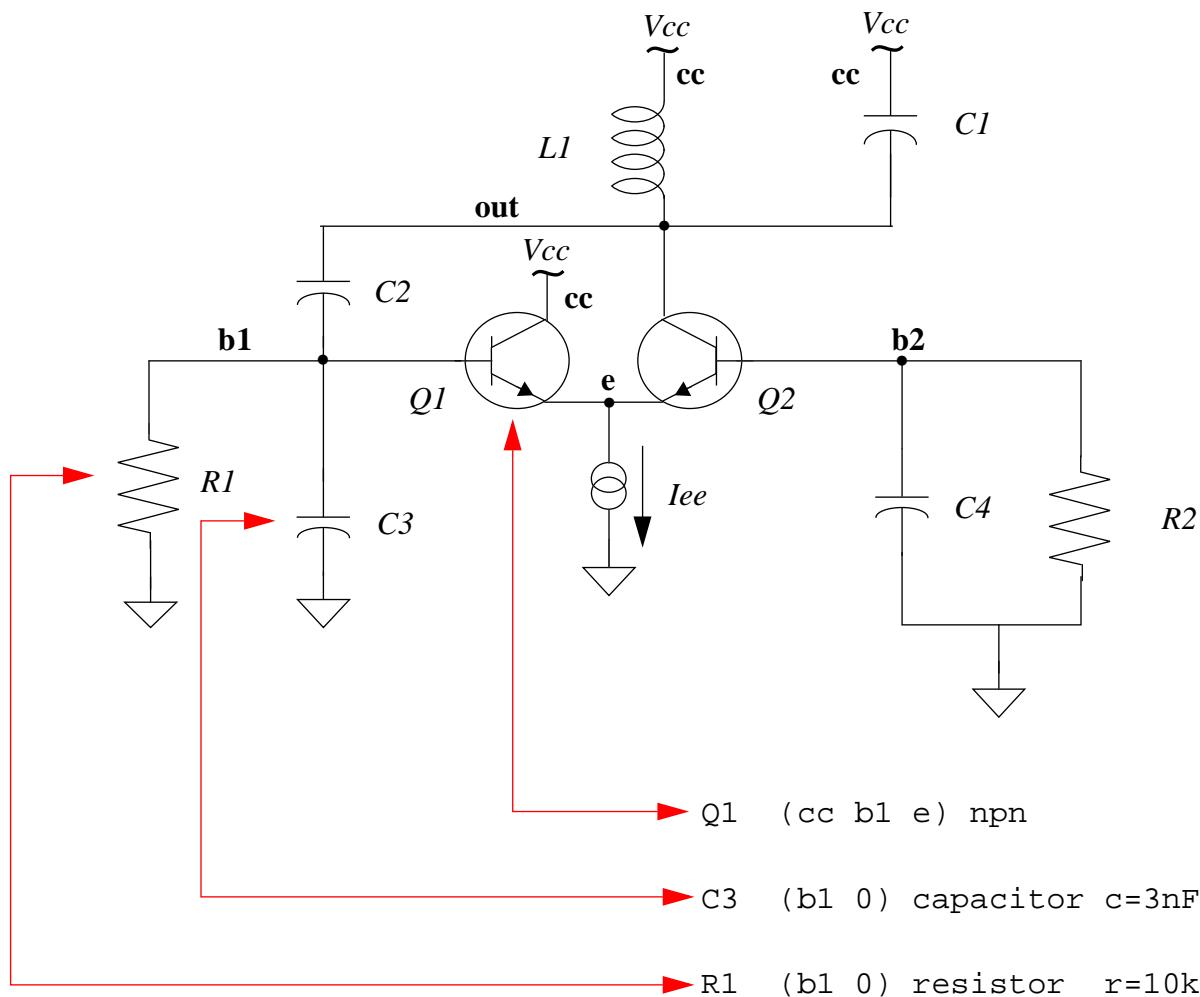
## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

**Bold Type** = Names of nodes. All connections to ground have the same node name.

*Italic Type* = Names of components (also appear in the instance statement for each component).

←→ = Link between components and instance statements.



## Sample Netlist

A netlist is an ASCII file that lists the components in a circuit, the nodes that the components are connected to, and parameter values. You create the netlist in a text editor such as `vi` or `emacs` or from one of the environments that support the Spectre simulator. The Spectre simulator uses a netlist to simulate a circuit.

```
// BJT ECP Oscillator      ← Comment (indicated by //)
simulator lang=spectre   ← Indicates the file contains a Spectre netlist
                           (see the next section). Place below first line.

Vcc (cc 0) vsource dc=5

Iee (e 0)  isource dc=1mA

Q1 (cc b1 e) npn
Q2 (out b2 e) npn

L1 (cc out) inductor l=1uH
C1 (cc out) capacitor c=1pf
C2 (out b1) capacitor c=272.7pF
C3 (b1 0) capacitor c=3nF
C4 (b2 0) capacitor c=3nF

R1 (b1 0) resistor r=10k
R2 (b2 0) resistor r=10k
```

Instance statements

```
ic b1=1      ← Control statement (sets initial conditions)

model npn bjt type=npn bf=80 rb=100 vaf=50 \
cjs=2pf tf=0.3ns tr=6ns cje=3pf cjc=2pf
```

Model statement

```
OscResp tran stop=80us maxstep=10ns    ← Analysis statement
```

## Elements of a Spectre Netlist

This section briefly explains the components, models, analyses, and control statements in a Spectre netlist. All topics discussed here (such as `model` statements or the `simulator lang` command) are presented in greater depth in later chapters. If you want more complete reference information about a topic, consult these discussions.

### Title Line

The first line is taken to be the title. It is used verbatim when labeling output. Any statement you place in the first line is ignored as a comment. For more information about comment lines, see [“Basic Syntax Rules”](#) on page 4-4.

### Simulation Language

The second line of the sample netlist indicates that the netlist is in the Spectre Netlist Language, instead of SPICE. For more information about the `simulator lang` command, see [Chapter 3, “SPICE Compatibility.”](#)

### Instance Statements

The next section in the sample netlist consists of instance statements. To specify a single component in a Spectre netlist, you place all the necessary information for the component in a netlist statement. Netlist statements that specify single components are called instance statements. (The instance statement also has other uses that are described in [Chapter 4, “Spectre Netlists.”](#))

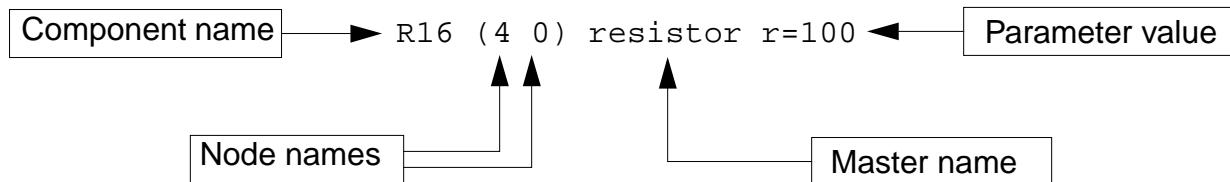
## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

To specify single components within a circuit, you must provide the following information:

- A unique component name for the component
- The names of nodes to which the component is connected
- The master name of the component (identifies the type of component)
- The parameter values associated with the component

A typical Spectre instance statement looks like this:



**Note:** You can use balanced parentheses to distinguish the various parts of the instance statement, although they are optional:

```
R1 (1 2) resistor r=1
Q1 (c b e s) npn area=10
Gm (1 2)(3 4) vccs gm=.01
R7 (x y) rmod (r=1k w=2u)
```

### Component Names

Unlike SPICE, the first character of the component name has no special meaning. You can use any character to start the component name. For example:

```
Load (out o) resistor r=50
Balun (in o pout nout) transformer
```

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

---

**Note:** You can find the exact format for any component in the parameter listings for that component in the Spectre online help.

#### Master Names

The type of a component depends on the name of the master, not on the first letter of the component name (as in SPICE); this feature gives you more flexibility in naming components. The master can be a built-in primitive, a model, a subcircuit, or an AHDL component.

#### Parameter Values

Real numbers can be specified using scientific notation or common engineering scale factors. For example, you can specify a 1 pF capacitor value either as `c=1pf` or `c=1e-12`. Depending on whether you are using the Spectre Netlist Language or SPICE, you might need to use different scale factors for parameter values. Only ANSI standard scale factors are used in Spectre netlists. For more information about scale factors, see [“Instance Statements”](#) on page 4-7.

#### Control Statements

The next section of the sample netlist contains a control statement, which sets initial conditions.

#### Model Statements

Some components allow you to specify parameters common to many instances using the `model` statement. The only parameters you need to specify in the instance statement are those that are generally unique for a given instance of a component.

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

---

You need to provide the following for a `model` statement:

- The keyword `model` at the beginning of the statement
- A unique name for the model (reference by master names in instance statements)
- The master name of the model (identifies the type of model)
- The parameter values associated with the model

The following example is a `model` statement for a bjt. The model name is `nPN`, and the component type name is `bjt`. The backslash (\) tells you that the statement continues on the next line. The backslash must be the last character in the line because it escapes the carriage return.

```
model nPN bjt type=nPN bf=80 rb=100 vaf=50 \
    cjs=2pf tf=0.3ns tr=6ns cje=3pf cjc=2pf
```

When you create an instance statement that refers to a `model` statement for its parameter values, you must specify the model name as the master name. For example, an instance statement that receives its parameter values from the previous `model` statement might look like this:

```
Q1 (vcc b1 e vcc) nPN
```

Check documentation for components to determine which parameters are expected to be provided on the instance statement and which are expected on the model statement.

## Analysis Statements

The last section of the sample netlist has the analysis statement. An analysis statement has the same syntax as an instance statement, except that the

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

---

analysis type name replaces the master name. To specify an analysis, you must include the following information in a netlist statement:

- A unique name for the analysis statement
- Possibly a set of node names
- The name of the type of analysis you want
- Any additional parameter values associated with the analysis

To find the analysis type name and the parameters you can specify for an analysis, consult the parameter listing for that analysis in the Spectre online help (`spectre -h`).

The following analysis statement specifies a transient analysis. The analysis name is `stepResponse`, and the analysis type name is `tran`.

```
stepResponse tran stop=100ns
```

## Instructions for a Spectre Simulation Run

When you complete a netlist, you can run the simulation with the `spectre` command.

- To run a simulation for the sample circuit, type the following at the command line:

```
spectre osc.cir
```

**Note:** `osc.cir` is the file that contains the netlist.

## **Following Simulation Progress**

As the simulation runs, the Spectre simulator sends messages to your screen that show the progress of the simulation and provide statistical information. In the simulation of `osc.cir`, the Spectre simulator prints some warnings and notifications. The Spectre simulator tells you about conditions that might reduce simulation accuracy. When you see a Spectre warning or notification, you must decide whether the information is significant for your particular simulation.

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

## Screen Printout

The printout for the `osc.cir` simulation looks like this:

```
spectre (ver. 4.4.3.52_339 -- 02 Jul 1998).  
Simulating `osc.cir' on cds8616 at 9:24:15 AM, Mon Jul 6, 1998.
```

Circuit inventory:

```
    nodes 5  
equations 9  
      bjt 2  
capacitor 4  
inductor 1  
  isource 1  
resistor 2  
  vsource 1
```

```
*****  
Transient Analysis `OscResp': time = (0 s -> 80 us)  
*****  
.....9.....8.....7.....6.....5.....4.....3.....2.....1.....0  
Number of accepted tran steps = 11090.  
Initial condition solution time = 10 ms.  
Intrinsic tran analysis time = 15.53 s.  
Total time required for tran analysis `OscResp' was 15.58 s.
```

↑ Narration of transient analysis progress

```
Aggregate audit (9:24:38 AM, Mon Jul 6, 1998):  
Time used: CPU = 17.7 s, elapsed = 23 s, util. = 77%.  
Virtual memory used = 1.39 Mbytes.  
spectre completes with 0 errors, 0 warnings, and 0 notices
```

# Viewing Your Output

The waveform display tool for this simulation example is AWD, a display tool you receive when you purchase the Spectre simulator. In this section you will learn the following:

- How to start AWD
- How to display waveforms with the Results Browser
- How to extract zero-crossing points of a waveform
- How to select new colors for waveforms
- How to access data from multiple directories

## Starting AWD

There are several ways to start AWD.

### Using the **-dataDir** Command Line Option

- To start AWD for the simulation example, type the following at the command line (from the directory where the Spectre simulator was run):  
`awd -dataDir osc.raw`

### Using Just the **awd** Command

1. To start AWD for the simulation example, type the following at the command line (from the directory where the Spectre simulator was run):  
`awd osc.raw`

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

---

#### 2. In the Browse Project Hierarchy form, click *OK*.

For this command to work, you need to specify which simulator you are using. The default is Spectre. To specify the simulator, create a `.cdsinit` file and put the following line in it:

```
envSetVal( "asimenv.startup" "simulator" 'string "spectre" )
```

Save the `.cdsinit` file in the directory from which you start AWD or in your home directory. AWD looks for the `.cdsinit` file first in the directory from which AWD was started; if the file is not there, AWD looks for it in your home directory.

Another option is to place the following entry in a `.cdsenv` file:

```
asimenv.startup simulator string "spectre"
```

Save the `.cdsenv` file in your home directory. If AWD does not find the `.cdsenv` file in your home directory, AWD looks for the system-defined default settings.

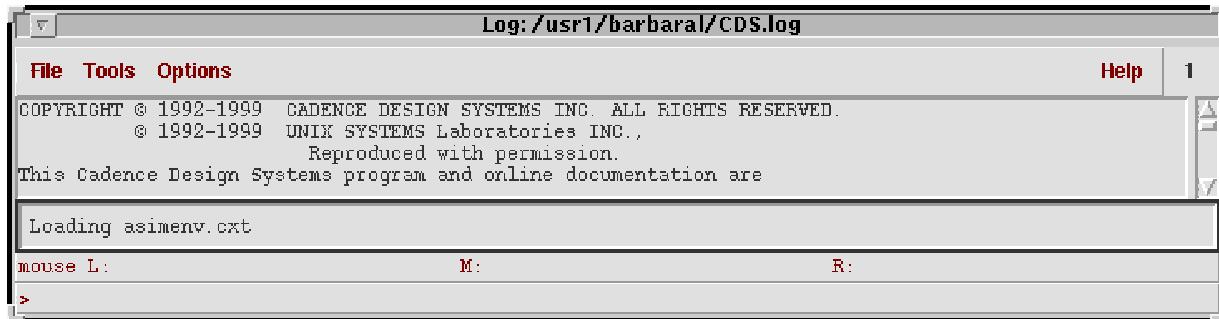
## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

---

#### AWD and Other Windows

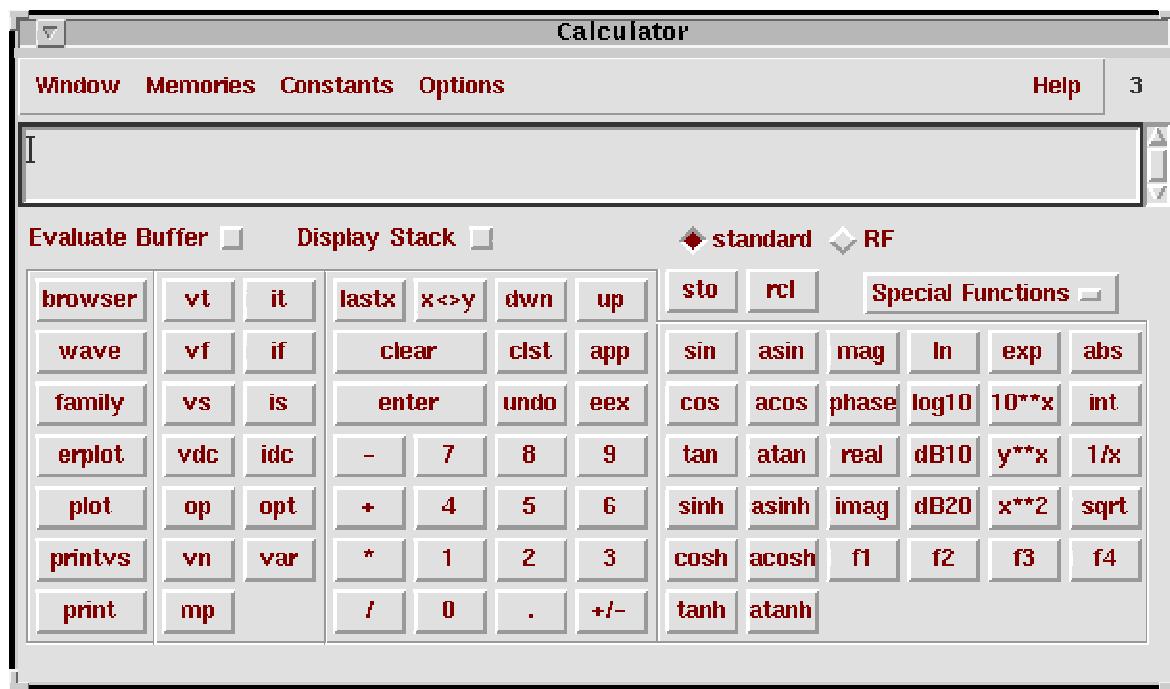
Once AWD is running, four windows appear on your screen. They are shown in the following figures.



Command Interpreter Window (CIW)

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre



Calculator Window

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

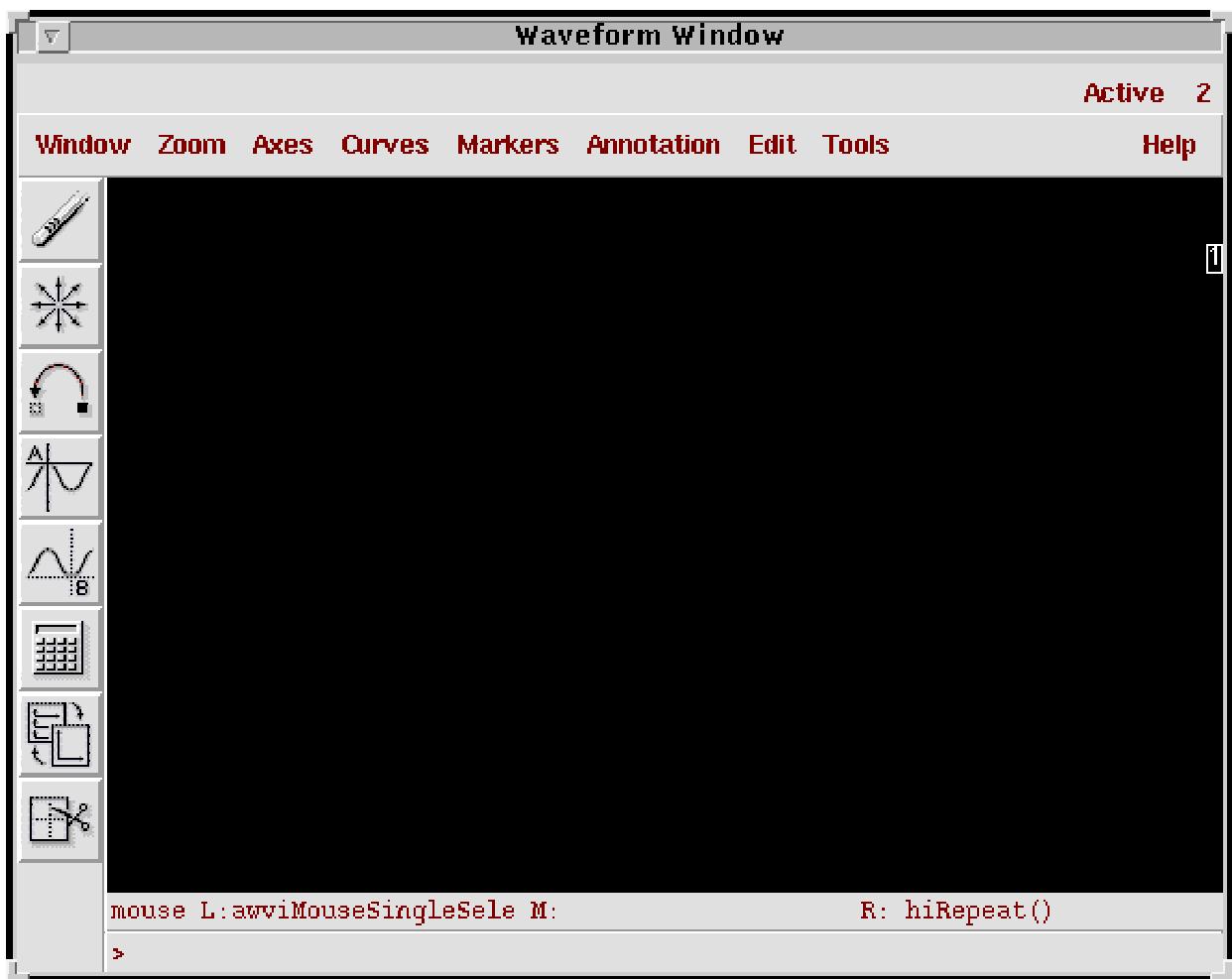
---



Results Browser

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

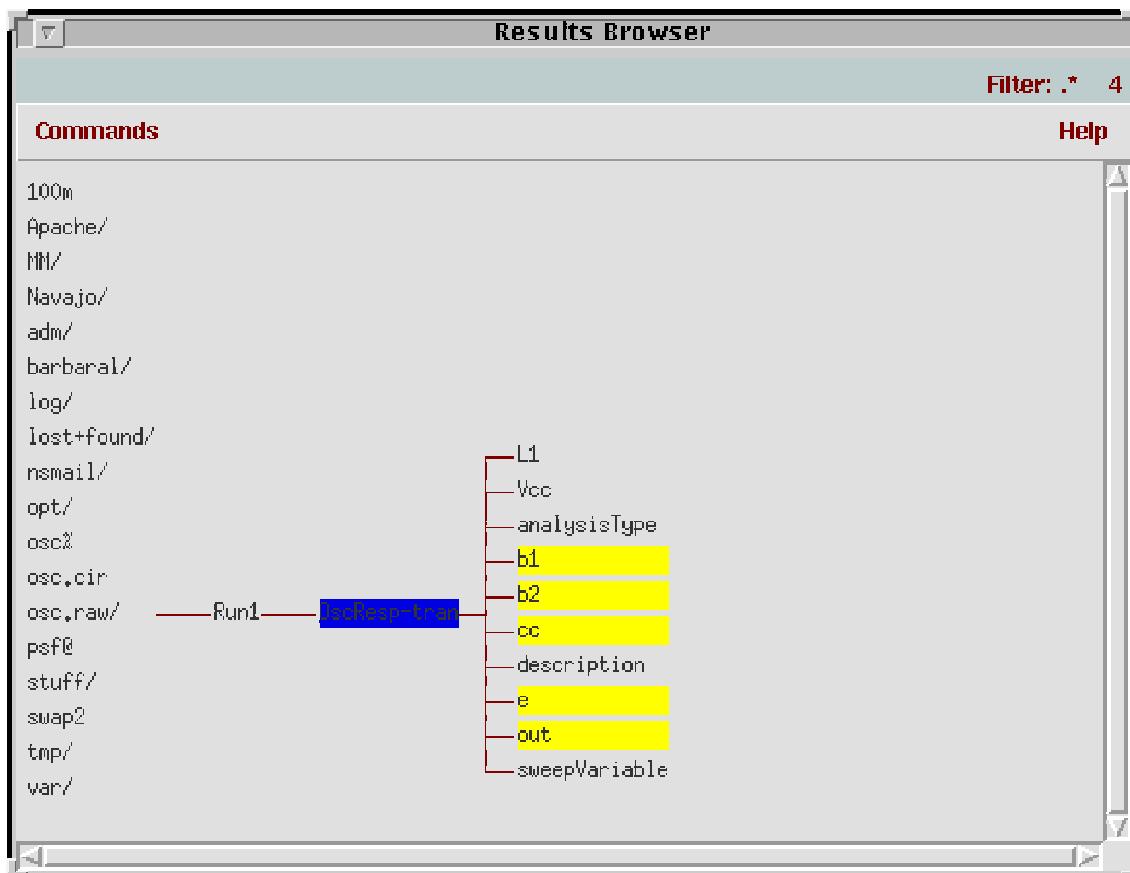


Waveform Window

## Displaying a Waveform

1. In the Results Browser, scroll, if necessary, until `osc.raw` is visible and then click on `osc.raw`.

The display in the Results Browser now looks like this:



The lowest level of the hierarchy in the Results Browser now contains the names of the waveforms you can display.

## **Affirma Spectre Circuit Simulator User Guide**

### Getting Started with Spectre

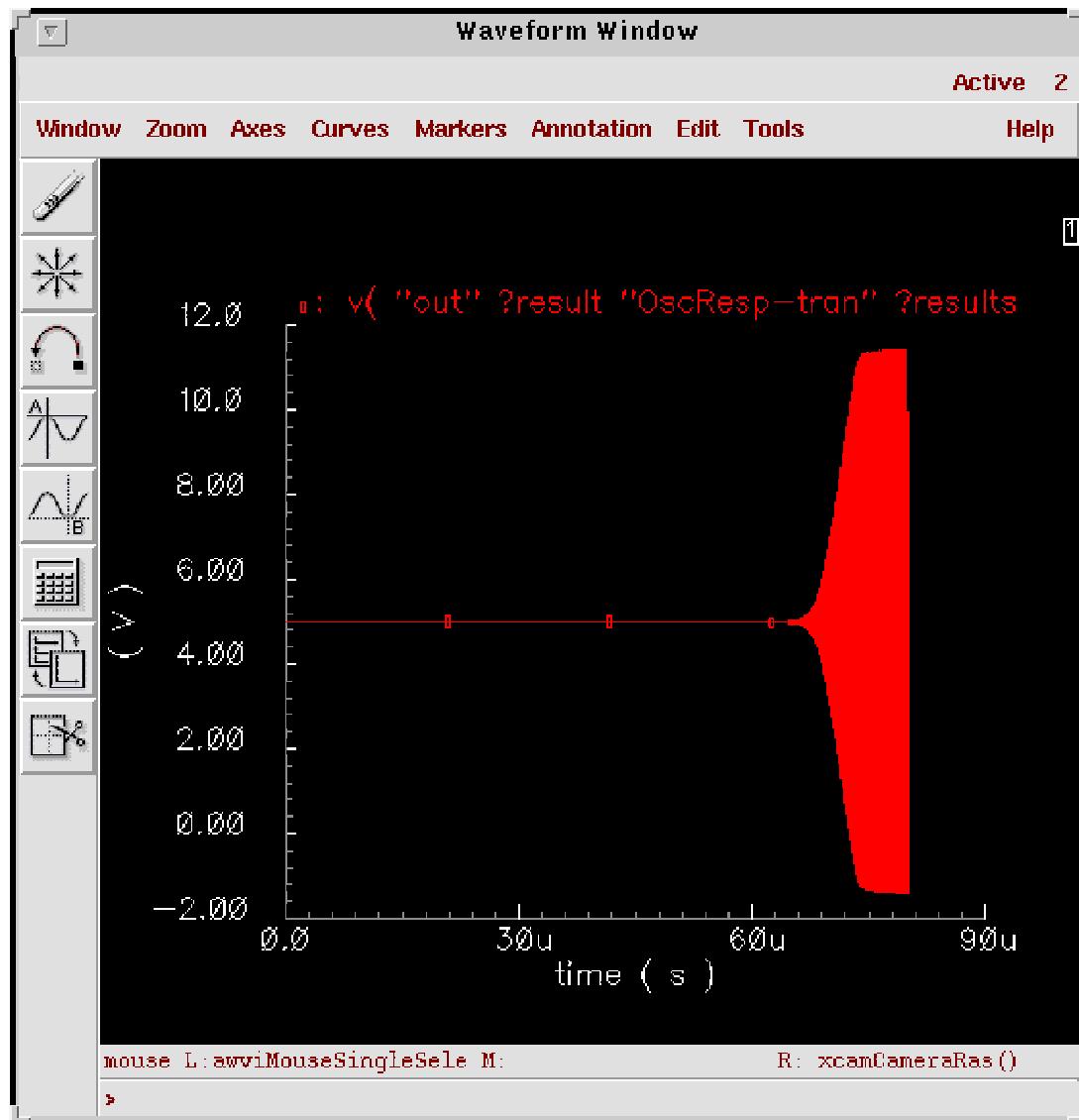
---

2. To display a waveform, click on its name in the Results Browser with the right mouse button.

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

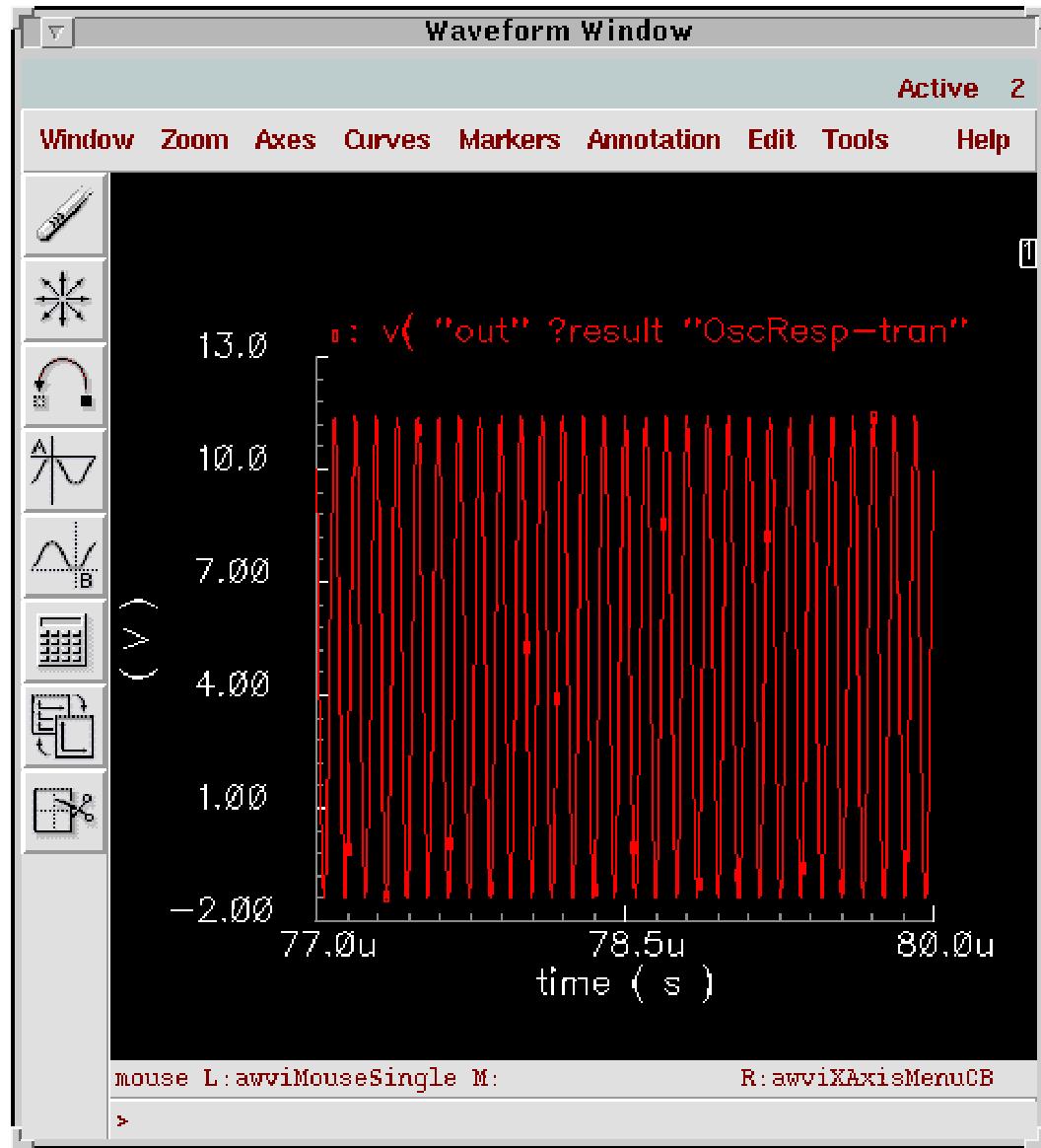
For example, if you click on *out*, the following display appears in the Waveform Window:



## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

If you zoom in on the right side of the waveform (*Zoom – Zoom In*) and adjust the X axis (*Axes – X Axis*), you see the following:



You can add a more meaningful title with the *Annotation – Title* command.

## Extracting Zero-Crossing Points

To extract zero-crossing points of a waveform using the calculator:

1. Make certain that the *Evaluate Buffer* button in the calculator is disabled.
2. Select the waveform in the Results Browser by clicking the left mouse button over the name of the waveform.
3. Choose *cross* from the Special Functions menu in the calculator  
The Threshold Crossing form appears.
4. Type 0 in the *Threshold Value* field, type 0 in the *Edge Number* field, and set *Edge Type* to *either*.
5. Click *OK*.
6. Enable the *Evaluate Buffer* button in the calculator.

This returns the list of zero-crossing points (time co-ordinates) of the waveform in the CIW.

## Selecting New Colors for Waveforms

To select a new color for a waveform:

1. In the Waveform Window, choose *Curves – Edit*.  
The Curves form appears.

2. In the Curves form, select the curve name you want and the pen color.
3. Click *OK*.

## Accessing Data from Multiple Directories

There are two ways to access data from multiple directories:

- If you are accessing data from multiple directories and each unique data directory path ends with a directory called `psf`, use the `awd` command by itself and click *OK* in the Browse Project Hierarchy form. In the Results Browser, expand the hierarchy down to a data directory, click the middle mouse button over the waveform name, and select the *Create ROF* command from the middle mouse button menu. Then expand this data tree to access the data.

To access a different data directory, back up in the hierarchy in the Results Browser and then expand down to some other data directory. Again, select the *Create ROF* command for that particular `psf` directory and expand down to the data. Data can be overlaid on top of data from a previous data directory.

- If you are accessing data from multiple directories and the data directory path names do not end with `psf` as the last directory, you can still use the `awd` command by itself.

When you expand down to a data directory and select the *Create ROF* command, the system creates a soft link from that data directory to `psf`. You can now expand down to the data. If you then move to a different data directory in the Results Browser, you have to again select the *Create ROF* command, even if a run object file already exists. This removes the old link

## Affirma Spectre Circuit Simulator User Guide

### Getting Started with Spectre

---

(if the two data directories exist in the same level of hierarchy) and reestablishes the `psf` link to the new data directory. Now you can expand down to data. If you want to plot from the calculator a net you had previously plotted already (for example, overlay the same net from two different runs), you need to first flush out the cache memory.

## Learning More about AWD

The AWD display tool gives you a number of additional options for displaying your data. To learn more about using AWD with the Spectre simulator, see the Cadence application note, “Using `awd` with Standalone Spectre,” available from Cadence Customer Support, and the [Analog Waveform User Guide](#) about displaying results for the Affirma analog design environment.