



Agilent Technologies

**Advanced Design System 2002
RFIC Dynamic Link**

February 2002

Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms that apply to this software product is available upon request from your Agilent Technologies representative.

Restricted Rights Legend

Use, duplication or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DoD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Agilent Technologies
395 Page Mill Road
Palo Alto, CA 94304 U.S.A.

Copyright © 2002, Agilent Technologies. All Rights Reserved.

Acknowledgments

Cadence® and Analog Artist® are registered trademarks of Cadence Design Systems Incorporated.

Design Framework II™ and Composer™ are trademarks of Cadence Design Systems Incorporated.

Copyright © 2001 Cadence Design Systems Incorporated. All rights reserved.

Contents

1 Introduction

Advanced Design System	1-1
Virtuoso Schematic Composer	1-1
RFIC Dynamic Link.....	1-2
RFIC Dynamic Link Use Model	1-2
What's in this Manual.....	1-3

2 Administrative Tasks

System Requirements	2-1
Hardware Requirements.....	2-1
Software Requirements	2-1
License Requirements.....	2-2
Installing RFIC Dynamic Link	2-3
Configuring the Cadence Install Directory	2-5
Running the idfConfigCadence Script	2-6
Configuring for a New Cadence Release	2-7
Configuring the Software	2-7
Configuring the UNIX Environment	2-8
Configuring the ADS Install Directory.....	2-8
Modifying the RFIC Dyanmic Link-Cadence Initialization File	2-9
Modifying the Configuration File.....	2-9
Modifying the BindKey Settings	2-12
Managing Projects and Designs.....	2-12

3 Getting Started Tutorial

Setting up the Examples Directory	3-1
Starting the Cadence Design Framework.....	3-1
Opening a Cadence Composer Schematic	3-2
Linking with Advanced Design System.....	3-4
Opening a Test Schematic Design	3-6
Adding a Symbol of the Cadence Cellview.....	3-7
Adding Design Variables	3-9
Adding Model Files	3-9
Performing a DC Simulation	3-13
Annotating DC Operating Points to a Selected Cellview	3-15
Node Probing with RFIC Dynamic Link	3-17
Performing an S-parameter Simulation	3-24
Displaying Your Results.....	3-25
Performing a Parameter Optimization.....	3-27
Verifying Your Results.....	3-32

Ending the Session.....	3-33
4 Starting, Viewing Designs and Exiting	
Starting Advanced Design System	4-1
Adding an Instance of a Cadence Design	4-2
Pushing into the Design Hierarchy	4-3
Exiting.....	4-3
5 Netlisting, Simulating, and Displaying Data	
Netlisting and Simulating a Design.....	5-1
Viewing Netlists	5-1
Viewing Netlists from Advanced Design System	5-1
Viewing Netlists from the Cadence Schematic Window	5-2
Net and Instance Name Mapping	5-3
Expression Name Mapping	5-4
Using Global Nodes.....	5-6
6 Using Design Variables	
Adding and Editing Design Variables	6-1
Updating Cadence Design Variables.....	6-2
7 Tuning and Optimizing Designs	
Tuning Cadence Instance Parameters and Design Variables.....	7-1
A Dynamic Link For Cadence Tuning Example	7-1
Optimizing Designs.....	7-8
Updating the Cadence Cellview	7-11
8 Annotating a DC Solution	
Annotating DC Voltages to a Cadence Cellview.....	8-1
Annotating DC Currents to a Cadence Cellview.....	8-2
Displaying Voltages or Currents from a Previous Simulation.....	8-5
Creating Symbols for Hierarchical Subcircuits with cdsTerm.....	8-6
9 Using Additional Features of RFIC Dynamic Link	
Using the Netlist File Include Component	9-1
Adding a Netlist File Include Component	9-1
Accessing the Netlist File Include Dialog	9-2
Summarizing the Netlist File Include Component	9-9
Freezing Selected Subcircuits	9-10
Setting the Freeze Parameter.....	9-10
Generating a Cadence Subcircuit Netlist.....	9-11
Setting the netlistFile Parameter	9-12
Using “Freeze” Mode to Simulate a Design in ADS Standalone	9-12
Compatibility between Advanced Design System and Cadence Tools.....	9-13
Support for Duplicate Pin Names	9-13

Using Buses	9-14
Setting up Unnamed Nets	9-14
Support for pPar and iPar	9-14
Using Inherited Connections	9-20
Using S-parameter File Devices from analogLib	9-21
10 Using Switch Views, Stop Views and the Hierarchy Editor	
Expanding Hierarchy with the Dynamic Link Netlister	10-1
Using the Hierarchy Editor with RFIC Dynamic Link	10-10
Placing the config view in ADS	10-18
A Command Reference and Troubleshooting	
Cadence Menu	A-1
ArtistUtilities Menu	A-3
ADS Menu	A-4
Troubleshooting	A-4
Known Problems and Solutions	A-4
Installation and Use Checklist	A-6
Glossary	
Index	

Chapter 1: Introduction

Agilent Technologies and Cadence Design Systems both offer powerful EDA design tools. Many of today's design engineers prefer to use a combination of these tools to take advantage of the strengths of both design environments. Because of this desire to use multiple tools, Agilent Technologies has developed the *RFIC Dynamic Link for Cadence*. The Dynamic Link enables both *tops-down* and *bottoms-up* design and simulation in Advanced Design System (ADS) using IC designs from the Cadence database.

Advanced Design System

Advanced Design System has been developed specifically to simulate the entire communications signal path. This unique solution integrates the widest variety of proven RF, DSP, and electromagnetic design tools into a single, flexible environment. Building on years of expertise developing new technologies for our EDA tools, such as Series IV and MDS, Advanced Design System provides a broad range of high-performance capability. This makes it easy to explore design ideas, then model the electrical and physical design of the best candidates.

Virtuoso Schematic Composer

The Virtuoso Schematic Composer from Cadence Design Systems is a hierarchical design entry tool used by RFIC circuit designers. Useful for both analog and digital designs, the database created is accessible by the Cadence simulation and physical layout tools. The tool supports multi-sheet schematics, including cross-referencing, symbol creation, automatic HDL cell template generation, global nets and hierarchical property definition for most database objects. The tool also provides hierarchical checking of connectivity, consistency of different cell representations and label attachments.

RFIC Dynamic Link

RFIC Dynamic Link is an EDA framework integration software product based on Inter-Process Communication (IPC), rather than data file translation, maximizing data integrity and ease of use. This manual describes how to install and configure the RFIC Dynamic Link product and assists you in designing and analyzing analog mixed-signal and RF circuits via Dynamic Link. [Chapter 3, Getting Started Tutorial](#) is provided to help you quickly get started with using the RFIC Dynamic Link. For information on Library Customization, refer to the *RFIC Dynamic Link Library Guide*.

RFIC Dynamic Link Use Model

The RFIC Dynamic Link use-model coincides with that of both Cadence and ADS, with only a few exceptions. Essentially, the *Affirma Analog Circuit Design Environment (Analog Artist* in 4.4.3) user interface is replaced with the Advanced Design System and all of its functionality. The Affirma features that are not directly replaced by ADS are provided on the *ArtistUtilities* pull-down menu in the Cadence Virtuoso Schematic window.

Note If the *ArtistUtilities* pull-down menu does not appear in the Cadence Virtuoso Schematic window, choose **Tools > ADS** in the Cadence schematic window.

Usage assumes basic familiarity with the Cadence *IC Design Framework II (DFII)*, including *Virtuoso* schematic capture and *Affirma Analog Circuit Design Environment*, as well as basic familiarity with design and simulation in the Advanced Design System.

Additional Information

- Wherever a shell variable is set, this manual uses the K-shell syntax.
- Unless otherwise mentioned, assume case sensitivity.
- Terminology used for Agilent Technologies and Cadence EDA Tools is frequently different. For example, a *project* in ADS is similar to a *library* in Cadence and *design* in ADS is similar to a *cellview* in Cadence.

What's in this Manual

The goal of this manual is to help you get started, providing relevant examples that teach you how to use the software, and show you where you can get more information as you need it. This manual contains:

- [Chapter 2, Administrative Tasks](#) describes the system requirements and how to install and configure the software.
- [Chapter 3, Getting Started Tutorial](#) steps you through the process of simulating a circuit using components from the Dynamic Link analogLib library. Other examples are also included to help you become more familiar with the product.
- [Chapter 4, Starting, Viewing Designs and Exiting](#) provides information on launching ADS from a Cadence Schematic window, performing some basic operations and closing the Dynamic Link between ADS and Cadence.
- [Chapter 5, Netlisting, Simulating, and Displaying Data](#) describes the procedures for netlisting and simulating a design as well as viewing the netlist from either ADS or a Cadence schematic window. Information on net, instance and expression name mapping is also provided.
- [Chapter 6, Using Design Variables](#) describes how to edit a design variable in ADS and also update your Cadence design variables.
- [Chapter 7, Tuning and Optimizing Designs](#) provides information on tuning and optimizing designs using the ADS tuning and optimization capabilities.
- [Chapter 8, Annotating a DC Solution](#) describes the steps necessary for annotating ADS DC and Operating Point simulation results to the Cadence schematic design.
- [Chapter 9, Using Additional Features of RFIC Dynamic Link](#) includes a collection of Dynamic Link features such as using the Netlist File Include component and “Freezing” selective subcircuits. Compatibility features covering support for Duplicate Pin Names, Bus-ports, Buses and Bundles, Unnamed Nets and pPar and iPar are also discussed.
- [Chapter 10, Using Switch Views, Stop Views and the Hierarchy Editor](#) provides information on using switch views, stop views and the Hierarchy Editor in Dynamic Link.
- [Appendix A, Command Reference and Troubleshooting](#) describes the function of each of the menu selections provided in Advanced Design System (Cadence Menu), Cadence Schematic window (ArtistUtilities Menu), and the Cadence

CIW (ADS Menu) while using the RFIC Dynamic Link. Troubleshooting information that can help you resolve common problems is also provided in this appendix.

Chapter 2: Administrative Tasks

This chapter describes system requirements and how to install and configure the software. You may require help from a UNIX or EDA Administrator to complete these tasks.

System Requirements

This section describes the minimum hardware, operating system, EDA Framework and License requirements necessary for using the RFIC Dynamic Link.

Hardware Requirements

The information in [Table 2-1](#) describes the *minimum* hardware requirements for the RFIC Dynamic Link.

Table 2-1. Dynamic Link Minimum Hardware Requirements

Hardware	Requirement
RAM	256MB
Swap Space	500MB
Hard Disk Space	20MB of disk space for installation

Software Requirements

RFIC Dynamic Link version 2002 requires ADS 2002. Dynamic Link is supported by Cadence DFII versions 4.4.3QSR1, 4.4.5 and 4.4.6 and on all UNIX operating system versions from Hewlett Packard, Sun and IBM which run this Cadence software. Refer to [Table 2-2](#) for a summary of supported platforms. For additional information, please contact Cadence Design Systems Inc.

Table 2-2. Supported Platforms

Cadence DFII Version	AIX 4.4.3	HPUX 10	HPUX 11	SUN 56	SUN 57	SUN 58
4.4.3QSR1	X	X	X	X	X	
4.4.5	X		X	X	X	X
4.4.6	X		X		X	X

License Requirements

In addition to your standard Advanced Design System licenses, the following additional product licenses are required.

RFIC Dynamic Link License

- trans_idf (for ADS 2002)
- Idf_c_interface (previous ADS Dynamic Link Versions)

Cadence Licenses

- OASIS_Simulation_Interface
- 34510 - Affirma(TM) analog design environment
- 300 - Virtuoso(R) layout editor (if using layout)

Note You must purchase all required Cadence licenses from Cadence Design Systems.

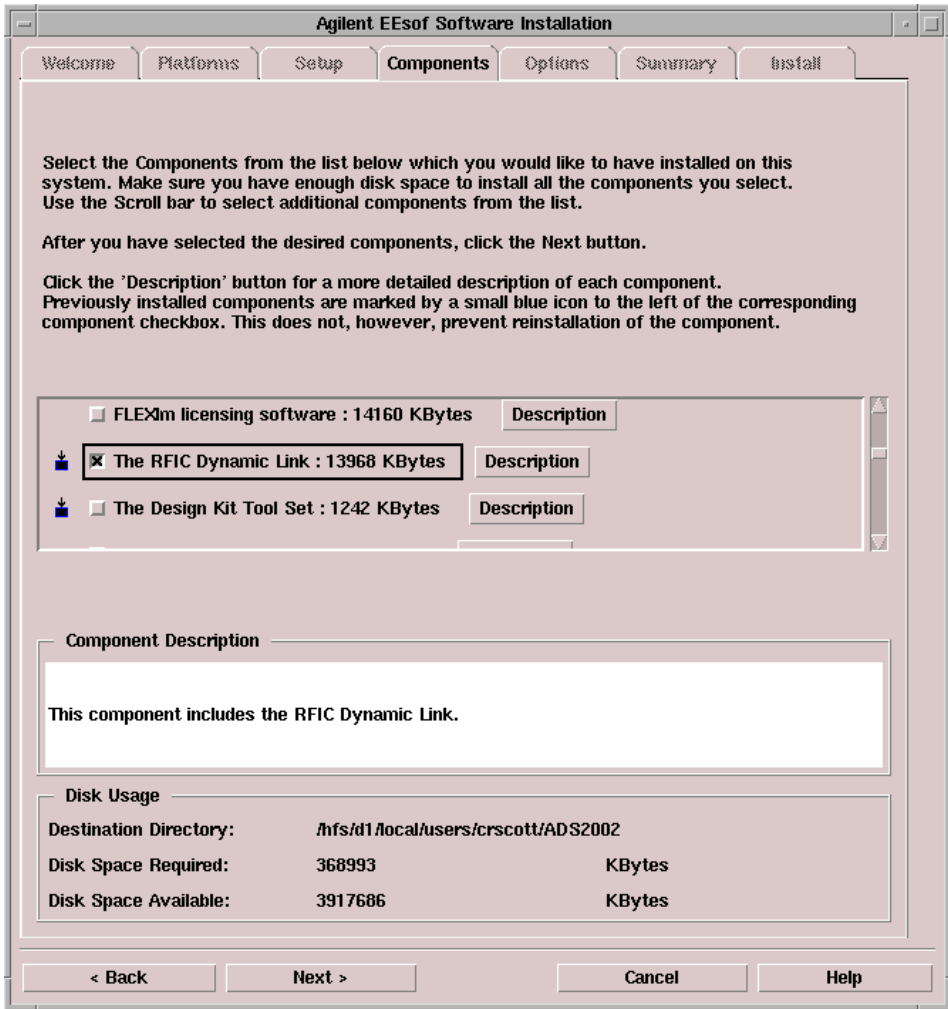
Installing RFIC Dynamic Link

The RFIC Dynamic Link installation procedure continues to be improved to make it easier for you to install and configure.

To install RFIC Dynamic Link:

1. Follow instructions in the ADS *“Installation on UNIX Systems”* manual to run the SETUP utility and load the *install* program.
2. After the *Agilent EEs of Installation Manager* starts, you are prompted to select one of the following installation options:
 - **Typical** - If you choose a *Typical* installation, the *RFIC Dynamic Link* will not be installed.
 - **Complete** - If you choose a *Complete* installation, the RFIC Dynamic Link will be automatically installed.
 - **Custom** - If you choose a *Custom* installation, you must select both *Simulators and Design Entry* and *The RFIC Dynamic Link* components from the scroll-down list in the *Agilent EEs of Software Installation* dialog box.

Note The *Simulators and Design Entry* component is the basic ADS software, including the Design Environment, Data Display, and Analog/RF Systems and Signal Processing simulators. This is a minimum requirement for RFIC Dynamic Link.



3. After the installation is complete, you will need to run the *idfConfigCadence* script to configure Cadence for use with RFIC Dynamic Link. This script is located under:

`$HPEESOF_DIR/bin/idfConfigCadence`

Note For ADS 2001, the Cadence configuration script was located at SHPEESOF_DIR/idf/config/configCadence.

For more information on the *idfConfigCadence* script, refer to [“Running the idfConfigCadence Script” on page 2-6](#).

For more information on installation procedures, refer to the ADS *“Installation on UNIX Systems”* manual.

Configuring the Cadence Install Directory

After the installation procedure (see [“Installing RFIC Dynamic Link” on page 2-3](#)), you will need to run the *idfConfigCadence* script. This script configures the Cadence install directory for use with ADS by adding or modifying five files.

Note You must have write access to your Cadence install directory.

The following four files are created:

```
<Cadence Installation Dir>/tools/dfII/etc/tools/ads/.cdsenv  
<Cadence Installation Dir>/share/cdssetup/hierEditor/ads  
<Cadence Installation Dir>/tools/dfII/etc/skill/hnl/ads.ile  
<Cadence Installation Dir>/tools/dfII/etc/skill/si/caplib/ads.ile
```

Note For Cadence versions 4.4.5 & 4.4.6, the second line above is replaced by:

```
<Cadence Installation Dir>/share/cdssetup/hierEditor/templates/ads
```

The fifth file, *<Cadence Installation Dir>/tools/dfII/etc/tools/auCore/.cdsenv*, is edited to add *ads* to the Tool Filter list of simulators.

Note The files described in this section are created or modified based on detailed instructions provided in the *Cadence OASIS Integrator's Guide*. The information in the Cadence manual describes how to integrate a simulator into *Cadence Analog Artist* using the Cadence *direct toolkit*.

Important While deviating from the information described in the *Cadence OASIS Integrator's Guide* may work for RIFC Dynamic Link, the files have been specifically set up to follow the Cadence documentation. Agilent Technologies does not support deviations from this information.

For more information, refer to:
 Cadence OASIS Integrator's Guide
 Version 4.4.5, December, 1999
 Pages 4-12, 4-13, 6-2 and 6-6.

Running the `idfConfigCadence` Script

Before running the `idfConfigCadence` script, set `HPEESOF_DIR` to your ADS installation directory and set your `PATH` to include Cadence software.

The `idfConfigCadence` command uses the following general syntax:

```
idfConfigCadence {-h | -ls | -rm}
```

Simply entering `idfConfigCadence` at the command line with no options runs the script to configure Cadence for RFIC Dyanmic Link. [Table 2-3](#) displays a list of the options and definitions used by the `idfConfigCadence` command.

Table 2-3. Option Definitions for `idfConfigCadence`

Option	Definition
-h	This option can be used to display the <code>idfConfigCadence</code> help file.
-ls	This option can be used to list the Cadence configuration files for ADS/RFIC Dynamic Link and indicate whether Cadence is ready for ADS RFIC Dynamic Link or not.
-rm	This option removes the Cadence configuration for ads/RFIC Dyanmic Link by deleting the files previously installed with the <code>idfConfigCadence</code> script and removing ads from the Cadence toolFilter. For more information on the Cadence Tool Filter, refer to your Cadence documentation.

Understanding the `idfConfigCadence` Script

When the `idfConfigCadence` script is executed without any options, the script performs the following operations:

- Copies

```
$HPEESOF_DIR/idf/skill/4.4.X/adsCdsenvFile
```

to

```
<Cadence Installation Dir>/tools/dfII/etc/tools/ads/.cdsenv
```

- Copies

```
$HPEESOF_DIR/idf/config/ads.hierEd
```

to

```
<Cadence Installation Dir>/share/cdssetup/hierEditor/ads
```

- Creates two new empty files called `ads.ile` in the following locations:

```
<Cadence Installation Dir>/tools/dfII/etc/skill/hnl/ads.ile
```

```
<Cadence Installation Dir>/tools/dfII/etc/skill/si/caplib/ads.ile
```

Configuring for a New Cadence Release

The `idfConfigCadence` script configures the Cadence installation directory for use with Dynamic Link. However, if a new version of the Cadence software is subsequently installed in a new directory, it will not have the Dynamic Link configuration. In this case, you would have to run the `idfConfigCadence` script as described in [“Configuring the Cadence Install Directory” on page 2-5](#).

Configuring the Software

This section describes the various aspects of configuring and/or modifying the software to provide additional flexibility.

Configuring the UNIX Environment

There are several UNIX environment variables relevant to Dynamic Link. These are described in the table below:

Table 2-4. UNIX Environment Variables

Environment Variable	Description
IDF_CONFIG_FILE	The name of the configuration file (only the file name, not the entire path). Default value is <i>idf.cfg</i> .
IDF_ADS_PROJ_DIR	If this environment variable is defined, ADS will attach to the specified project when RFIC Dyanmic Link launches ADS.
IDF_DEBUG_MODE	If set to TRUE debugging will be turned on and additional log messages will be written to the CIW; the log files <i>mps.log</i> and <i>emx.log</i> are also created. Default value is FALSE.
IDF_LOG_FILE	Name of the file (only the file name, not the entire path) to which ADS messages, normally written to <i>stderr</i> are redirected. Default value is <i>idf.log</i> .
PATH	The UNIX path variable.

Configuring the ADS Install Directory

The installation procedure configures the ADS install directory by automatically adding, modifying or replacing some files and/or directories.

The following files and/or directories are created, modified or replaced:

```

$HPEESOF_DIR/bin/idfmp
$HPEESOF_DIR/bin/idf
$HPEESOF_DIR/bin/idfConfigCadence
$HPEESOF_DIR/idf
$HPEESOF_DIR/idf/ael
$HPEESOF_DIR/idf/cdslib
$HPEESOF_DIR/idf/components
$HPEESOF_DIR/idf/config
$HPEESOF_DIR/idf/examples
$HPEESOF_DIR/idf/skill
$HPEESOF_DIR/idf/symbols
$HPEESOF_DIR/circuit/symbols/idfSymbol.dsn
$HPEESOF_DIR/circuit/config/ADSlibconfig
$HPEESOF_DIR/tools/lib/dpkg/info/idf*

```

Modifying the RFIC Dynamic Link-Cadence Initialization File

The RFIC Dynamic Link-Cadence initialization file (*.cdsinit*) is located in the directory *\$HPEESOF_DIR/idf/config/.cdsinit*. Append it to or load it from the first available *.cdsinit* file from the list below:

- *<Cadence Installation Dir>/tools/dfII/local/.cdsinit*
- *./cdsinit*
- *\$HOME/.cdsinit*

Modifying the Configuration File

Dynamic Link comes with the default configuration file *\$HPEESOF_DIR/idf/config/idf.cfg*. This file is used to set various site-specific or user-specific options. It is searched for and read sequentially from the following locations in the order given, so that settings in files read later override those of earlier files:

```
$HPEESOF_DIR/idf/config/  
$HOME/hpeesof/config/  
./
```

The name of the configuration file can be set via the UNIX environment variable *IDF_CONFIG_FILE*. By default the configuration file is named *idf.cfg*.

The configuration file consists of lines in the form *<parameter> = <value>*. The various parameters that can be set in the configuration file are listed below, with brief descriptions and an example for each. If no configuration file is found or some parameters are not set, internal default values are used. Note that wherever a file name is required for a configuration parameter value, it may be specified with a path prefix that is a UNIX environment variable value or using standard UNIX conventions such as *~*. A complete example configuration file can be found at the end of this section.

- **Model Path:** This is a space-separated list of directories that is searched in sequence until a model file for each implemented component is found. The default is *~/models*. Example:

```
IDF_MODEL_PATH = “~/models /usr/local/eda/cmos/models”
```

Note This applies to Cadence IC 4.4.3 only.

- **Switch View List:** If an instance has none of the views listed in the switch view list, the netlister reports an error. The default is *ads sch.model schematic*. Example:

```
IDF_SWITCH_VIEW_LIST = "ads sch.model schematic"
```

- **Stop View List:** The netlister identifies primitives with a stop list. When it reaches a view that is listed in both the switch view and stop view lists, the instance is netlisted and no expansion occurs below this level. There is normally no reason to change the Stop View list to anything other than *ads*. Example:

```
IDF_STOP_VIEW_LIST = "ads"
```

- **Project Path:** This indicates the path to the ADS project to attach to upon starting ADS via RFIC Dynamic Link. Example:

```
IDF_ADS_PROJ_DIR = /tmp/ads_prj
```

- **Netlist Filter:** When you have site customization that is not performed by the supplied netlister, this option enables you to specify the name of the program or script used to post-process the netlist generated by Analog Artist. The position of the netlist file name in your command string is indicated by %s. Example:

```
IDF_NETLIST_FILTER = "$HPEESOF_DIR/bin/myfilter %s"
```

- **Netlist Suffix:** This is the suffix for the Advanced Design System netlist file generated by Analog Artist for each sub-circuit. The default is *.net*. Example:

```
IDF_NETLIST_SUFFIX = ".net"
```

- **Model Suffix:** This is the suffix for each Advanced Design System model file in the specified model search path directories. The default is *.ads*. Example:

```
IDF_MODEL_SUFFIX = ".ads"
```

Note This applies to Cadence IC 4.4.3 only.

- **Debug Mode:** This option enables you to turn debugging messages on or off. These messages appear in order to help you determine the cause and/or location of problems.

By default, debugging is turned off. To enable debugging, set this option to *TRUE*. Example:

```
IDF_DEBUG_MODE = TRUE
```

- **Symbol Generation:** This option enables you to specify whether to generate a missing symbol using the Cadence symbol generator or the Advanced Design System symbol generator. The Cadence symbol generator is used as the default. Example:

```
IDF_CADENCE_SYMBOL = FALSE
```

- **User AEL Files:** Users can define their own AEL functions and load them into the ADS environment via a list of comma separated file names. These files get loaded just after the Dynamic Link environment is initialized. Example:

```
IDF_USER_AEL_FILES = "file1.ael, $HOME/file2.atf"
```

- **Expression Mapping:** This causes sub-strings in Cadence expressions to be mapped to corresponding sub-strings in ADS expressions in the netlist file and/or in the design variable values used. Example:

```
IDF_EXPR_MAP = "foo bar"
```

- **Freezing Subcircuit Netlists:** When this variable is set to *yes* (or TRUE), all Cadence subcircuits for which a netlist already exists are not re-netlisted. The default is *FALSE*. Example:

```
IDF_FREEZE_NETLISTS = TRUE
```

Note The Freeze parameter can be set to *yes*(TRUE) or *no* (FALSE).

To freeze selected subcircuits, see [“Freezing Selected Subcircuits” on page 9-10](#).

- **Message Timeout:** This specifies the timeout period in seconds for message actions initiated in ADS to complete in DFII. The default is 45. Example:

```
IDF_MSG_TIMEOUT = 120
```

- **Other Options:** The following special options should not be altered.

```
IDF_AEL_FILES = "globals.atf, utils.atf, commands.atf, callbacks.atf,  
symbol.atf, include.atf"
```

```
IDF_PDE_EXEC = hpeesofde
```

```
IDF_PDE_ARGS = "-env de_sim"
```

```
IDF_PRODUCT_NAME = idf
```

```
IDF_SCALE_FACTOR = 2.0
```

Example Configuration File

```
IDF_MODEL_PATH = "~/models $HOME/myModels"
IDF_SWITCH_VIEW_LIST = "ads sch.model schematic"
IDF_ADS_PROJ_DIR = "./examples_prj"
IDF_NETLIST_FILTER = "~/bin/mynetlistfilter"
IDF_NETLIST_SUFFIX = ".net"
IDF_MODEL_SUFFIX = ".ads"
```

Modifying the BindKey Settings

When using Composer, the Dynamic Link bindkey settings are inherited from the Cadence *Schematics* application, just as happens for the *analogArtist-Schematic* application. This is the default or preferred behavior however, if custom bindkeys are required:

1. Modify the `IdfSetBindKeys()` function accordingly. This function is provided in the file:

```
$HPEESOF_DIR/idf/skill/bindKeys.il
```

For example, if you would like to modify your key mappings to match the ADS key mappings because you are more familiar with ADS, modify the bindkeys as described above.

2. Edit the *ads.ini* file replacing the call to the function, `hInheritBindKey("IdfSchematic" "Schematics")` with a call to `IdfSetBindKeys()`. The *ads.ini* file is located in:

```
$HPEESOF_DIR/idf/skill/4.4.x/ads.ini
```

Managing Projects and Designs

Your Cadence designs will remain in their original locations. They are not copied, translated, or otherwise modified. When ADS starts up in Dynamic Link mode, it puts you in a project directory defined by the `IDF_ADS_PROJ_DIR` environment variable. If this variable is not defined, only the ADS Main window will open and no project will be attached.

To change this behavior, you can do one of the following:

- Specify your own startup project directory by defining `IDF_ADS_PROJ_DIR` in your UNIX environment or via the configuration file *idf.cfg*.

- After the Advanced Design System has come up, go to the Main window and open a new or existing project (**File > New** or **File > Open**).

For more information on projects and design files, refer to “*Managing Projects and Designs*” in the *ADS User’s Guide*.

Chapter 3: Getting Started Tutorial

This tutorial steps you through the process of simulating a circuit using components from the Dynamic Link *analogLib* library. Other examples are also included to help you become familiar with the product.

Both the drop-down menus and icons are described to help familiarize you with the Advanced Design System environment.

Setting up the Examples Directory

From any directory of your choice, enter:

```
cp -r $HPEESOF_DIR/idf/examples ./
cd examples
```

Note This must be done before attempting the Getting Started Tutorial. The *cds.lib* file under the examples directory defines libraries provided by Dynamic Link. The *.cdsinit* file under this directory loads the Dynamic Link *.cdsinit* file which then loads the context files required to run Dynamic Link.

Starting the Cadence Design Framework

Ensure that you are in the examples directory then open Cadence by typing the appropriate command (typically *icms* or *msfb*). The Cadence *Command Interpreter Window* (CIW) appears.

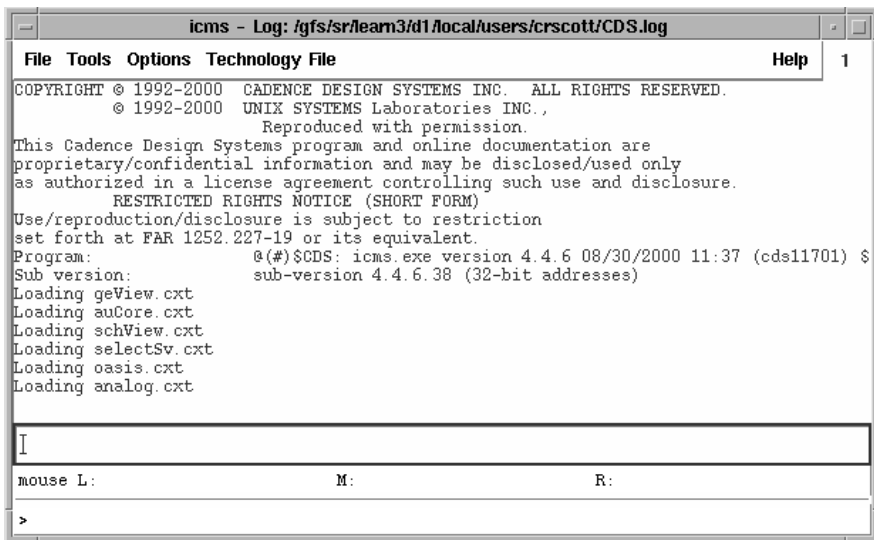


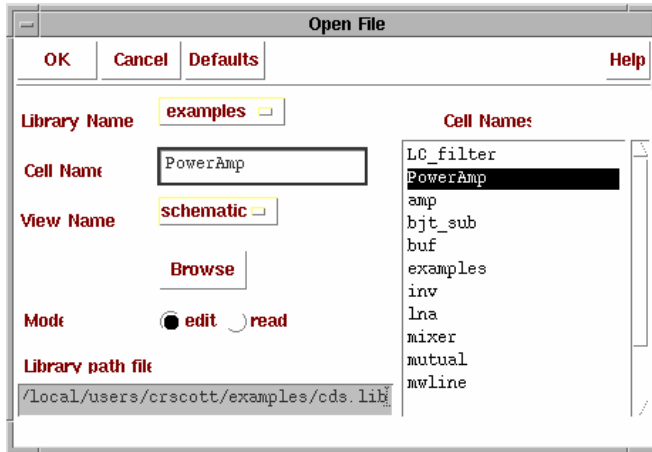
Figure 3-1. Cadence CIW Window

Note Non-standard or customized start-up scripts for Cadence Design Framework II may not be supported. If you have difficulties, contact your system administrator.

Opening a Cadence Composer Schematic

To open a schematic in Cadence Composer:

1. Choose **File > Open** from the Cadence CIW. The Open File dialog box appears.



2. Select *examples* from the Library Name drop-down list.
3. Click *PowerAmp* in the Cell Names list. This sets the Cell Name field to *PowerAmp*.
4. Select *schematic* from the View Name drop-down list if not already selected.
5. Select the *edit* Mode if not already selected.
6. Click **OK**. The Cadence *examples, PowerAmp* schematic appears.

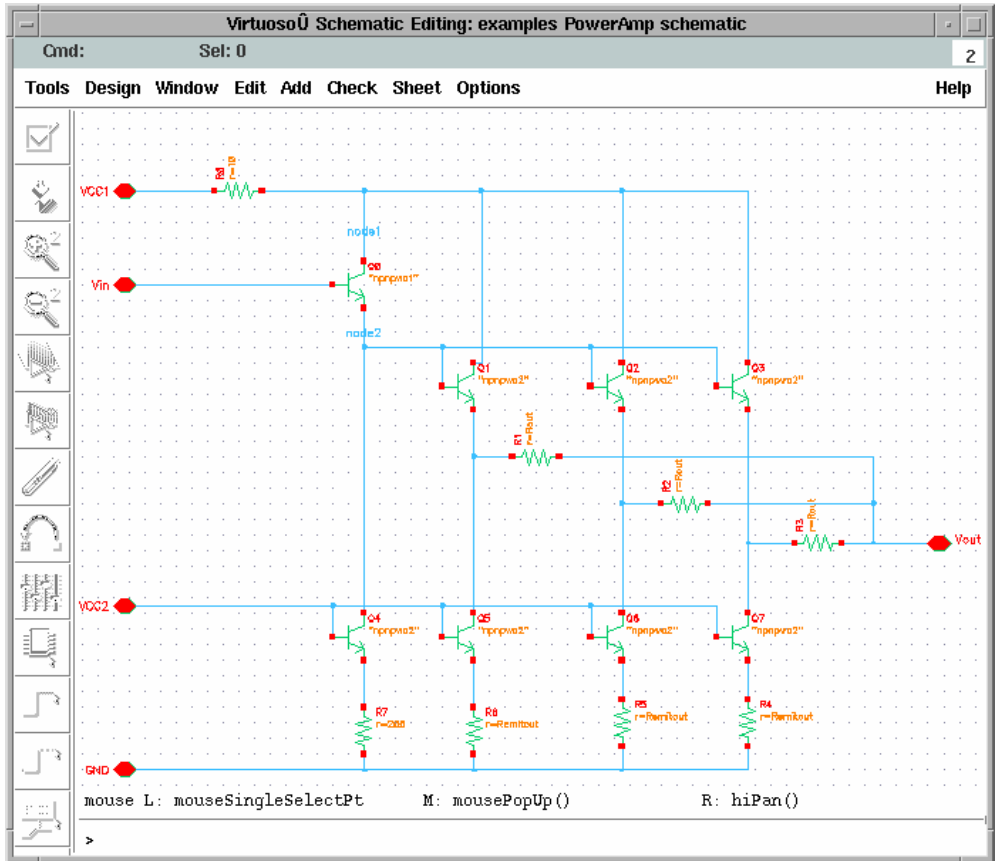


Figure 3-2. Cadence Virtuoso Schematic Composer Window

Linking with Advanced Design System

To link the Cadence design environment to Advanced Design System:

1. Choose **Tools > ADS** from the menu bar in the Cadence Schematic window. In a few moments, the Advanced Design System Main window appears in the upper-left hand corner of your display. This is followed by an empty ADS Schematic window to the right of the Main window.

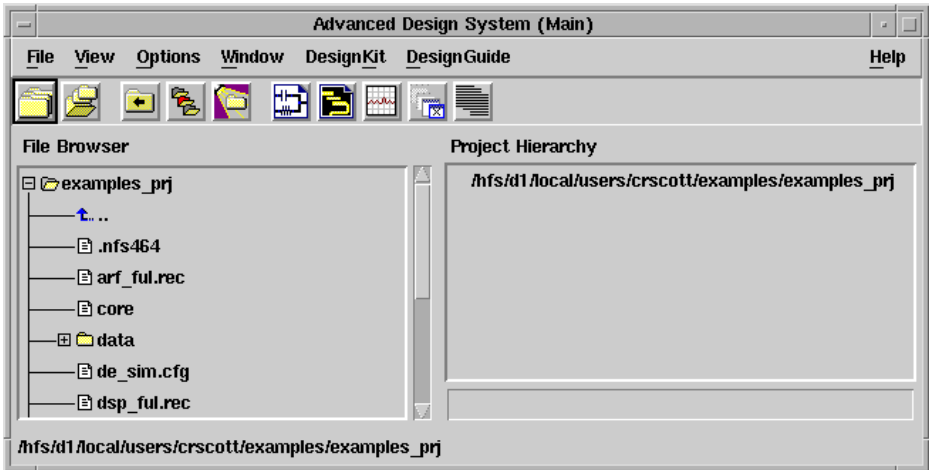


Figure 3-3. Advanced Design System Main Window

Note Depending on your system, it may take a few moments for the ADS windows to appear. View the Cadence CIW window for the link status.

The ADS Schematic window should display a **Cadence** menu item and is automatically titled *untitled1* (see [Figure 3-4](#)).

ADS Schematic Title

Cadence Menu Item

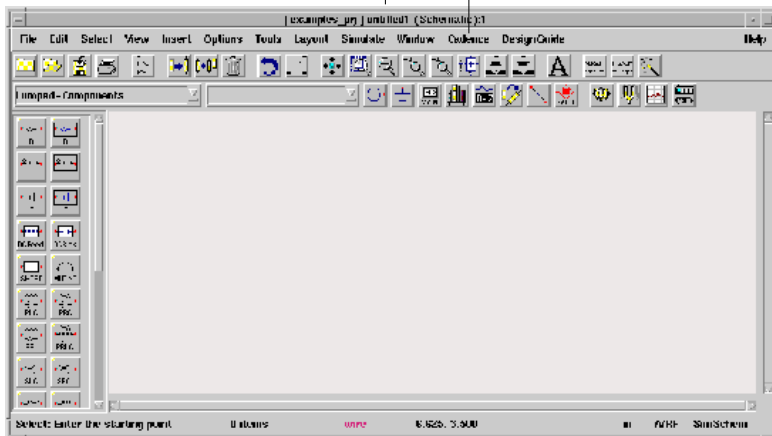


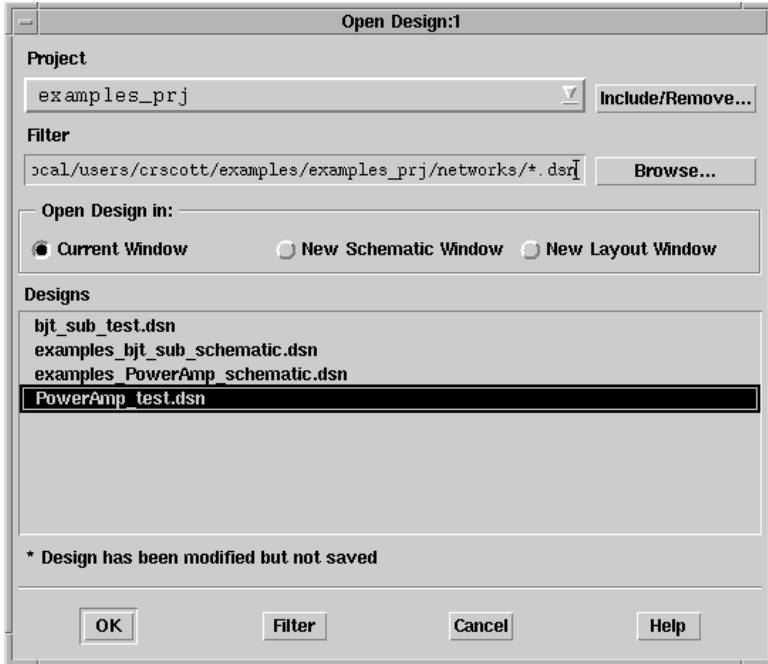
Figure 3-4. Advanced Design System Schematic Window

At this point, ADS and the Cadence Design Framework II are working together.

Opening a Test Schematic Design

To open a test schematic design:

1. Choose **File > Open Design** in the ADS Schematic window to display the Open Design dialog box. Use this dialog box to select the design you wish to simulate.



2. The *examples_prj* selection in the Project drop-down list is set by default in the Open Design dialog box.
3. Select *PowerAmp_test.dsn* from the Designs list. The *PowerAmp_test.dsn* schematic contains simulation components that can be selectively activated or deactivated.
4. Click **OK** to include the *PowerAmp_test.dsn* schematic in the ADS Schematic window.

Adding a Symbol of the Cadence Cellview

To add a symbol of the Cadence cellview in the Advanced Design System Schematic window:

1. Choose **Cadence > Instance > Add Instance of Cellview** in the ADS Schematic window.

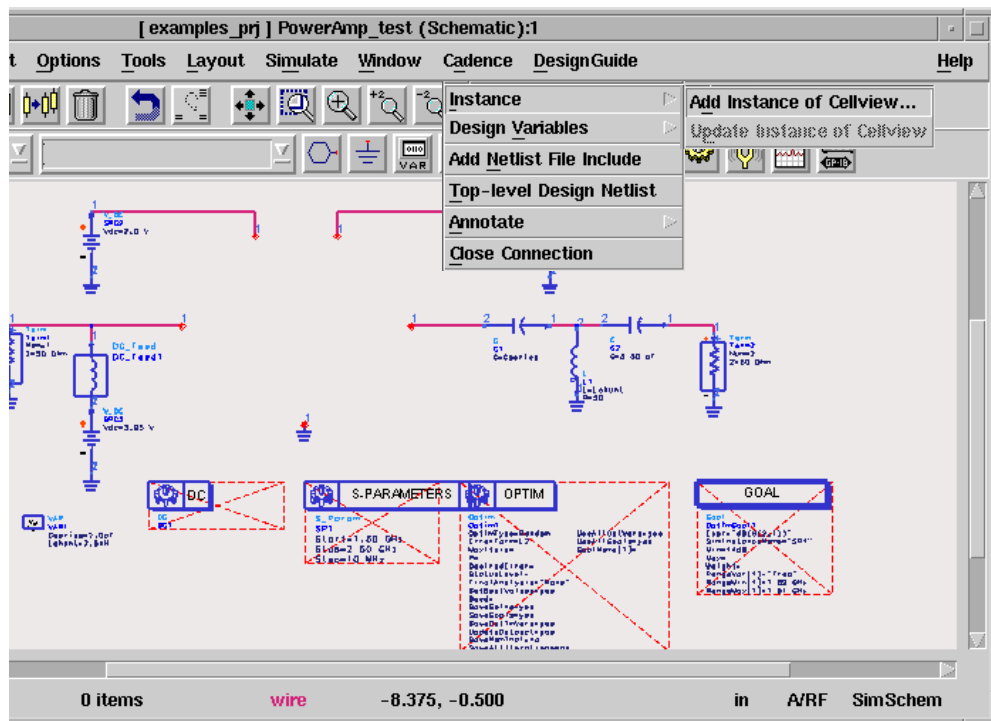
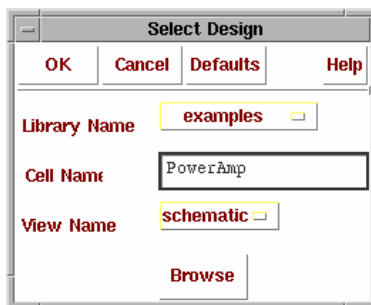


Figure 3-5. Adding a Cellview

A Select Design dialog box appears, enabling you to select the Cadence Cellview to simulate.



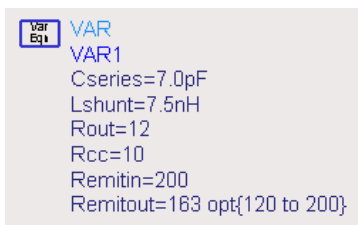
2. In the *Cell Name* field of this dialog, verify the entry or type the name of the Cellview you want to simulate (in this case *PowerAmp*). Alternatively, you can use the *Browse* button and library manager to select the name.
3. Click **OK**. A symbol of your Cadence Cellview is automatically generated.
4. An instance of the symbol is attached to the cursor for you to place. In the Advanced Design System Schematic window, click the left mouse button to place the symbol as desired.
5. You may continue placing more instances of the same cellview, or, in this case, choose the *Cancel Command And Return To Select Mode* icon to proceed with the next step. Similarly, you may place instances of other Cadence designs.



Adding Design Variables

To add the design variables from the Cadence Cellview to the ADS schematic window choose **Cadence > Design Variables > Get Design Variables**.

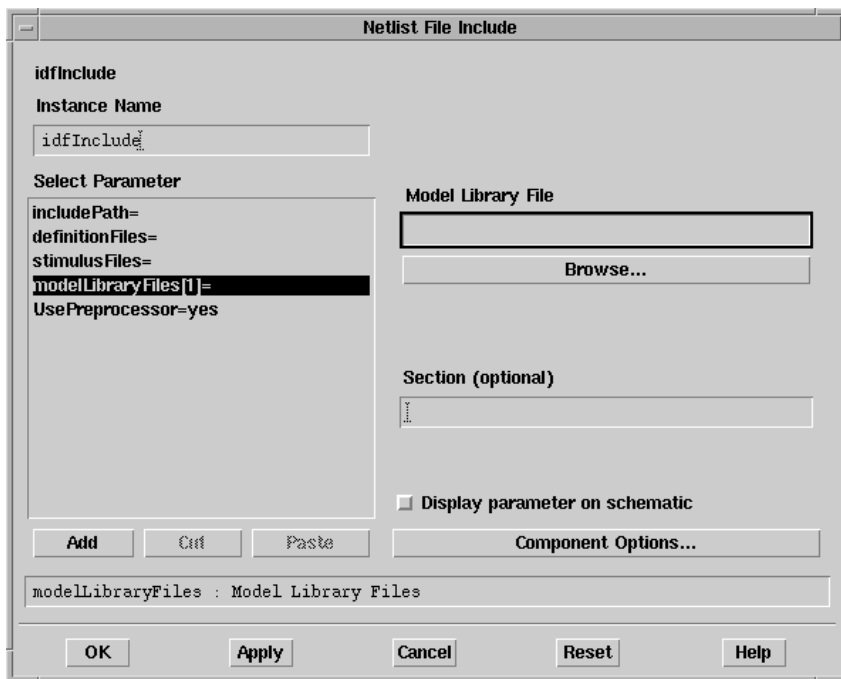
This places a corresponding *VarEqn* component on the ADS schematic containing the design variables from Cadence (i.e. Rcc, Rout, Remitin and Remitout).



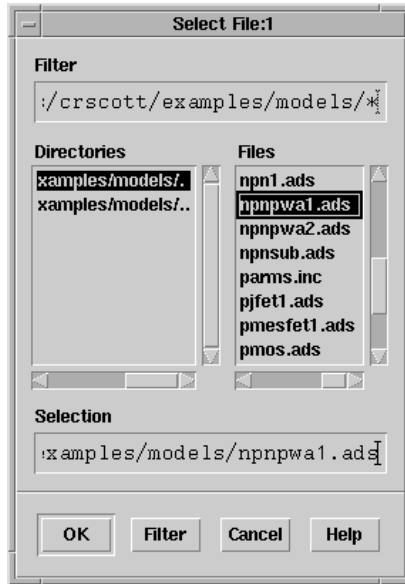
Adding Model Files

To add a model file

1. Choose **Cadence > Add Netlist File Include**
2. Place the *Netlist File Include* component in an open area on the schematic.
3. Double click the include component icon. The *Netlist File Include* dialog box appears.



4. In the Netlist File Include dialog box, click **modelLibraryFiles** in the Select Parameter list box. The *Model Library File* field appears in the dialog box.
5. Click **Browse** just below the *Model Library File* field to locate the first model library file. The Select File dialog box appears.

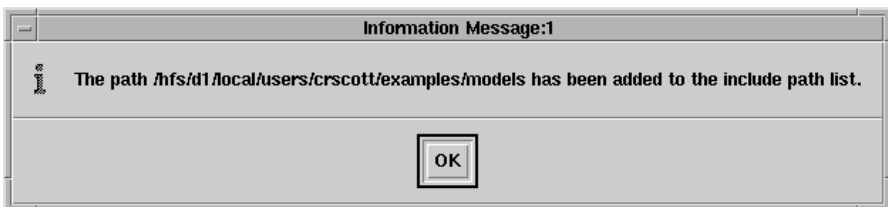


6. In the Select File dialog box, use the *Directories* field to locate the *models* directory.

`<your_current_working_dir>/examples/models`

This sets the path for the location of the model library files.

7. In the Select File dialog box, use the *Files* field to locate and click the **nnpwa1.ads** model file, then click **OK**. An information message appears stating that a new path has been added to the include path list.



Click **OK** in the *Information Message* dialog box. You are returned to the Netlist File Include dialog box.

8. In the Netlist File Include dialog box, notice that the Model Library File field now contains the *nnpwa1.ads* model file. Click **Apply** to add the *nnpwa1.ads* model file.
9. Click **Browse** again to locate the second model library file. The Select File dialog box appears.
10. In the Select File dialog box, use the *Files* field to locate and click the **nnpwa2.ads** model file, then click **OK**. You are returned to the Netlist File Include dialog box.
11. In the Netlist File Include dialog box, notice that the Model Library File field now contains the *nnpwa2.ads* model file. Click **Add** to add the *nnpwa2.ads* model file.
12. The *Select Parameter* field should now contain the information below.

```
modelLibraryFiles[1]=nnpwa1.ads  
modelLibraryFiles[2]=nnpwa2.ads.
```
13. Click **OK** in the Netlist File Include dialog box.

For more information on the Netlist File Include Component, refer to [“Using the Netlist File Include Component” on page 9-1](#).

Performing a DC Simulation

To run a DC simulation on an ADS schematic and then annotate the results to the Cadence Composer Window:

1. Choose **Edit > Component > Activate** then click on the DC component in the ADS Schematic window or choose the *Activate Components* icon to activate the DC component.

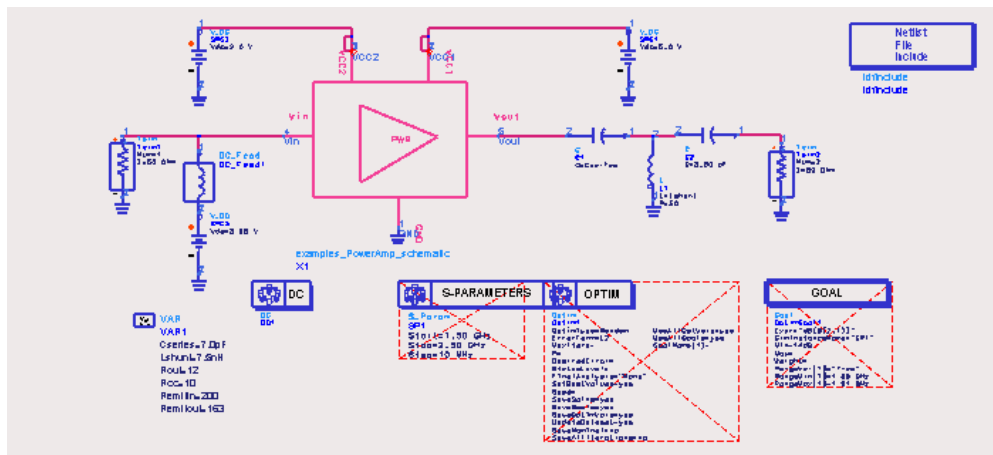
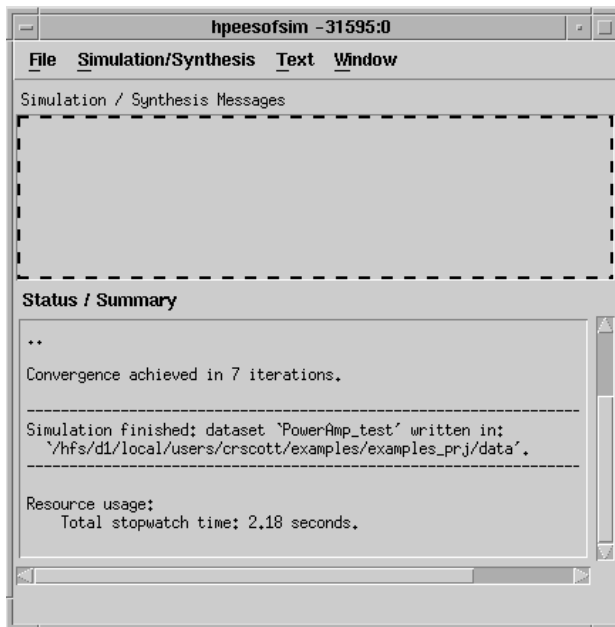


Figure 3-6. Activating Components in an ADS Schematic Window

2. Choose **Simulate > Simulate** or choose the *Simulate* icon to run a simulation. A simulation dialog box appears in your display.



3. After the simulation is complete, a Data Display window titled *PowerAmp_test* automatically appears. Close this window using the **File > Close Window** menu option.
4. Click the *PowerAmp* schematic symbol in the ADS Schematic window.
5. Choose **Cadence > Annotate > Annotate DC Solution to Selected Cellview**. This displays the DC node voltages on the Cadence schematic.

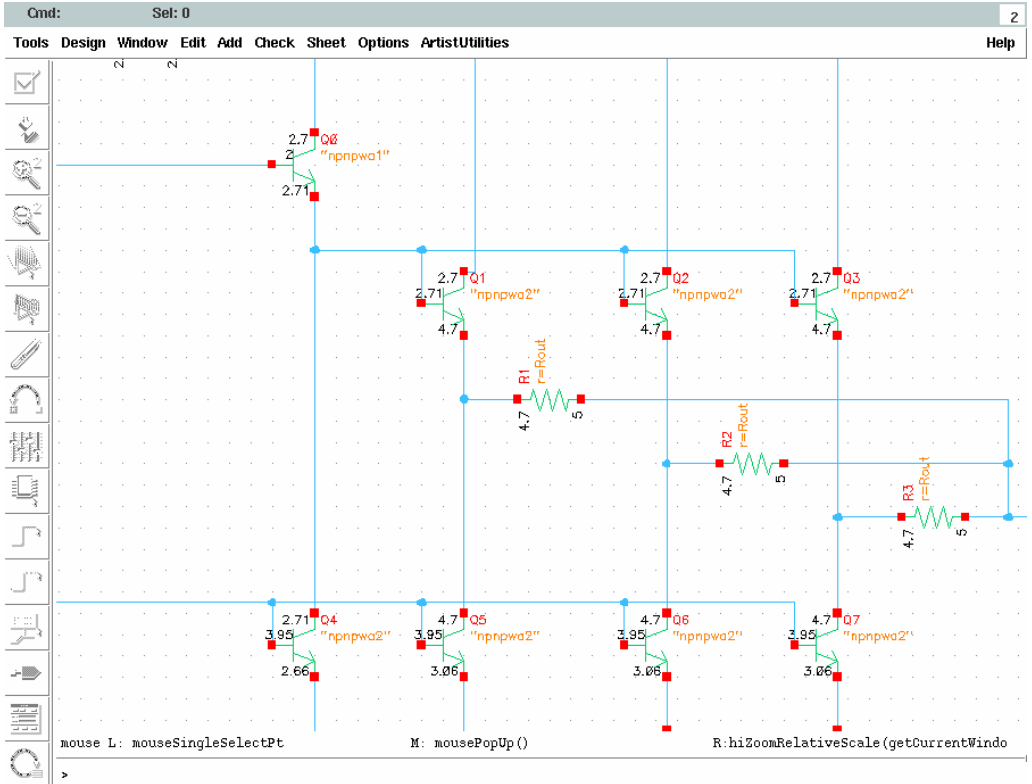


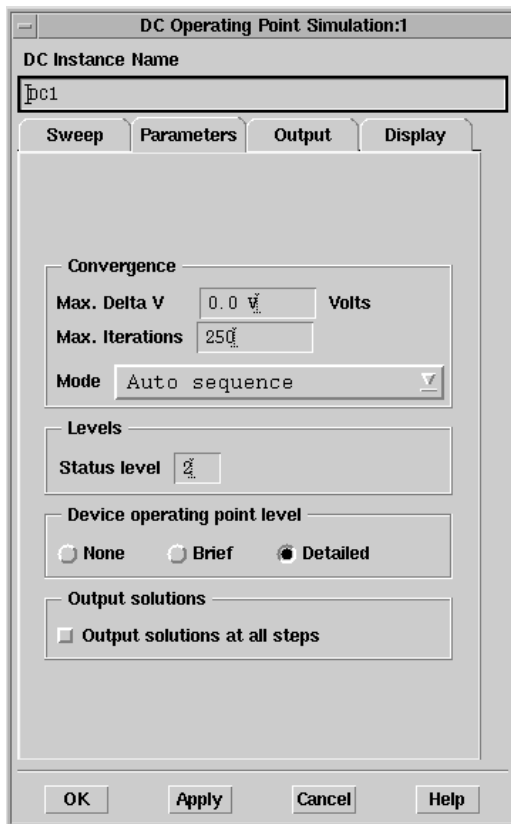
Figure 3-7. Cadence Schematic with Annotated Display

Annotating DC Operating Points to a Selected Cellview

Any simulation that includes a DC analysis produces DC operating point information for most active and some passive devices in the circuit. This data includes currents, power, voltages, and linearized device parameters of the selected device.


To run a DC Operating Point Simulation on an ADS schematic and then annotate the results to the Cadence Composer Window:

1. Double-click the DC Simulation component in the ADS schematic window. The DC Operating Point Simulation dialog box appears.



2. Click the *Parameters* tab in the DC Operating Point Simulation dialog box.
3. Click **Detailed** in the *Device operating point level* section of the DC Operating Point Simulation dialog box.

Note For a subset of the detailed DC Operating Point Simulation information that covers most common parameters, click **Brief** in the *Device operating point level* section of the DC Operating Point Simulation dialog box.

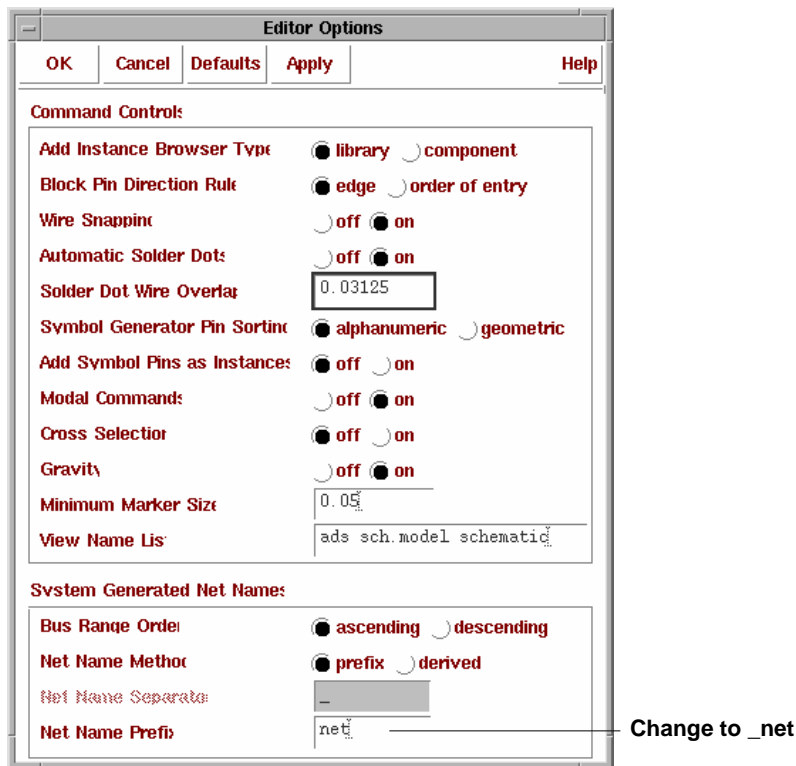
4. Choose **Simulate > Simulate** or choose the *Simulate* icon to run a simulation.  A simulation dialog box appears in your display.
5. After the simulation is complete, a Data Display window titled *PowerAmp_test* automatically appears. Close this window using the **File > Close Window** menu option.
6. Click the *PowerAmp* schematic symbol in the ADS Schematic window.
7. Choose **Cadence > Annotate > Annotate Operating Points to Selected Cellview**. This displays the DC operating point values for each component on the Cadence schematic.

Node Probing with RFIC Dynamic Link

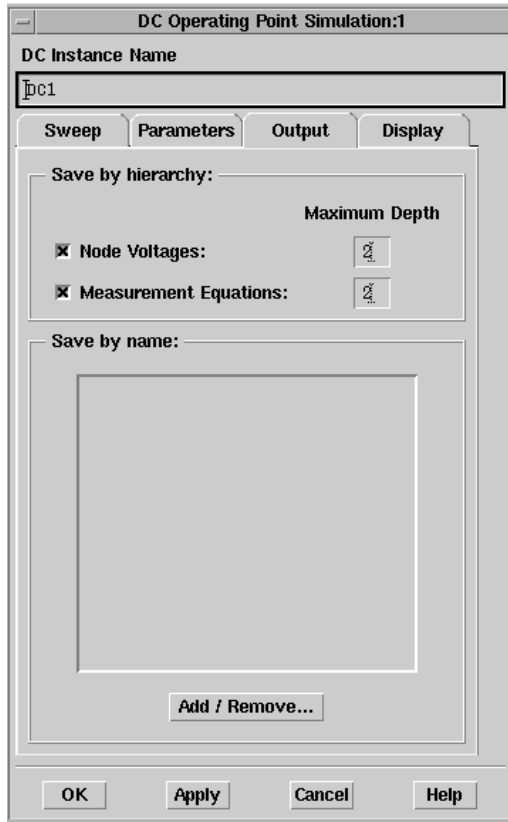
With the RFIC Dynamic Link Node-Probing feature, you can select nodes of interest in a Cadence subcircuit and display their voltages in an ADS Data Display window. ADS saves voltage data for all the named nodes. Any node with a node name prefix different from *_net* is considered a named node. For detailed information on the ADS Node-Probing feature, refer to the ADS “*Circuit Simulation*” manual.

The following is an example session of using the Dynamic Link Node-Probing feature with the PowerAmp example.

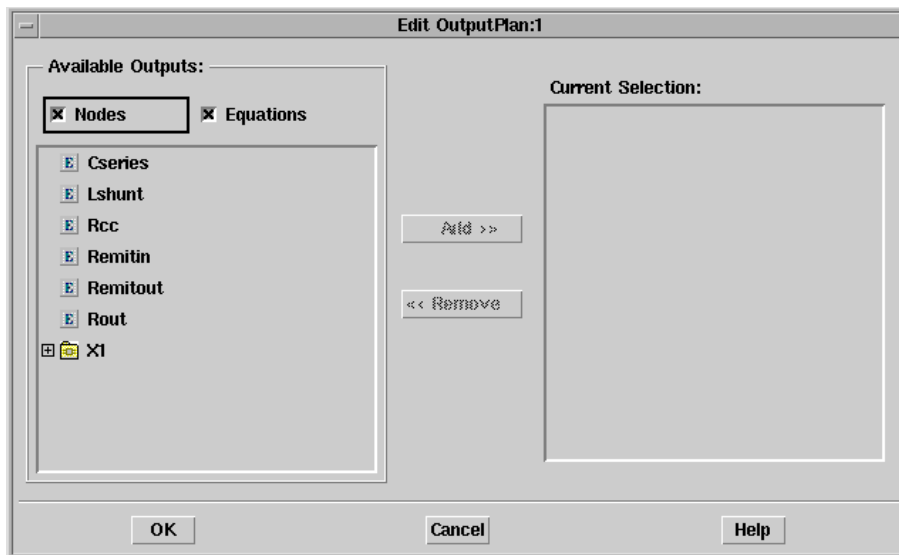
1. In the Cadence schematic window, choose **Options > Editors**. The Cadence *Editor Options* form appears.



2. Change Net Name Prefix field from *net* to *_net* then click **OK**. This prevents ADS from saving data for all the nodes in the PowerAmp cellview.
3. Click **Design > Check and Save** in the Cadence schematic window to ensure a new ADS netlist for the *PowerAmp* cellview is generated the next time a simulation is performed for the ADS *PowerAmp_test* design.
4. Double click the DC Simulation controller component in the ADS schematic window to open the *DC Operating Point Simulation* dialog box.



5. Click the **Output** tab in the *DC Operating Point Simulation* dialog box.
6. Click **Add/Remove** in the *Output* section of the *DC Operating Point Simulation* dialog box to open the *Edit OutputPlan* dialog box.



7. Click the + sign to the left of the X1 component in the *Available Outputs* field to start Dynamic Link Node Probing setup. This raises the Cadence schematic window containing the PowerAmp cellview. The prompt in the Cadence schematic window is changed to **ADS Node Probing Setup: Click a wire>**.
8. Click the wire connecting to the collector of the Q0 transistor in the upper-left region of the schematic window. A *node1* label is added to that wire in the Cadence schematic window. Meanwhile, *node1* also appears in the *Current Selection* field on the right side of the ADS *Edit OutputPlan* dialog box.
9. Click the wire connecting to the emitter of Q0 to add another node, *node2*, to the ADS *Edit OutputPlan* dialog box for DC Simulation. The Q0 transistor in the upper-left region of the Cadence schematic window should now appear similar to [Figure 3-8](#).

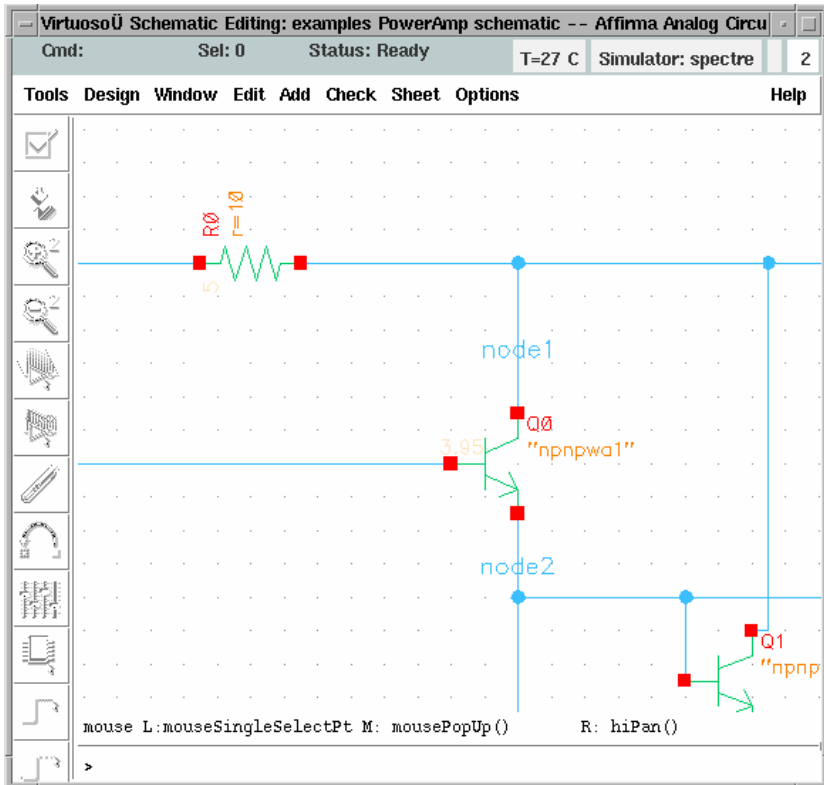


Figure 3-8. Q0 with Node Labels

10. Click **OK** in the *ADS Edit OutputPlan* dialog box to close the dialog box. The *DC Operating Point Simulation* dialog box should now appear similar to [Figure 3-9](#).

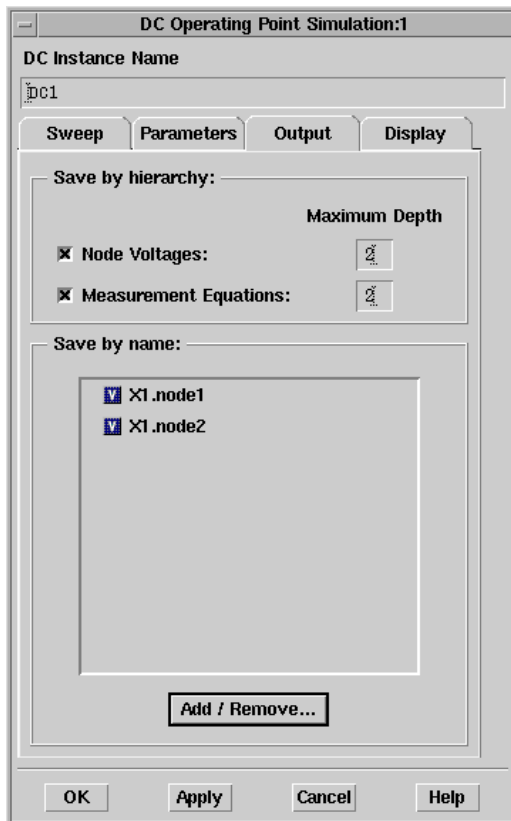


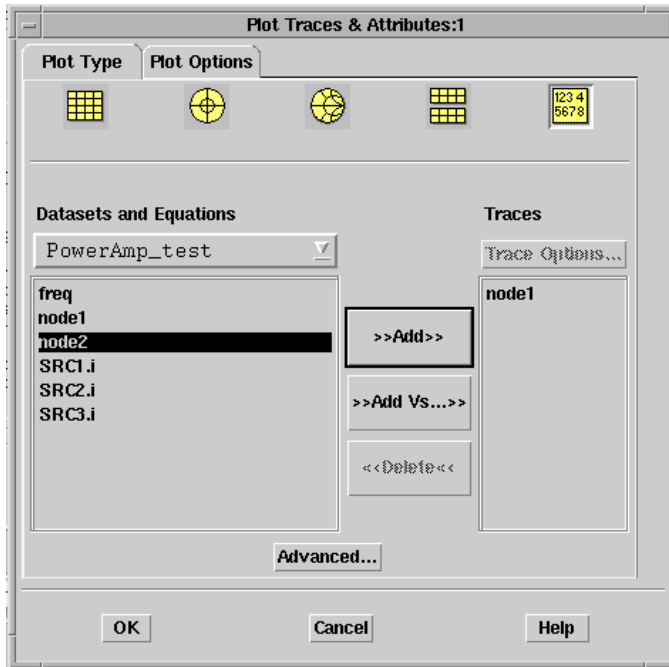
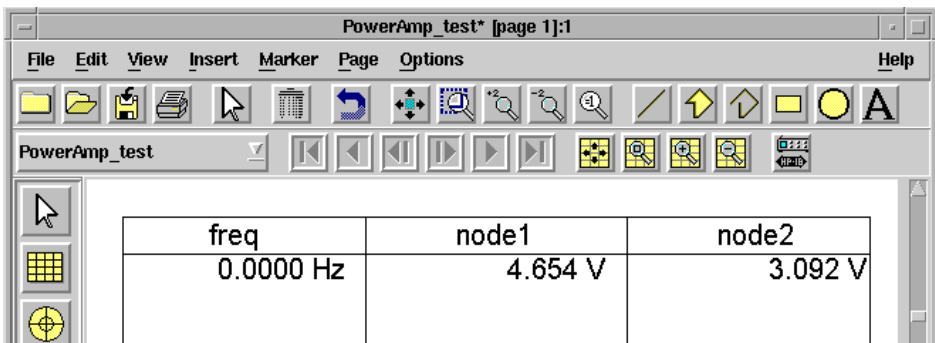


Figure 3-9. DC Operating Point Simulation Dialog Box

11. Click **OK** in the *DC Operating Point Simulation* dialog box to instruct ADS to save voltage values for **X1.node1** and **X2.node2** after DC simulation.
12. Click the Simulate icon or select **Simulate > Simulate** menu item in the ADS schematic window to run a DC Simulation. The ADS Data Display window appears automatically once the ADS DC Simulation is complete. 
13. Click the *List* icon on the left of the Data Display window then click a desired location in the ADS Data Display window to bring up the Plot Traces & Attributes dialog box. 



14. In the *Plot Traces & Attributes* dialog box, click *node1* on the left, then click **>>Add>>** in the middle to add *node1* to the *Traces* field on the right. Click *node2* and then click **>>Add>>** to add *node2*.
15. Click **OK** to close the *Plot Traces & Attributes* dialog box. The voltages of *node1* and *node2* are immediately listed in the ADS Data Display window.



Additional Notes:

- When the **Add/Remove** button in the *DC Operating Point Simulation* dialog box is clicked to bring up the *Edit OutputPlan* dialog box, as described in step 6 above, all of the named nodes and subcircuit components at current circuit level appear in the *Available Outputs* field on the left. You can select a named node from the *Available Outputs* field and then click **Add >>** to add it to the *Current Selection* field on the right. You can also remove a named node from the *Current Selection* field by clicking **<< Remove** after selecting it.
- Clicking the + sign to the left of a subcircuit component in the *Available Outputs* field displays all the named nodes and subcircuit components within that subcircuit. In step 7 above, nothing was displayed initially because there was no named node or subcircuit in the PowerAmp cellview.
- After a Cadence schematic window is raised as a result of clicking the + sign mentioned above, any named or unnamed node in the Cadence subcircuit hierarchy can be selected by clicking the left mouse button at any point on the wire. To select a node in Cadence subcircuit, choose **Design > Hierarchy > Descend Edit** menu item in the Cadence schematic window, click **OK** in the form popped up, and then click a wire of interest. Valid selections are limited to the current window and the current circuit hierarchy.
- Expanding a Cadence circuit component in the *Edit OutputPlan* dialog box for *Scattering-Parameter Simulation* will also raise a Cadence schematic window. Any unnamed wire selected will also have a label attached, but this node will not be added to the *Current Selections* field in the *Edit OutputPlan* dialog box because no DC voltage data will be available for S-Parameter simulation.
- During the Node Probing operation, the left mouse button in the Cadence Virtuoso schematic window is mapped to a Dynamic Link SKILL procedure. Do not bind any function to the left mouse button during this period. Any bindkey function previously mapped to the left mouse button will not work until the tune mode ends.

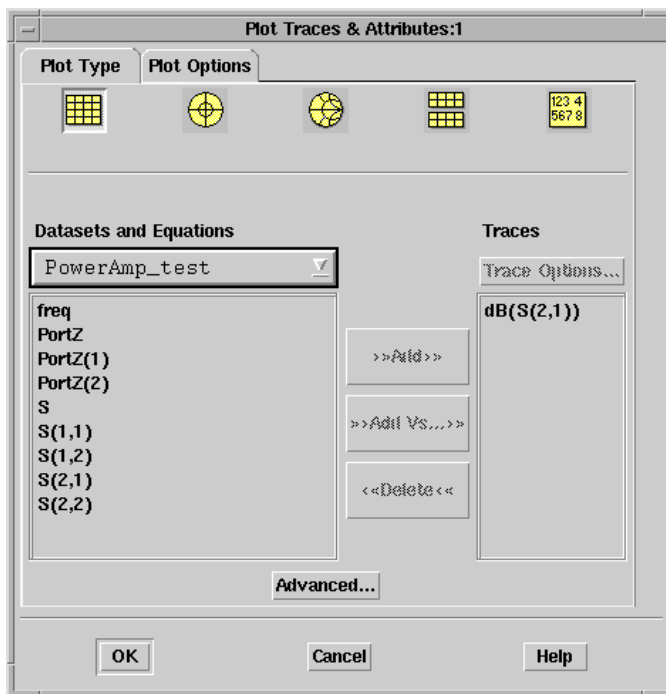
Performing an S-parameter Simulation

To perform an S-parameter Simulation on an ADS schematic:

1. Choose **Edit > Component > Deactivate** then place the cross hair over the DC component and click or choose the *Deactivate Components* icon.



1. Select *PowerAmp_test* from the Default Dataset drop-down list if not already selected.
2. From the Data Display window, choose **Insert > Plot**, move the frame to an appropriate location within the window and click. This anchors a frame for your plot. Similarly, you can choose the *Rectangular Plot* icon to drag and drop the plot frame.
3. The *Plot Traces & Attributes* dialog box appears. Select *S(2,1)* and then click **Add**. Select *dB* from the dialog box then click **OK**.



4. Click **OK** again to view the *Data Display*.

Note Figure 3-11 shows the forward gain, $S(2,1)$, at 1.9 GHz to be approximately 12.57 dB. By varying the value of *Remitout* in this case, you can modify the circuit to achieve the desired results. After you have completed the tutorial, take some time to experiment with different values of *Remitout*.

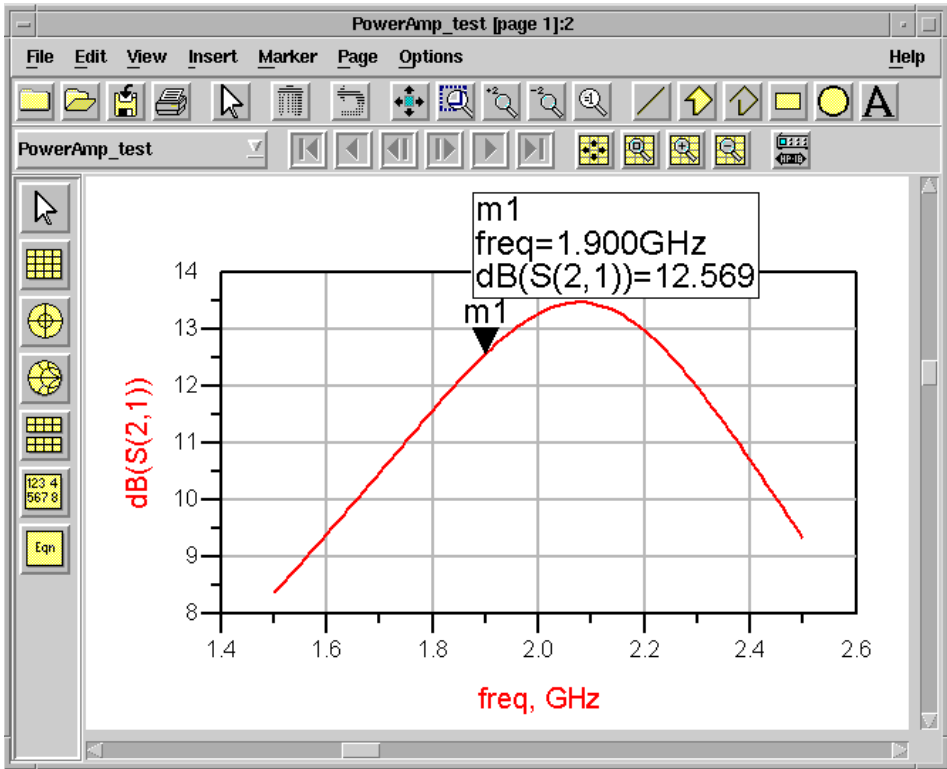


Figure 3-11. Data Display with S-parameter Simulation Results

Performing a Parameter Optimization

To optimize the parameters of *Var1* in the ADS schematic:

1. Choose the *Activate Components* icon then place the cross hair over the *Nominal Optimization* component. Repeat this step for the *Goal* component. The ADS schematic is now ready to optimize the S-parameter.

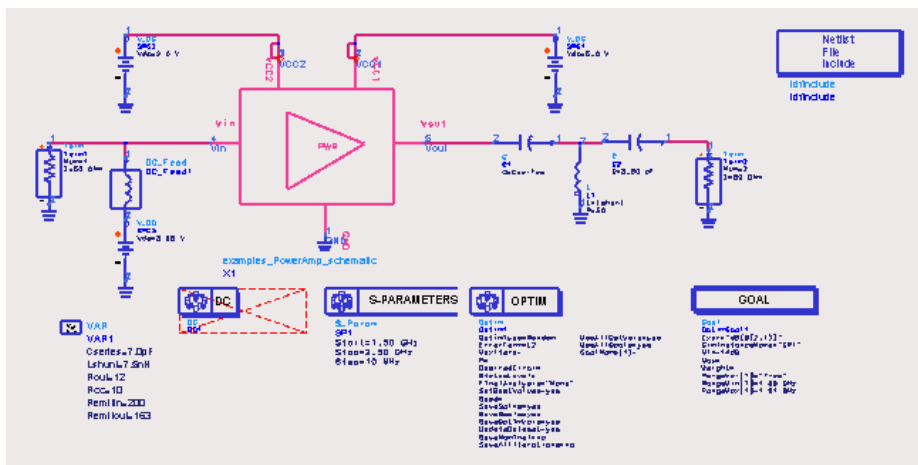
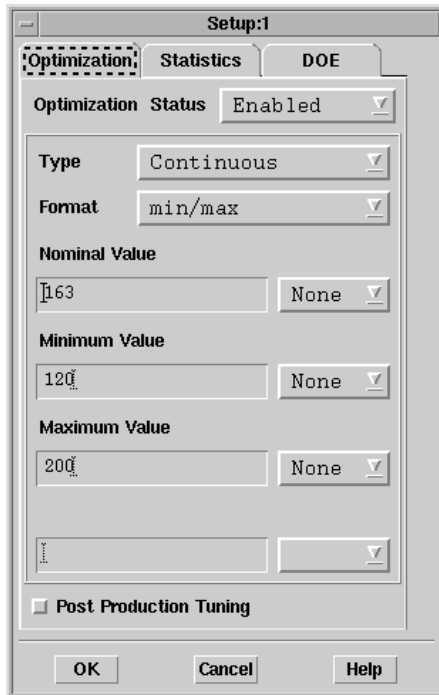



Figure 3-12. Activating Components in an ADS Schematic Window

2. Choose **Edit > Component > Edit Component Parameters** and then click the *VarEqn* component. The *Variables and equations* dialog box appears.
3. In the *Variables and equations* dialog box, select the parameter *Reemitout* in the *Select Parameter* field.
4. Click **Optimization/Statistics Setup**. The Setup dialog box appears.
5. Click the **Optimization** tab and set the **Optimization status** to *Enabled*.
6. Set the **Minimum Value** to *120* and the **Maximum Value** to *200*.



- Click **OK** twice, once in the *Setup* dialog box and once in the *Variables and equations* dialog box. Note that the *Remitout* parameter on the ADS schematic now displays:

Remitout= 163 opt{120 to 200}

- Choose **Simulate > Simulate** or choose the *Simulate* icon. This netlists each Cadence subcircuit in the *Affirma Analog Circuit Design Environment* (Analog Artist in 4.4.3), as well as the top-level ADS schematic, and starts the Advanced Design System simulator. 

Note A simulation status window appears, reporting the status of the simulation; depending on your system, this may take some time. Check this status window to see if any errors occurred during netlisting or simulation.

After the simulation is complete, a Data Display window titled *PowerAmp_test* automatically appears. For information on configuring the Data Display, refer to “[Displaying Your Results](#)” on page 3-25.

Note Figure 3-13 shows the optimized forward gain, $S(2,1)$, at 1.9 GHz to be approximately 13.08dB as set by the *Goal* component in the ADS Schematic window.

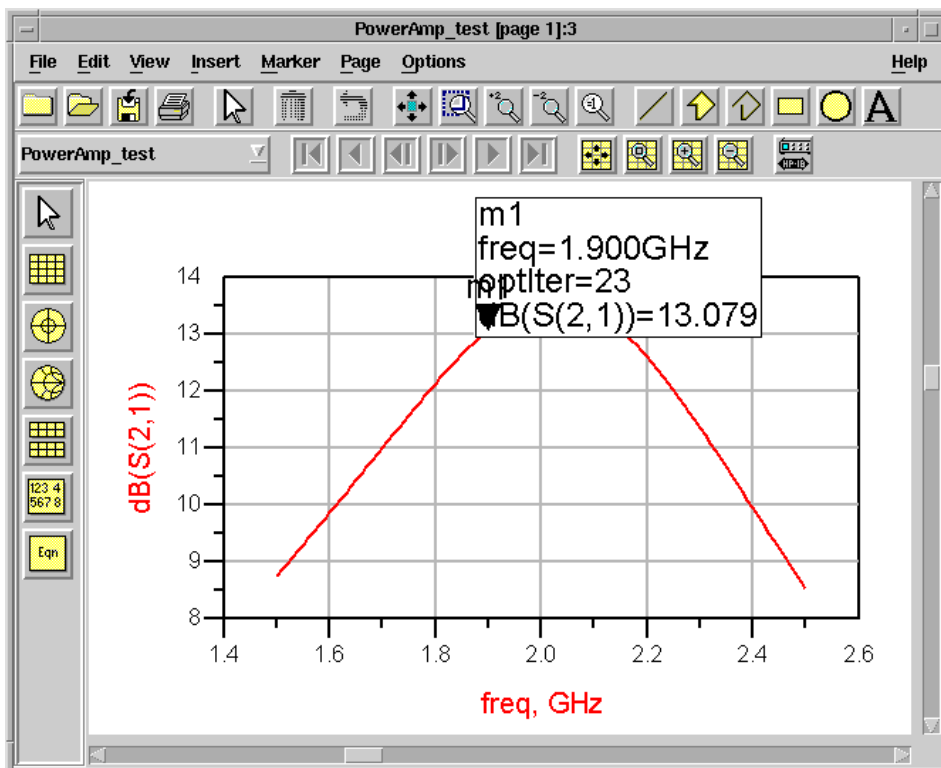
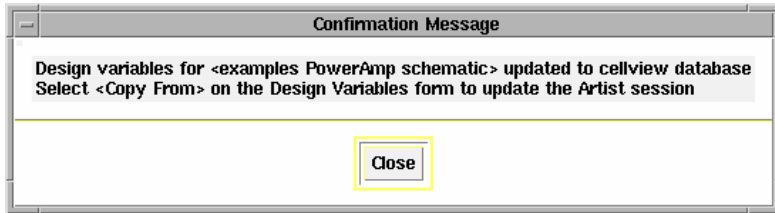


Figure 3-13. Data Display with Optimized Simulation Results

9. Choose **Simulate > Update Optimized Values** to update the optimized values. This changes the value of *Remitout* in the *VarEqn* component to the optimized value.

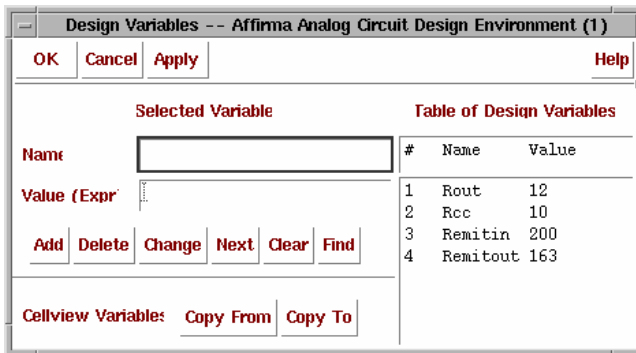
10. Choose **Cadence > Design Variables > Update Design Variables to Cellviews** to update the optimized value to the Cadence cellview. A *Confirmation Message* dialog box appears.



11. Click the **Close** button.
12. From the Cadence menu bar, choose **ArtistUtilities > Design Variables**. A *Design Variables* form appears.

Note If the *ArtistUtilities* pull-down menu does not appear in the Cadence Virtuoso Schematic window, choose **Tools > ADS** in the Cadence schematic window.

13. Click **Copy From** in the *Cellview Variables* section. This enables you to update the design variables to the Artist session.



Verifying Your Results

You may now verify the results of your optimization by using the optimized value of *Remitout* in an S-parameter simulation. Set the value of *Remitout* = 120 (approximate optimum value) and repeat the steps in “[Performing an S-parameter Simulation](#)” on page 3-24 with the new value of *Remitout*.

Note Figure 3-14 again shows the optimized forward gain, $S(2,1)$, at 1.9 GHz to be approximately 13.08 dB as set by the *Goal* component in the ADS Schematic window. By using the optimized value of *Remitout*, you have verified the optimum desired results of the circuit.

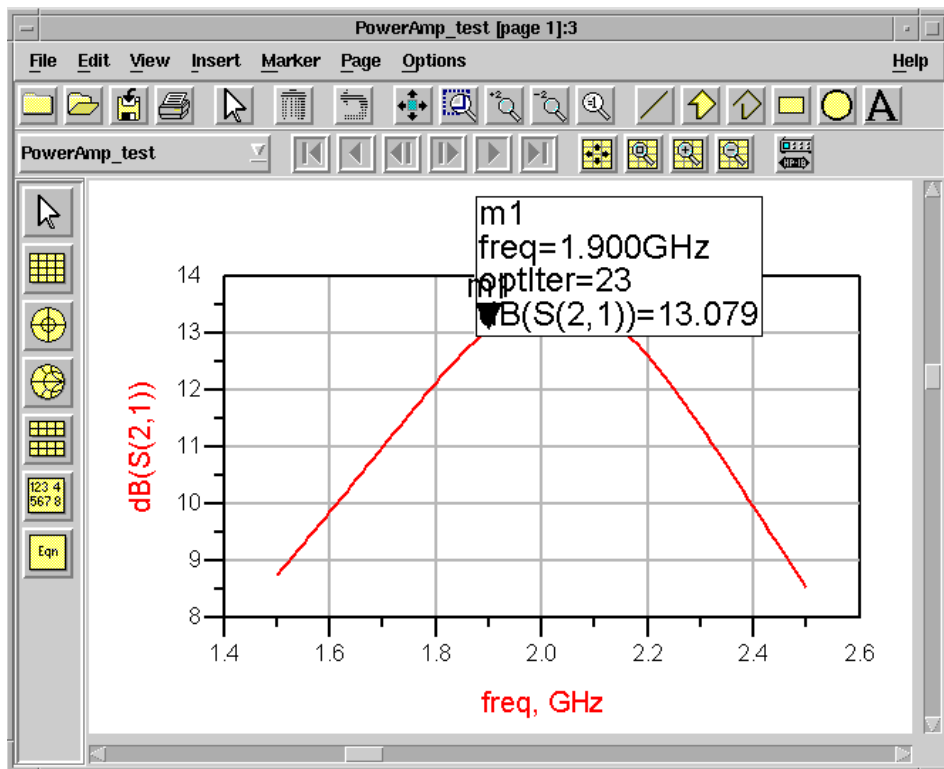


Figure 3-14. Data Display with Optimized Simulation Results

Ending the Session

Use the following steps to exit the Dynamic Link environment and close both ADS and Cadence.

1. Choose the Advanced Design System Schematic menu option **Cadence > Close Cadence Connection**. This closes ADS and terminates the link between Cadence and the Advanced Design System.
2. Exit from the Cadence CIW.
3. Congratulations... you have now successfully completed the Tutorial.

Chapter 4: Starting, Viewing Designs and Exiting

This chapter describes the procedures for:

- Launching Advanced Design System from a Cadence Schematic window via the RFIC Dynamic Link
- Adding an instance of a Cadence design to an ADS schematic
- Pushing into a design hierarchy
- Exiting Dynamic Link, ADS and the Cadence Schematic window

Starting Advanced Design System

To run Advanced Design System from the Cadence Schematic window using RFIC Dynamic Link:

1. In the Cadence Schematic window, open the desired cellview.
2. Choose **Tools > ADS** from the Cadence Schematic window banner menu. The Advanced Design System Main window appears in the upper left corner of your screen followed, to the right, by an empty ADS Schematic window (this may take some time).

The Cadence schematic window displays an *ArtistUtilities* pull-down menu. This menu provides some familiar, useful *Affirma Analog Circuit Design Environment* (4.4.5 & 4.4.6) interface functionality. For further information about these options, consult your Cadence documentation.

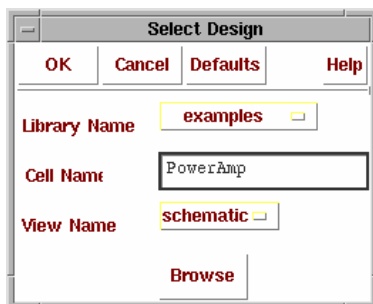


Note The terminal output (*stderr*) of ADS gets redirected to the file *idf.log* in the directory in which the Cadence framework is started. Once the link is started, subsequently opening a Cadence design will not involve the overhead of re-starting ADS, but you will need to select **Tools > ADS** just to see the *ArtistUtilities* pull-down menu.

Adding an Instance of a Cadence Design

To add an instance of a Cadence design to an ADS test schematic, choose **Cadence > Instance > Add Instance of Cellview**.

A dialog box appears, allowing the selection of a Cadence design.



If a symbol already exists for the design in Cadence, the symbol geometry is duplicated in ADS; otherwise the Cadence symbol generator is automatically invoked to generate a Cadence symbol, which is then automatically duplicated in ADS.

If **Cancel** is selected, or if the configuration file has the entry,


```
IDF_CADENCE_SYMBOL = FALSE
```

the ADS symbol generation is automatically invoked (as opposed to the Cadence symbol generation). This generates the symbol in ADS.

Note The generated symbol can be edited and modified if needed. If aesthetics are a concern, it is recommended that the symbol be manually created in Cadence and then automatically replicated in ADS as described above. The symbol of the Cadence design is given the following nomenclature `<library>_<cell>_<view>`. For example, *examples_PowerAmp_schematic*. There is a skeleton schematic of the Cadence design that also gets created in ADS. This is used as a *placeholder* to link with the actual Cadence design; you do not need to edit this.

Pushing into the Design Hierarchy

To view a design deeper in the Advanced Design System schematic hierarchy:

1. Select the component you want to push into in the ADS Schematic window.
2. Choose the *Push Into Hierarchy* icon. This downward arrow icon is located below the Cadence menu item in the tool bar. 
3. If the selected component is a Cadence cellview instance, the corresponding Cadence cellview is opened in the Cadence design editor. If this view is already open, it is simply raised to the top of the window stack.

Exiting

To exit RFIC Dynamic Link and related tools:

1. Choose **Cadence > Close Connection**. This terminates the link between Cadence and ADS and exits ADS.
2. From the Cadence CIW select **File > Exit**. This closes all Cadence tools, after possibly prompting you to save your changes.

Chapter 5: Netlisting, Simulating, and Displaying Data

This chapter describes the procedures for netlisting and simulating a design as well as viewing the netlist from either Advanced Design System or a Cadence Schematic window. Information on net, instance and expression name mapping is also provided.

Netlisting and Simulating a Design

Netlisting automatically occurs when you simulate your schematic in Advanced Design System. The complete netlist is sent to the simulator, stored in memory, and written to a *netlist.log* file in the project directory of ADS. You can view the netlist in the *netlist.log* file as needed.

To netlist and simulate a schematic in Advanced Design System:

1. In the ADS Schematic window, choose the *Simulate* icon or choose the menu item **Simulate > Simulate**.



A Simulation window appears, indicating the netlisting status and listing any errors encountered. If the netlisting is successful, the design is then simulated; otherwise, act on the errors displayed in the Simulation window and repeat step one above.

2. For information on configuring and viewing the simulation results in the Data Display window, refer to the “*Data Display Basics*” in the ADS *Data Display* manual.

Viewing Netlists

This section describes how to view the top-level netlist from Advanced Design System as well as how to view an ADS subnetwork netlist for a Cadence design from a Cadence Schematic window.

Viewing Netlists from Advanced Design System

To generate and display the entire top-level ADS netlist, select **Cadence > Top-level Design Netlist**.

```

Options ResourceUsage=yes UseNutmegFormat=no TopDesignName="/a/new/sr/learn2/d1/local/users/crsc0

; Library name: examples
; Cell name: PowerAmp
; View name: schematic
#ifndef inc_examples_PowerAmp_schematic
#define inc_examples_PowerAmp_schematic
define examples_PowerAmp_schematic ( GND VCC1 VCC2 Vin Vout)
nnpwa2:Q7 net6 VCC2 net37 0
nnpwa2:Q6 net9 VCC2 net35 0
nnpwa2:Q5 net12 VCC2 net13 0
nnpwa2:Q4 net15 VCC2 net31 0
nnpwa2:Q3 net44 net15 net6 0
nnpwa2:Q2 net44 net15 net9 0
nnpwa2:Q1 net44 net15 net12 0
nnpwa1:Q0 net44 Vin net15 0
R:R7 net31 GND R=200
R:R6 net13 GND R=Remitout
R:R5 net35 GND R=Remitout
R:R4 net37 GND R=Remitout
R:R3 net6 Vout R=Rout
R:R2 net9 Vout R=Rout
R:R1 net12 Vout R=Rout
R:R0 VCC1 net44 R=10 Tnom=25

```

Figure 5-1. Viewing Top-Level ADS Netlist

Viewing Netlists from the Cadence Schematic Window

To generate and display the ADS subnetwork netlist for the Cadence design displayed in a particular Cadence Schematic window:

1. From the menu bar, select **ArtistUtilities > Subcircuit Netlist**. Netlisting progress is displayed in the Cadence CIW.

Note If you do not see the *ArtistUtilities* pull down menu in the Cadence tool bar, and you only want to generate a Cadence Subcircuit Netlist, select **Tools > ADS > Turn On ArtistUtilities** from the Cadence Command Interpreter Window (CIW).

```
; Library name: examples
; Cell name: PowerAmp
; View name: schematic
#ifndef inc_examples_PowerAmp_schematic
#define inc_examples_PowerAmp_schematic
define examples_PowerAmp_schematic ( GND VCC1 VCC2 Vin Vout)
nprnpwa2:Q7 net6 VCC2 net37 0
nprnpwa2:Q6 net9 VCC2 net35 0
nprnpwa2:Q5 net12 VCC2 net13 0
nprnpwa2:Q4 net15 VCC2 net31 0
nprnpwa2:Q3 net44 net15 net6 0
nprnpwa2:Q2 net44 net15 net9 0
nprnpwa2:Q1 net44 net15 net12 0
nprnpwa1:Q0 net44 Vin net15 0
R:R7 net31 GND R=200
R:R6 net13 GND R=Remitout
R:R5 net35 GND R=Remitout
R:R4 net37 GND R=Remitout
R:R3 net6 Vout R=Rout
R:R2 net9 Vout R=Rout
R:R1 net12 Vout R=Rout
R:R0 VCC1 net44 R=10 Tnom=25
end examples_PowerAmp_schematic
#endif
```

Figure 5-2. Viewing a Subnetwork Netlist

2. A log window pops up, displaying the netlist results. Once you have viewed the results, you can select **File > Close Window** to exit this window.

Net and Instance Name Mapping

Since Advanced Design System nomenclature rules differ from those of Cadence, nets, instances, etc. must be properly mapped. This mapping is done automatically as part of the netlisting function. The mapping rules are as follows.

- Advanced Design System keywords used as net or instance names are mapped by appending an underscore (_) to the name.

Table 5-1. Net and Instance Name Mapping

Name	Map
then	then_
else	else_
elseif	elseif_
endif	endif_
equals	equals_
notequals	notequals_
and	and_
not	not_
or	or_
global	global_
model	model_
define	define_
end	end_
parameters	parameters_

- Any non-alphabetical character (e.g. not a-z) in a net or instance name is mapped to an under bar (_).
- Advanced Design System uses a single name space for all names, regardless of object type (net, instance, etc.). This may necessitate name mapping in addition to the above.

Expression Name Mapping

Most Cadence *Analog Expression Language* (AEL) expressions contain constants, functions and suffixes with equivalents in ADS. In most cases the names of these equivalents are identical, requiring no mapping. As far as possible, Cadence expressions are pre-evaluated in the Cadence environment, prior to netlisting and prior to getting design variables from Cadence. This leaves only a few built-in *function names* to map (i.e. names that are not identical in the two environments).

Table 5-2. Function Name Mapping

Cadence	ADS
complex	cmplx
fabs	abs
log	ln
log10	log

Some built-in operator and function names in the Cadence *Affirma Analog Circuit Design Environment* (4.4.5 and 4.4.6) as yet do not map to anything in the ADS environment.

Table 5-3. Non-Mapping Operators

Cadence	Description
-	unary minus
~	unary one's complement
%	modulo
<<	left shift
>>	right shift
&	bitwise AND
	bitwise OR
^	bitwise XOR
?:	conditional expression

For these *non-mapping functions*, custom equivalents in ADS need to be written and mapped until they are available as *built-ins* in ADS. Custom mapping is enabled via the configuration file option `IDF_EXPR_MAP`. For more information, refer to Expression Mapping in [“Modifying the Configuration File” on page 2-9](#).

Table 5-4. Non-Mapping Functions

Cadence	Description
acosh	inverse hyperbolic cosine
asinh	inverse hyperbolic sine
atanh	inverse hyperbolic tangent

Table 5-4. Non-Mapping Functions

aeICheckRange	determines if a number falls within a range
conjugate	complex conjugate
floor	floor of a real number
ceil	ceiling of real number
mag	magnitude
db10	10 times log10
db20	20 times log10

Using Global Nodes

Cadence designs typically use implicit global nodes (names ending in *!*) for substrate power and ground connections. This notation is now supported by Advanced Design System. If the exclamation point suffix is used, a globalnode does not need to be placed in the ADS schematic.

Chapter 6: Using Design Variables

This chapter describes how to add or edit a design variable in Advanced Design System and also update your Cadence design variables.

Cadence *Affirma Analog Circuit Design Environment* (*Analog Artist* in 4.4.3) design variables are intended to be global in the context of a particular Artist session or Cellview. When you select **Cadence > Design Variables > Get Design Variables**, these variables are automatically mapped to corresponding variables in a *VarEqn* component in the Advanced Design System schematic. This mapping ensures that these variables can be used for optimization or statistical analysis in ADS.

All the design variables for each Cadence design are put into a single *VarEqn* component. Each time the menu item **Cadence > Design Variables > Get Design Variables** is selected, this component is updated with the most recent values from Cadence.

Adding and Editing Design Variables

To add or edit a design variable for the ADS schematic:

1. From the Cadence schematic menu bar, choose **ArtistUtilities > Design Variables**.
2. In the ADS Schematic window, choose **Cadence > Design Variables > Get Design Variables**. This places a corresponding *VarEqn* component on the ADS schematic containing the design variables from Cadence. If the *VarEqn* component already exists it is updated only with variables and values that are not already there.

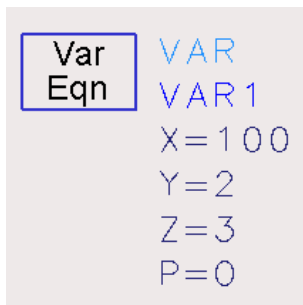
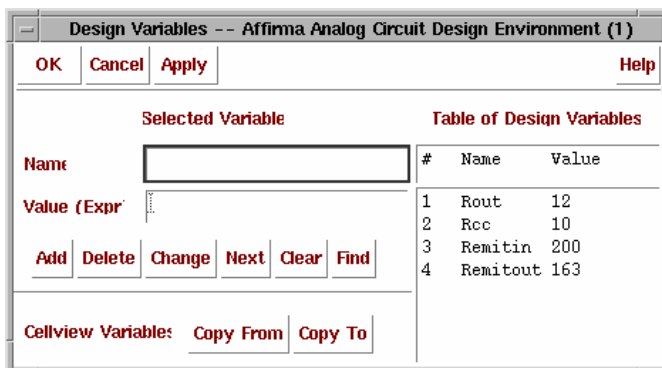


Figure 6-1. VarEqn Block corresponding to Design Variables

Note There is no way to distinguish a design variable of the same name coming from different Cadence cellviews. If a variable has different values in different cellviews, the value sent to ADS is chosen arbitrarily. Non-alphanumeric characters, like parentheses, in variable expressions must be preceded by a backslash ('\').



Updating Cadence Design Variables

To update your Cadence design variables:

1. In the Advanced Design System Schematic window, select **Cadence > Design Variables > Update Design Variables**.

Note Design variables can be used for optimizations and sweeps in ADS. Variables used for this purpose should not be assigned a value or expression in Cadence; the Value (Expr) text entry box should be left blank.

Chapter 7: Tuning and Optimizing Designs

This chapter provides information on tuning and optimizing designs using the Advanced Design System tuning and optimization capabilities.

Tuning Cadence Instance Parameters and Design Variables

The ADS tuning capability enables you to change one or more design parameter values and see its effect on the output without simulating the entire design again from the beginning. Dynamic Link extends the ADS tuning function to instance parameters and design variables in Cadence subcircuits. For more information on using the ADS tuning feature, refer to “*Tuning*” in the *ADS Tuning, Optimization, and Statistical Design* manual.

This section uses the *PowerAmp* example used in [Chapter 3, Getting Started Tutorial](#) for demonstrating tuning of Cadence instance parameters and design variables.

A Dynamic Link For Cadence Tuning Example

1. Follow the steps listed in “[Performing an S-parameter Simulation](#)” on page 3-24 in the [Chapter 3, Getting Started Tutorial](#). Plot S(2,1) in the ADS Data Display window after ADS S-parameter simulation is completed. The S(2,1) plot should resemble [Figure 7-1](#).

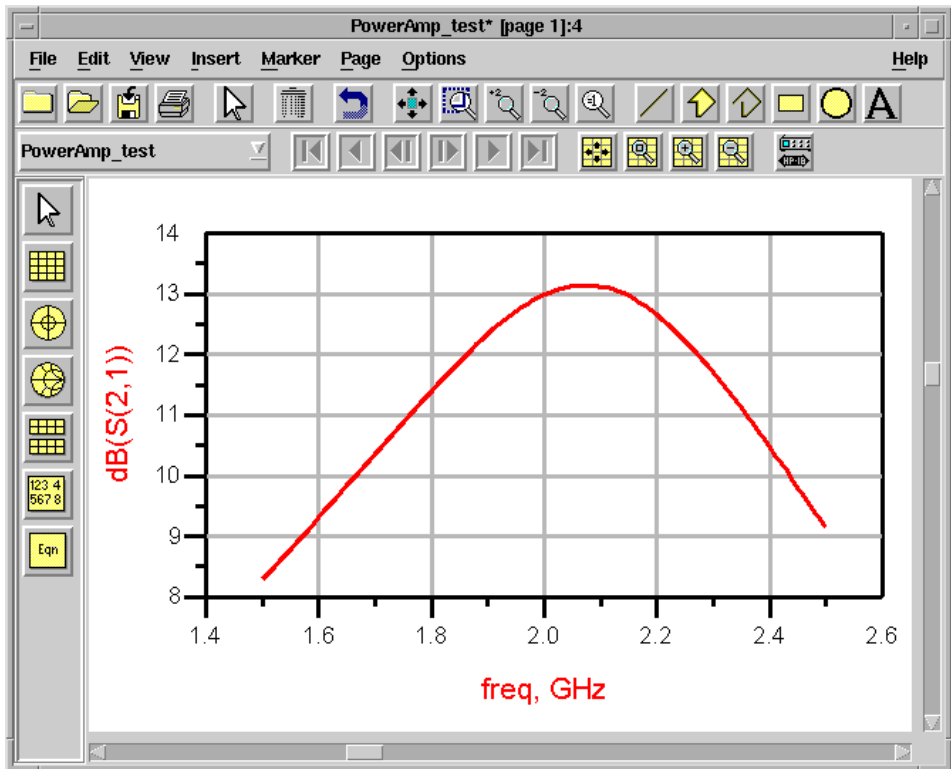



Figure 7-1. Data Display with S(2,1) of the *PowerAmp_test* Design.

2. In the ADS Schematic window, choose **Simulate > Tuning** or click the Tune Parameters icon (tuning fork) in the toolbar. 
3. Wait for the initial analysis to complete. The *Tune Control* dialog box appears as shown in [Figure 7-2](#).

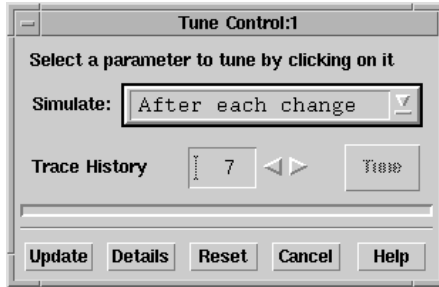


Figure 7-2. The Initial ADS Tune Control Dialog Box.

Note that the prompt at the bottom of the Cadence schematic window says:

ADS Tune Mode: Click an instance>

4. Move your cursor over resistor $R7$ in the Cadence schematic window and click the left mouse button (left click). A slider control for $R7.R$ immediately appears in the ADS Tune Control dialog box. Note that only integer and floating point parameters are tunable. $R7$ contains only one tunable parameter R (r in Cadence spectre), the resistance.
5. Left click resistor $R0$ in the Cadence schematic window. A menu pops up beneath $R0$ as illustrated in [Figure 7-3](#). The popup menu is displayed because there is more than one tunable parameter in $R0$, R and $Tnom$. Cadence instance parameters are sent to ADS one at a time.

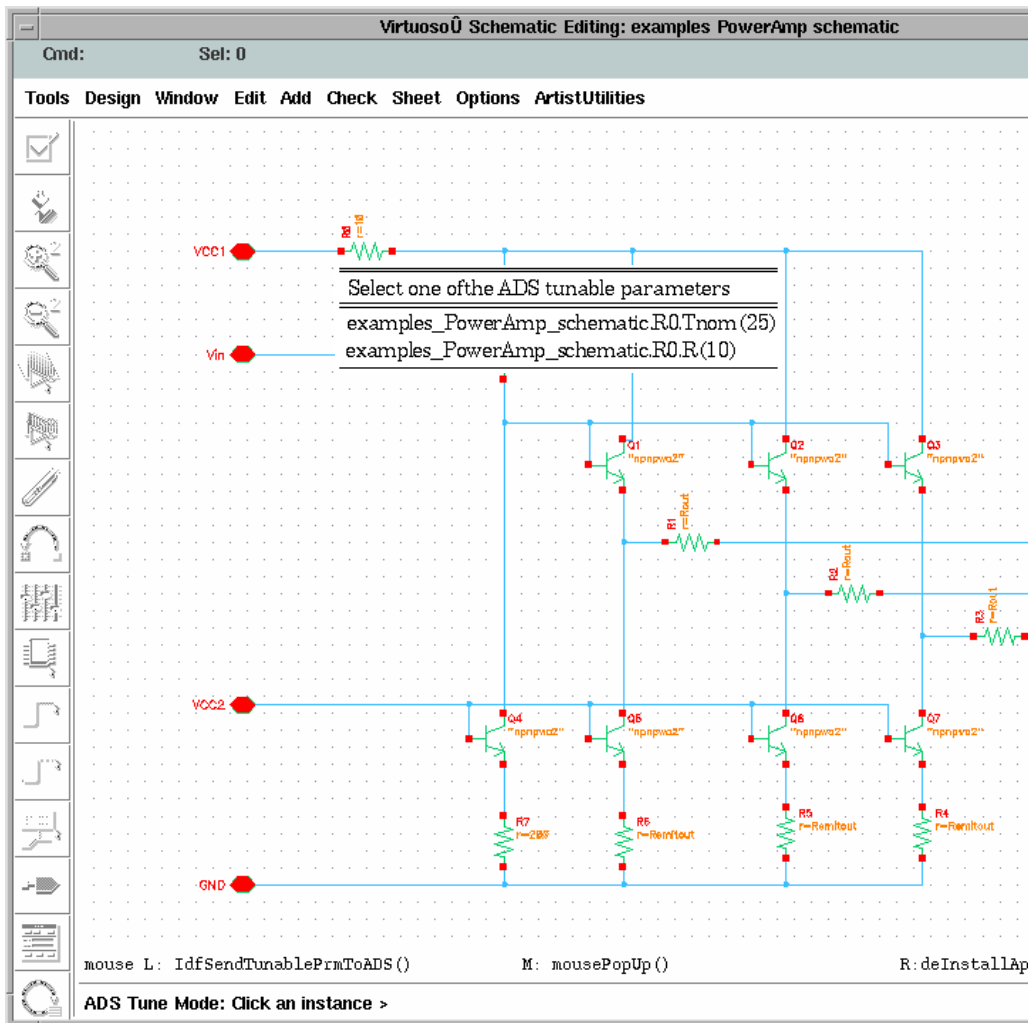


Figure 7-3. Select Tunable Parameter Pop-up Menu in the Cadence Virtuoso Schematic Window.

6. Select *examples_PowerAmp_schematic.R0.R(10)* in the pop-up menu. This creates a slider control for *R0.R* in the ADS Tune Control dialog box.

7. In the ADS Schematic window, click the design variable *Remitout* in the VAR1 block. Figure 7-4 shows the ADS Tune Control dialog box with *R7.R*, *R0.R*, and *VAR1.Remitout* being selected. Recall that *VAR1.Remitout* is a design variable originated from the Cadence subcircuit.

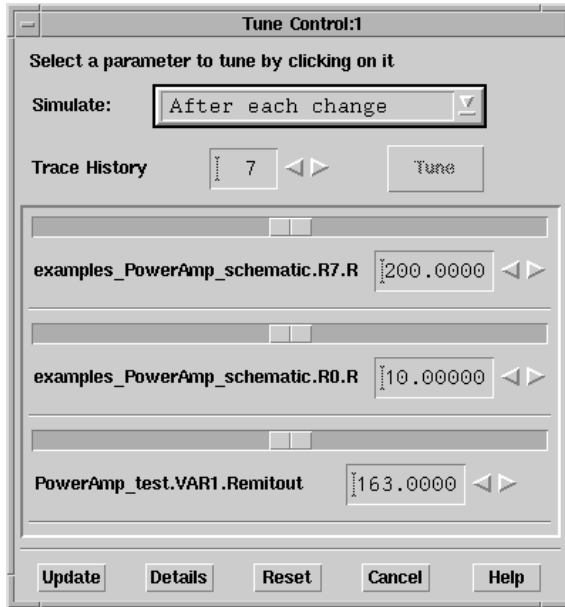


Figure 7-4. ADS Tune Control Dialog Box

Notice the distinction that a Cadence design variable is selected directly from the *VAR1* block in the ADS Schematic window while an instance is chosen at first in the Cadence Schematic window.

8. In the ADS *Tune Control* dialog box, select a tune analysis mode from the *Simulate* drop-down list. This tells ADS when you want tuning to occur. For this example, choose *After each change* if not already selected.

After you finish with all of the steps in this example, try each tuning analysis method (after each change, after pressing the Tune button, while the slider is moving) to see which one works best for you.

9. Drag the slider for *examples_PowerAmp_schematic.R7.R* to 300 (Ohms). You also can change the tunable parameters by doing the following:

- Click the left or right arrows.
 - Type the value in the box.
10. Drag the slider for *examples_PowerAmp_schematic.R0.R* to 5 (Ohms).
 11. Finally, drag the slider for *PowerAmp_test.VAR1.Remitout* to 120 (Ohms). Observe the results in Data Display window each time you release the mouse button after dragging the slider in the Tune Control dialog box to a desired location. **Figure 7-6** shows four S(2,1) curves displayed in the same Data Display window as results of changing *R7.R*, *R0.R* and *VAR1.Remitout* in the Tune Control dialog box.

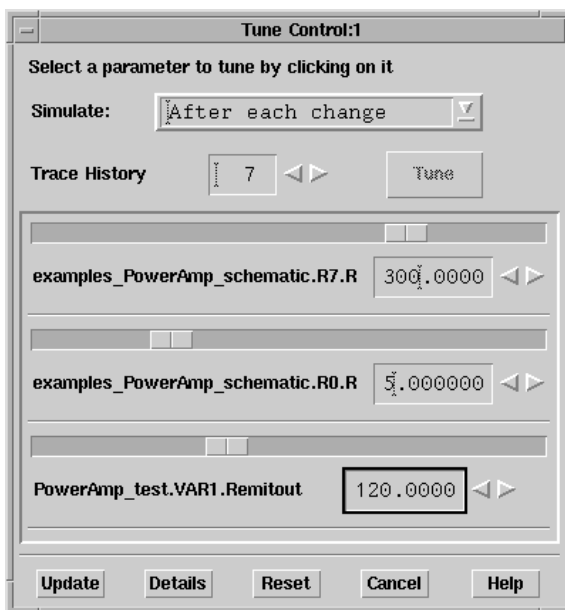


Figure 7-5. ADS Tune Control Dialog Box
(With *R7.R*, *R0.R* and *VAR1.Remitout* values Changed).

12. You can click the **Reset** button to restore all the sliders to their original values. The **Update** button in the Tune Control dialog box enables you to write the instance parameter values currently displayed in the dialog box into the Cadence Schematic window. For Cadence design variables, such as *VAR1.remitout*, you still need to select *Cadence > Update Design Variables to*

Cellviews in ADS schematic window and then follow the instruction in the Confirmation Message form to complete the update.

There is no undo function for the *Update* operation, therefore, do not click the **Update** button if you do not want to change values in the Cadence subcircuit.

- Click the **Cancel** button in the Tune Control Dialog box to end tuning. The prompt at the bottom of the Cadence Virtuoso schematic window returns to its default greater than sign, ">."

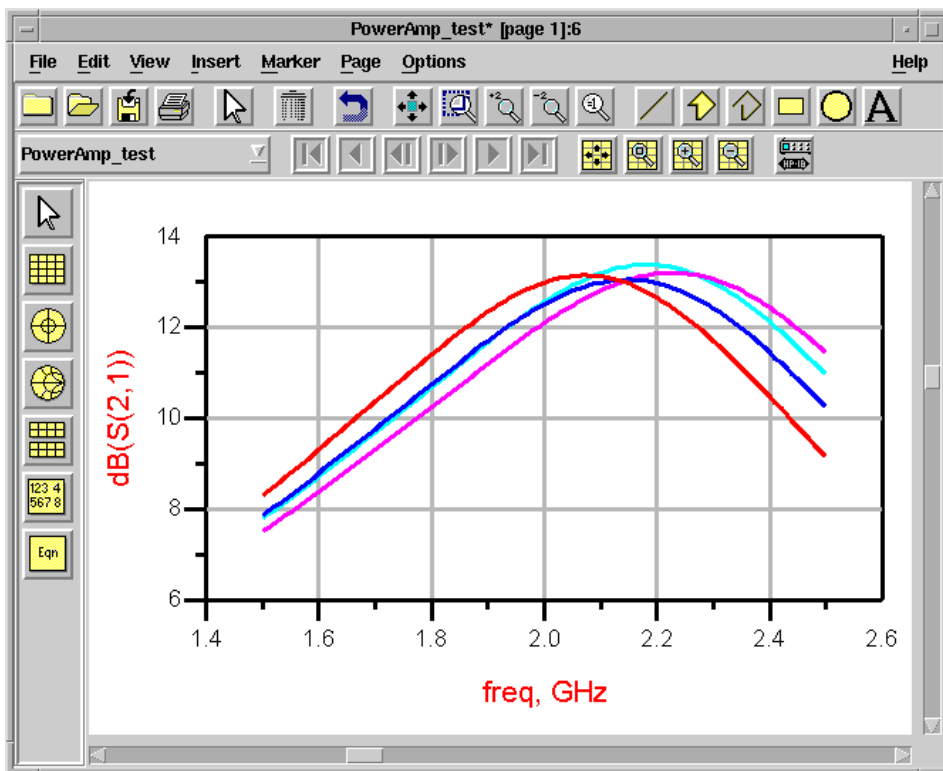


Figure 7-6. Four results of $S(2,1)$
(as $R7.R$, $R0.R$ and $VARI.Remitout$ are changed in the Tune Control dialog box)

This example demonstrated three types of tuning operations in Dynamic Link:

- Clicking a Cadence instance with a single tunable parameter causes that parameter to be sent to ADS for tuning.

- Clicking a Cadence instance with multiple tunable parameters results in a pop-up menu being displayed. Selecting an item from the pop-up menu causes the parameter associated with that menu item to be sent to ADS for tuning.
- After obtaining Cadence design variables by selecting *Cadence > Get Design Variables* in the ADS schematic window, clicking a Cadence design variable in the ADS VAR1 block sends that variable for tuning.

All the above three types of operation act like a toggle switch. Selecting an item already on the Tune Control dialog box removes it from the dialog box.

You can descend down a Cadence design hierarchy to find an instance parameter for tuning. You can then return to higher level Cadence design hierarchy to select another instance parameter.

Note During the node probing operation, the left mouse button in the Cadence Virtuoso schematic window is mapped to a Dynamic Link SKILL procedure. Do not bind any function to the left mouse button during this period. Any bindkey function previously mapped to the left mouse button will not work until the tune mode ends.

If you have a problem while tuning and need to discontinue the operation, enter *IdfMpsTuneEnd* in the Cadence CIW input area. This will end the Dynamic Link Tune Mode operation.

Optimizing Designs

Performance optimization enables you to specify a range of device or component values. The software can then automatically compute the nominal values that best meet your performance goals or specifications. A family of optimizers come with ADS, each with a different mathematical effect or use. For more information on performance optimization, refer to “*Performing Nominal Optimization*” in the ADS *Tuning, Optimization, and Statistical Design* manual.

To optimize a design in the Advanced Design System:

1. In the Schematic window containing the design you want to optimize, choose **Optim/Stat/Yield/DOE** from the component palette. The Optim/Stat/Yield/DOE palette is displayed.

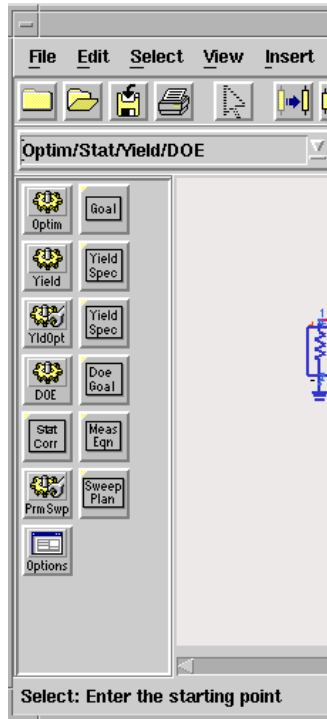
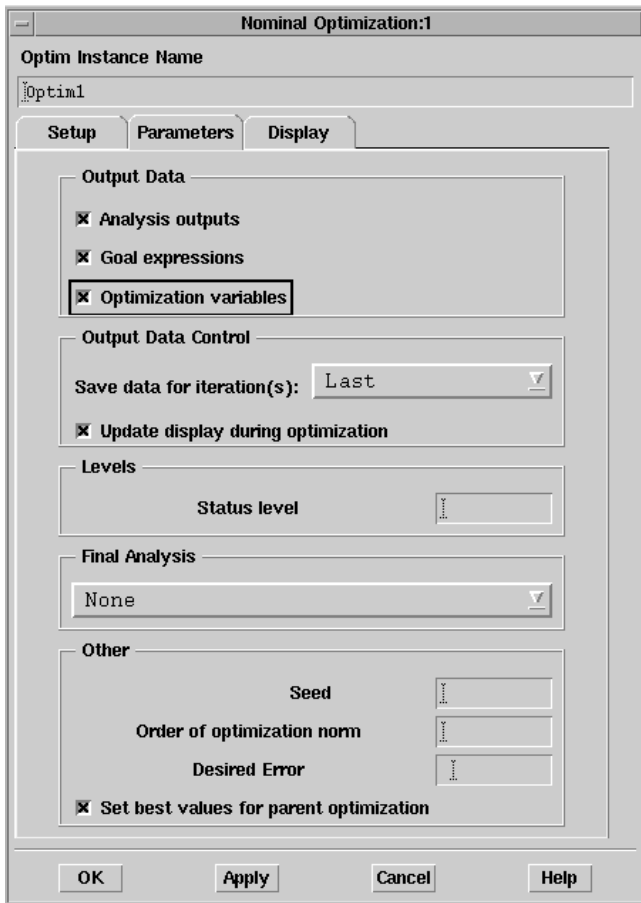


Figure 7-7. The Optim/Stat/Yield/DOE Component Palette

2. Set the options (*Goal, Nominal Optimization, etc.*) as desired. When a *Nominal Optimization* component is first added, you need to enable the output to the dataset. Click the *Nominal Optimization* component and choose **Edit > Component > Edit Component Parameters**. This brings up a *Nominal Optimization* dialog box. Select the *Parameters* tab and ensure that the *Analysis outputs* and *Optimization variables* are checked. Click **OK**.



3. Proceed with the Advanced Design System optimization.

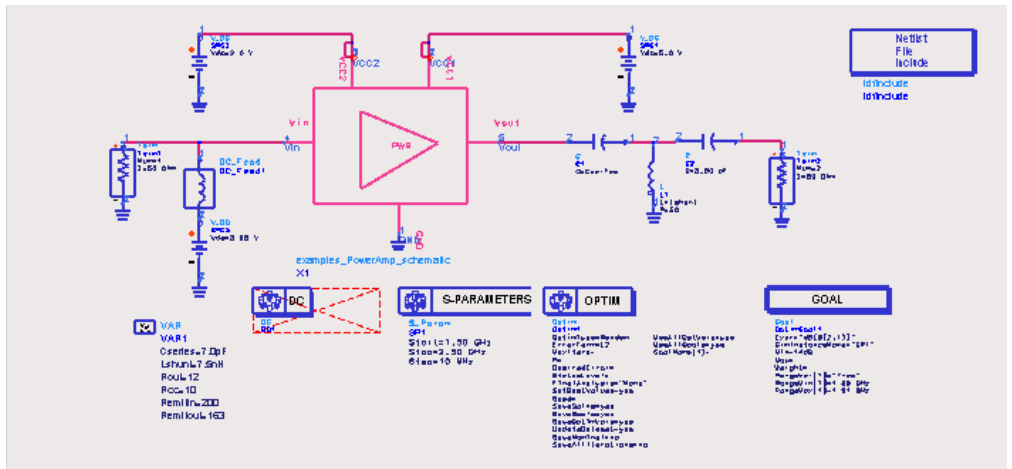


Figure 7-8. An Optimization Setup

Updating the Cadence Cellview

Once the optimum value of a variable is computed by an ADS simulation, you can update the value to the Cadence cellview. To update the Cadence cellview:

1. In the Schematic window, choose **Cadence > Design Variables > Update Design Variables to Cellviews**.

Note Only the nominal values of variables get updated to Cadence; any range values are ignored. Variables to be optimized should not be assigned a value in Cadence; they may be assigned a nominal value and a range in ADS.

Chapter 8: Annotating a DC Solution

This chapter describes how to annotate your simulation results in Advanced Design System to a Cadence cellview.

Annotating DC Voltages to a Cadence Cellview

To annotate a DC voltage solution in Advanced Design System to the Cadence cellview:

1. In Advanced Design System, set up and simulate your schematic. This schematic must contain a *DC Simulation Component* as shown in [Figure 8-1](#).

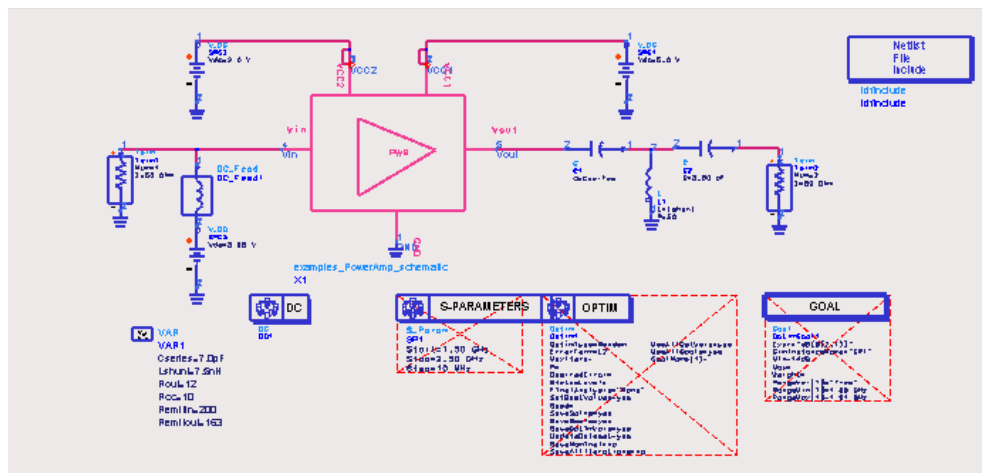


Figure 8-1. Example setup for DC Simulation

2. Select the schematic symbol in the ADS schematic that represents the Cadence circuit you want to back annotate. For example, the amplifier block in [Figure 8-1](#).
3. From the ADS Schematic window, choose **Cadence > Annotate > Annotate DC Solution to Selected Cellview**.
4. The voltages are then displayed on the Cadence schematic as shown in [Figure 8-2](#).

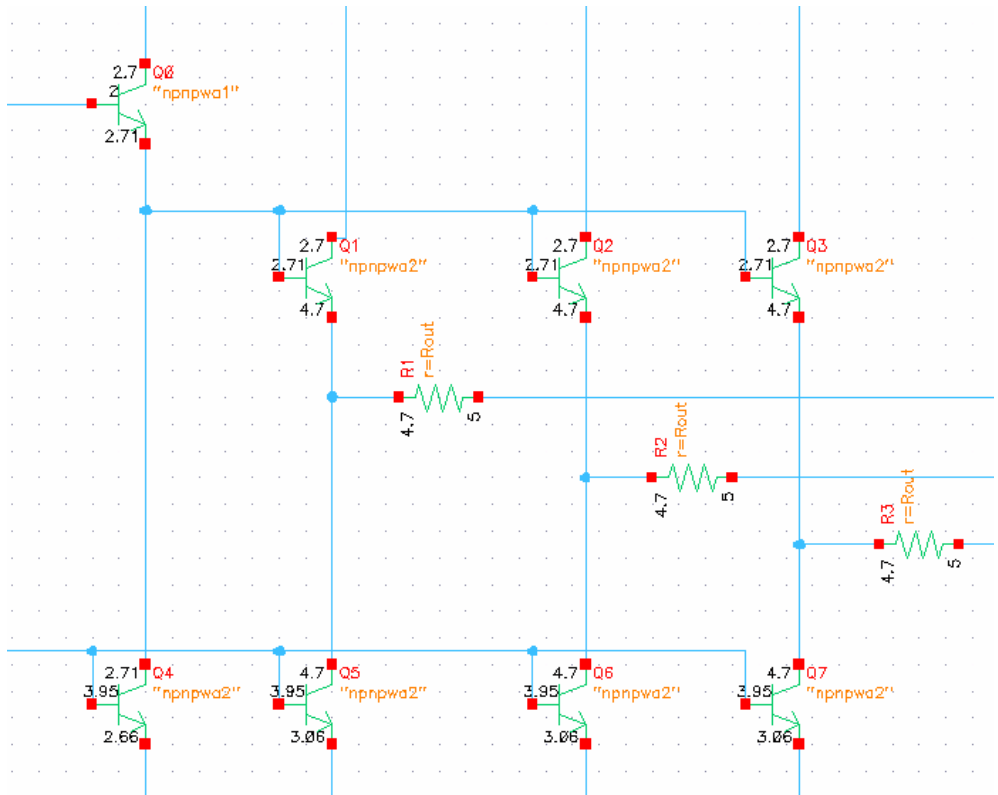


Figure 8-2. DC Voltage Annotation on the Cadence Schematic

Annotating DC Currents to a Cadence Cellview

To annotate a DC current solution in Advanced Design System to the Cadence cellview:

1. First annotate the DC voltages as described in [“Annotating DC Voltages to a Cadence Cellview”](#) on page 8-1.
2. From the Cadence schematic window, choose **Edit > Component Display**. The *Edit Component Display Options* form appears.
3. Click an instance in the Cadence Schematic window. For this example, *Q0* was selected. Note that the title of the Edit Component Display Options form

changes to include the component selected. In this case, the title *Edit Component 'Q0' Display* appears.

4. Click the *terminal* checkbox from the *Select Label* options. Notice that the form now displays a *Terminal Labels* section. This section shows that DC and voltage is currently selected as seen in [Figure 8-3](#).

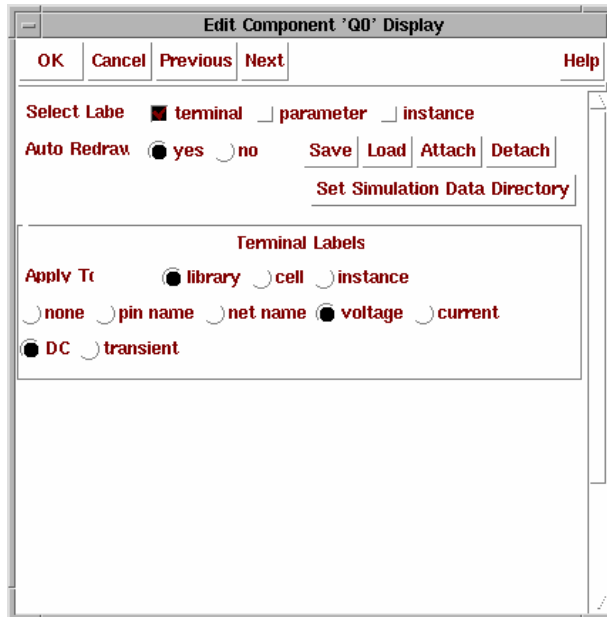


Figure 8-3. *Edit Component 'Q0' Display* form showing DC Voltage

5. Click the *currents* checkbox in the *Terminal Labels* section of the *Edit Component Display* form to display currents instead of voltage. [Figure 8-4](#) shows the design with the DC currents annotated.
6. Click OK to clear the *Edit Component 'Q0' Display* form.

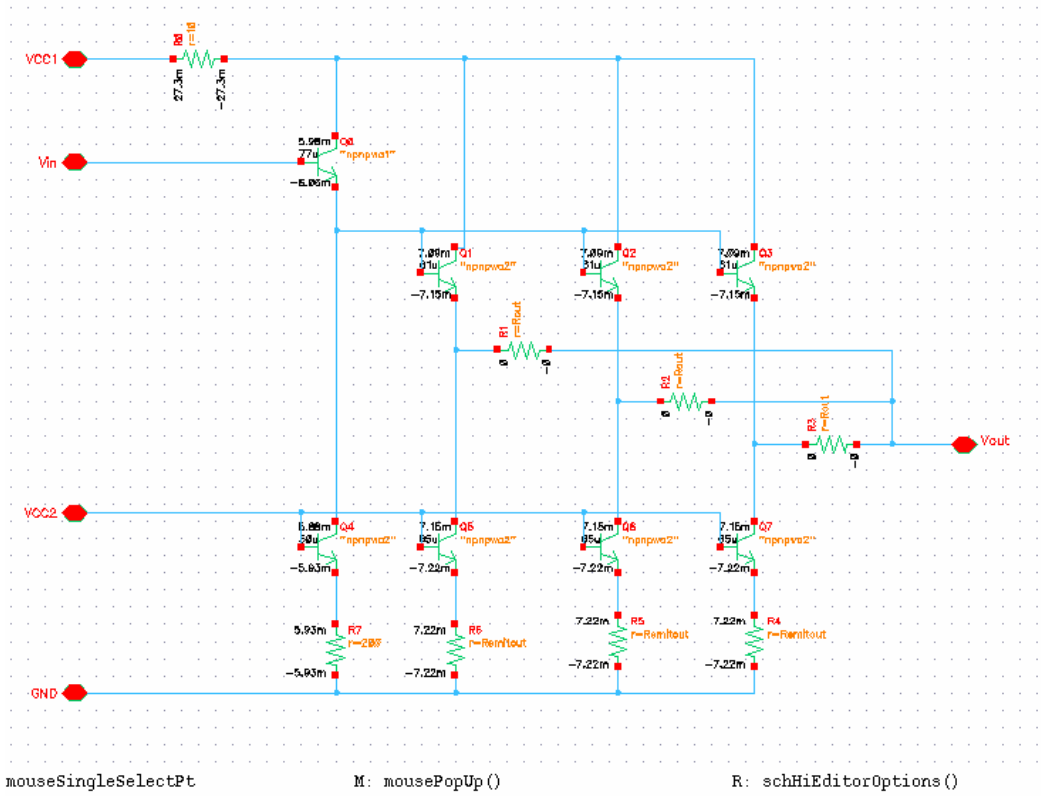


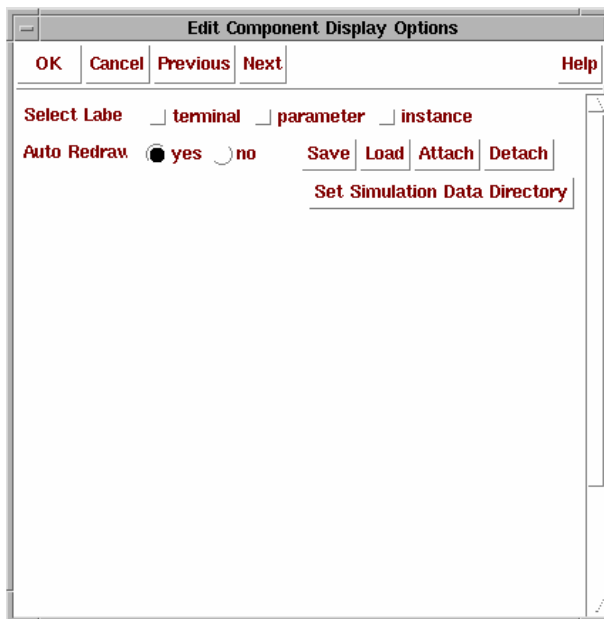
Figure 8-4. DC Current Annotation on the Cadence Schematic

Note If you do not select an instance, the library that your changes will apply to will be the library that contains the schematic (i.e. *examples_lib* for the *PowerAmp* example). Since the primitives (i.e. the *res* and *nnp* cells) are not in the schematic's library, you will not see any changes to the annotation if you do not first select an instance from the proper library. The Edit Component Display form enables you to control how labels are displayed for each library, cell and instance in the design.

Displaying Voltages or Currents from a Previous Simulation

To display voltages or currents from a previous simulation:

1. Before displaying voltages or currents you must have annotated a DC solution to the schematic in a prior Cadence session. Follow the instructions for annotating a DC solution (see [Chapter 8, Annotating a DC Solution](#)) to a schematic if you have not already done so.
2. From the Cadence Schematic window, choose **Edit > Component Display**. The *Edit Component Display Options* form appears.



3. Click **Set Simulation Data Directory**. The *Set Label Display Simulation Data Directory* form appears.



4. Enter the full path to the Data Directory. This is everything up to the *psf* directory. The *psf* (storage format) directory contains Cadence formatted data. The structure for the path name is,

`<simulation_directory>/<cell_name>/<simulator>/<view>`

The path for the Data Directory used in the example for [Figure 8-2](#) was,

`~/simulation/PowerAmp/ads/schematic`

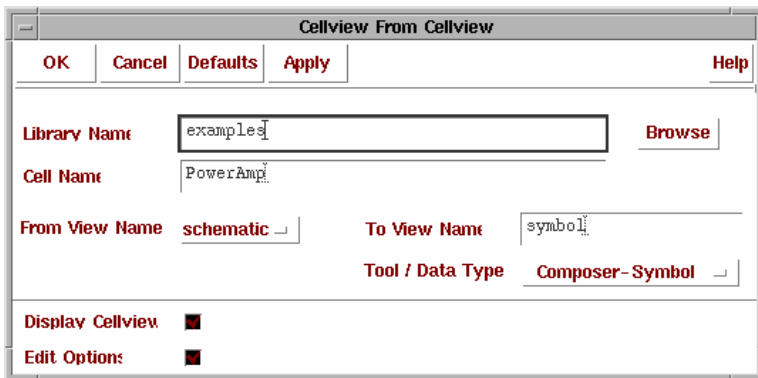
The annotation code looks in the *psf* directory.

5. Click OK in the *Set Label Display Simulation Data Directory* form.
6. Click OK in the *Edit Component Display Options* form.

Creating Symbols for Hierarchical Subcircuits with cdsTerm

To create symbols for hierarchical subcircuits using *cdsTerm*:

1. From the Cadence Schematic window, choose **Design > Create Cellview > From Cellview**. The *Cellview From Cellview* form appears.



2. In the *Cellview From Cellview* form, ensure the following settings are correct:

- From View Name - *schematic*
- To View Name - *symbol*
- Tool / Data Type - *Composer-Symbol*

Click **OK**. The Symbol Generation Options form appears, assuming a symbol does not already exist.

3. In the *Symbol Generation Options* form, click the *Edit Labels* checkbox. Your form will display the *Label* options.

4. In the *Symbol Generation Options* form, select *analog pin annotate* from the *Label Choice* pull-down menu. The Name field should now display `cdsTerm("(pinname)")`.

5. Select *all pins* from the *Apply To* drop-down menu and click **Add**. This generates a new label rule that creates a *cdsTerm* for each pin. You may alter the location if you choose. The form with all appropriate option settings is shown in [Figure 8-5](#).

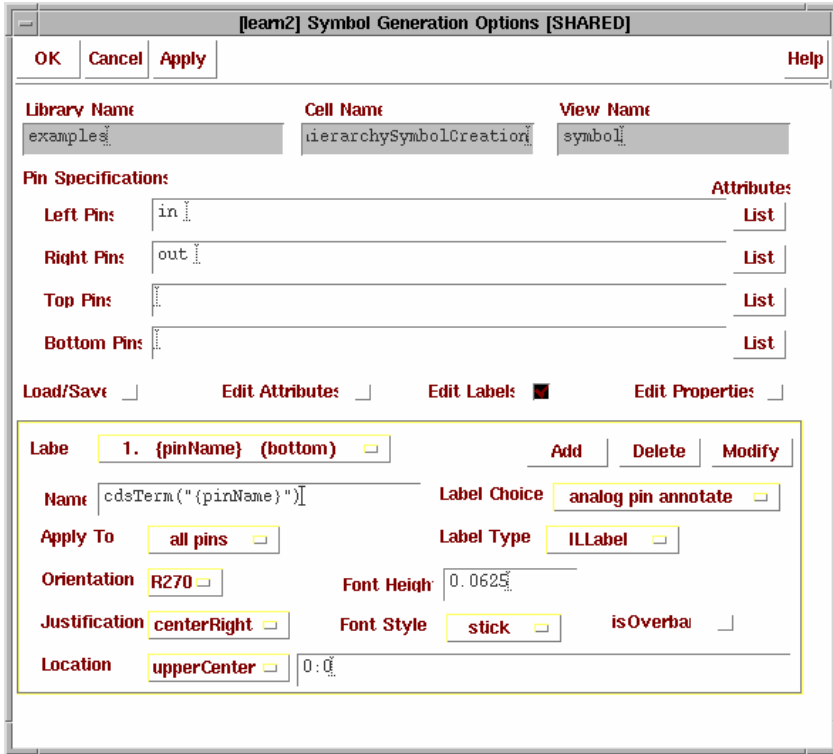


Figure 8-5. Symbol Generation Options to create a symbol with cdsTerms on each pin
 For more information on the *Symbol Generation Option* form, refer to your Cadence documentation.

Chapter 9: Using Additional Features of RFIC Dynamic Link

This chapter describes some of the additional features provided by the RFIC Dynamic Link. Some of the issues related to compatibility between Advanced Design System and the Cadence tools are also discussed in this chapter.

Using the Netlist File Include Component

This section describes how to use the *Netlist File Include Component* in RFIC Dynamic Link. The Netlist File Include component is provided as a means of duplicating the Definition, Stimulus, and Model Library File include used in Cadence/Affirma. This include component can be used with any 4.4.* version of the Cadence DFII.

The Netlist File Include Component is intended to be placed at the top-level ADS schematic. If the component exists in a design below the top level, it *will not be netlisted*.

Adding a Netlist File Include Component

To place an instance of the *Netlist File Include Component*:

1. From the top-level ADS schematic window, choose **Cadence > Add Netlist File Include**. An instance of the Netlist File Include Component is attached to your cursor.
2. Move the cursor to where you want to place the component, then single click. A *Netlist File Include* component symbol is placed on the schematic as shown in [Figure 9-1](#).

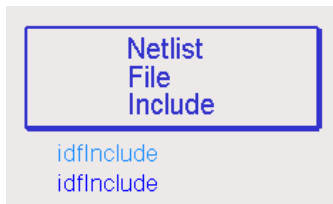
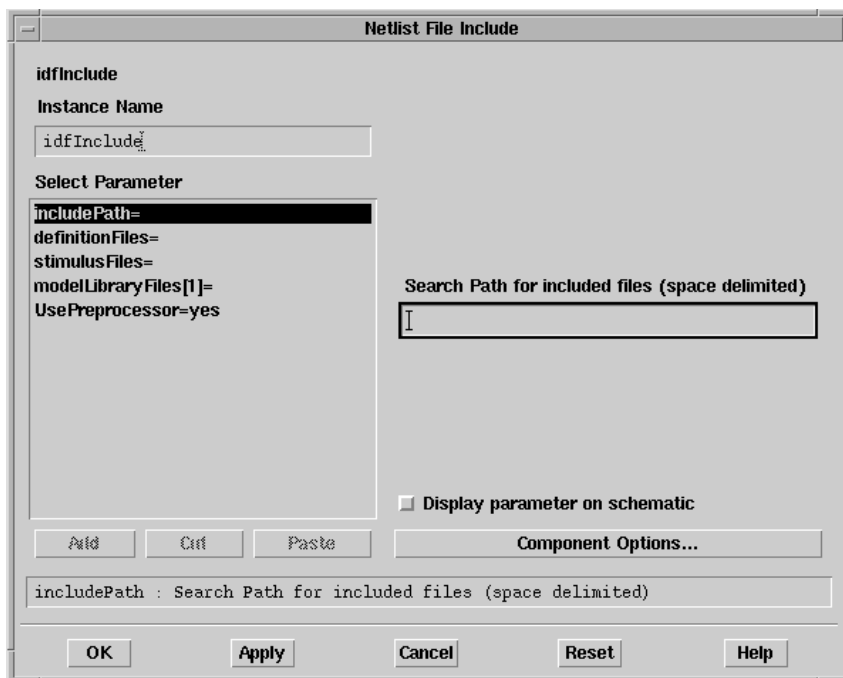


Figure 9-1. The Netlist File Include Component Symbol

Note Only *one* Netlist File Include Component may be placed in a design. This is to ensure that files are not multiply included (this cause's redefinition errors within the ADS simulator). Because the Netlist File Include component must be unique, you cannot place multiple instances and then disable the ones you do not want.

Accessing the Netlist File Include Dialog

To access and edit information in the Netlist File Include Component, double click the Netlist File Include Component icon. The *Netlist File Include* dialog box appears.



Alternatively, you can choose **Edit > Component > Edit Component Parameters** and select the Netlist File Include Component icon or use the *Edit Component Parameters* icon.



Select Parameter

The *Select Parameter* list box displays a list of four parameters that enable you to create your include definition. Refer to each of the sections listed for detailed information on defining these parameters.

[“includePath” on page 9-4](#)

[“definitionFiles” on page 9-5](#)

[“stimulusFiles” on page 9-5](#)

[“modelLibraryFiles” on page 9-6](#)

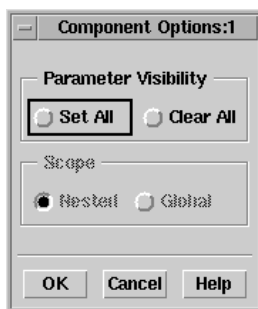
[“UsePreprocessor” on page 9-8](#)

Display parameter on schematic

The *Display parameter on schematic* check box in the Netlist File Include dialog box is used to list the individual parameters and their associated values on the schematic. If you want to display the parameters, activate the check box.

Component Options

The *Component Options* dialog box enables you to change the visibility of the component parameters on a schematic and/or reference items in hierarchical designs. To access the Component Options, click the Component Options button on the Netlist File Include dialog box. A *Component Options* dialog box appears.



Changing the Visibility of Component Parameters on a Schematic

You can change the visibility status of all parameters of the Netlist File Include component through the Component Options dialog box.

- **Set All**—Displays all parameters for this component on the schematic. Use this option to display all, or almost all, parameters for this component. To display most—but not all—parameters, select *Set All* and then go back and turn off the display of individual parameters as desired.
- **Clear All**—Clears the display of all parameters for this component from the schematic. Use this option to turn off the display of all, or almost all, parameters for this component. To display a small subset of parameters, select *Clear All* and then go back and turn on the display of individual parameters as desired.

Referencing VAR Data Items and Model Items in Hierarchical Designs

The *Scope* option applies to the VAR (Variables and Equations) data item and most model items (such as R_Model, BJT_Model, BSIM3_Model). Exception: it does not apply to multi-layer models. Scope indicates the levels, from a hierarchical standpoint, that recognize the expressions defined in the VAR data item or model item.

- **Nested**—VAR or model item expressions are recognized within the design containing the VAR or model item, as well as within any subnetworks (designs at lower levels) referenced by the design containing the VAR or model item.
- **Global**—VAR or model item expressions are recognized throughout the entire design, no matter what level in the design hierarchy the VAR or model item is placed.

includePath

The *includePath* parameter is a space delimited search path that is used to locate definition, stimulus, and model files. The include path needs to be set up for the simulation machine in order to work properly. However, there is an issue with this. The netlister searches through the include path to find files, and then outputs the values as expanded full paths (the simulator requires this). If the expanded full path on the netlisting machine is different from the expanded full path of the simulation machine, the simulator will not find the file to be included. If you want to do remote simulations, ensure that the expanded full path of your included file is the same on the netlisting machine and the simulation machine. Note that, in directory names, path prefixes such as '.', '..', '~', and '\$' all have the usual UNIX interpretation.

To enter a group of include paths:

1. Click the *includePath=* parameter in the *Select Parameter* list box.

2. Enter the name of the search path in the *Search Path for included files (space delimited)* field separating each search path by a space.

definitionFiles

The *definitionFiles* parameter is a space delimited list of definition files. In Affirma parlance, a definition file is a file that contains process definition information. Note that there is no way to specify a particular *segment* of a definition file — this means that the entire file will be used. Do not set up corner case analysis and then use the definition file list to include a library, use the *modelLibraryFiles* parameter instead.

To enter a list of definition files:

1. Click the *definitionFiles=* parameter in the *Select Parameter* list box.
2. Enter the name of each definition file in the *Definition Files List (space delimited)* field separating each file by a space.

stimulusFiles

The *stimulusFiles* parameter is a space delimited list of stimulus files. In Affirma parlance, a stimulus file is a file that contains voltage source and current source definitions. Thus, if you create a global node *VCC*, you can make a stimulus file that contains something like:

```
V_Source:V1 VCC 0 Vdc=5V
```

This would then set *VCC* to be a *5 Volt* global DC node, without requiring you to put the voltage supply into the schematic. No preprocessing is done on any of the stimulus file — everything must be set up properly on your own. This is primarily useful if you are simulating different levels of hierarchy and have used global power rails — the stimulus file can be shared between all of the levels, so that you do not need to disable voltage supplies at different levels of hierarchy to do different simulations.

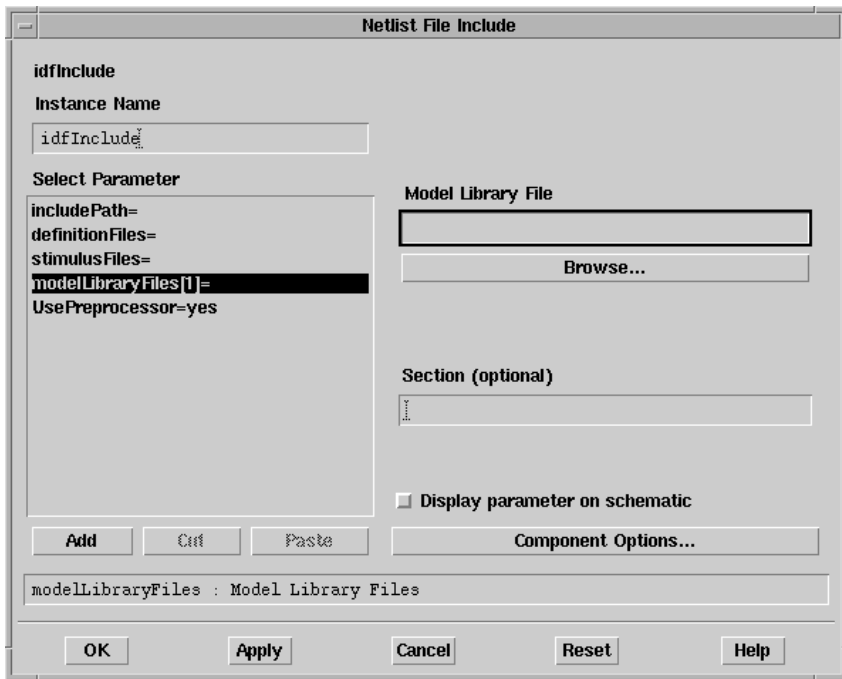
To enter a list of Stimulus files:

1. Click the *stimulusFiles=* selection in the *Select Parameter* list box.
2. Enter the name of the file in the *Stimulus Files List (space delimited)* field. If you want to add additional stimulus files, simply separate each file by a space.

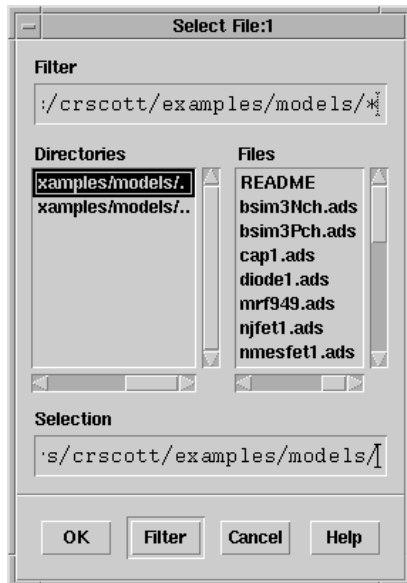
modelLibraryFiles

This parameter enables you to build a list of model files that you want to include. To specify the path and filename of a model file to include:

1. Click the *modelLibraryFiles[n]=* in the *Select Parameter* list box. This activates the *Model Library File* selection field.



2. Click the **Browse** button. The *Select File* dialog appears.



3. Double-click as needed to locate the directory containing your model file or enter the full path and file name in the *Selection* field. Click **OK** to return to the Netlist File Include dialog.
4. Once selected, the filename of the model file is displayed in the Library Model File field. Note that the path is appended to the *includePath* parameter and the file name is added to the *modelLibraryFile* parameter definition.

Example:

```
includePath=Path1 Path2 ... ModelFilePath
modelLibraryFiles[n]=filename
```

5. To add additional model files, click **Add**. This creates additional model file parameter definitions in the *Select Parameter* list box. Repeat steps 1 through 4 to define the path and file name. You can continue adding model file parameters as needed. You can also use the **Cut** and **Paste** buttons to move or delete any model file parameters.

Section (optional)

You may only have a single file for each *modelLibraryFiles[n]* parameter — unlike the prior parameters — this is not a space delimited list. Each model file can have a *Section* designator. This enables you to include only a portion of a model file for *corner analysis*, provided your model file has been set up properly. The section designator is optional; if it is left empty, the entire file will be included (provided it has no dependencies on needing a particular section set up).

To properly set up a model file to utilize the section directive, you must bracket the sections using `#ifdef <section>/#endif` C-Pre-Processor(CPP) directives. The netlister automatically defines and undefines a variable with the name section before and after the `#include` statement. As an example, if you wanted to have a file with corner cases, and had a *Nominal* section, you would make the file as follows:

```
#ifdef Nominal
; Nominal section
R:R1 in out R=50
#endif
```

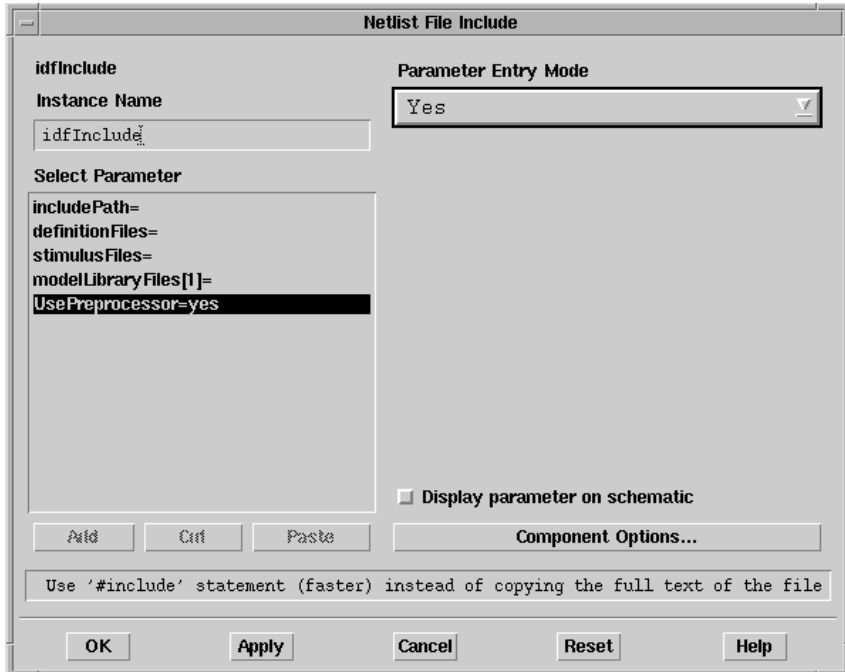
If the same library file is named, with a different section, a single `#include` is generated, with multiple `#define` statements around it.

UsePreprocessor

The *UsePreprocessor* parameter defaults to “yes”. Setting this parameter to *yes* causes a preprocessor directive (`#include`) to be used which results in faster behavior by not copying the full text of the file. This parameter can also be set to *no* to provide a slower, but more compatible, full-text inclusion behavior.

To set the UsePreprocessor parameter,

1. Click the *UsePreprocessor=yes* parameter in the *Select Parameter* field of the Netlist File Include dialog box. The *Parameter Entry Mode* pull-down menu appears.
2. Click the *Parameter Entry Mode* pull-down menu in the Netlist File Include dialog box and select the appropriate option.



The *UsePreprocessor* parameter was developed for delivering increased speed to old ADS designs that use the *geminiInclude*, *spiceInclude* or *csvInclude* component and for delivering increased flexibility to old ADS designs using the *idfInclude* component.

Summarizing the Netlist File Include Component

For all of the Netlist File Include component parameters, a single include statement is netlisted for each file. The netlister checks to see if a file has already been output, to avoid having multiple definitions of files. The precedence is that model files are output first, so that the segment directives can be placed around the **#include**.

If you are using the Netlist File Include component, it is not putting out **#ifdef <file>** statements to further ensure that files are not multiply included. If you use a Netlist File Include component, you should not additionally use other file *include* components to avoid multiple inclusions which will cause a simulator redefinition error.

Example:**Parameter settings**

```
includePath=". ./models"
definitionFiles="functions.def"
stimulusFiles="vccdef.stim"
modelLibraryFiles[1]="resistor.lib Nominal"
```

Netlist File Output (Note that . is `/users/default/default_prj` in this example):

```
#define Nominal
#include "/users/default/default_prj/models/resistor.lib"
#undef Nominal
#include "/users/default/default_prj/models/functions.def"
#include "/users/default/default_prj/models/vccdef.stim"
```

It is worth noting that, once the Netlist File Include component is netlisted, the simulator makes no differentiation between definition, stimulus, or model files. Each file will generate the **#include** statements.

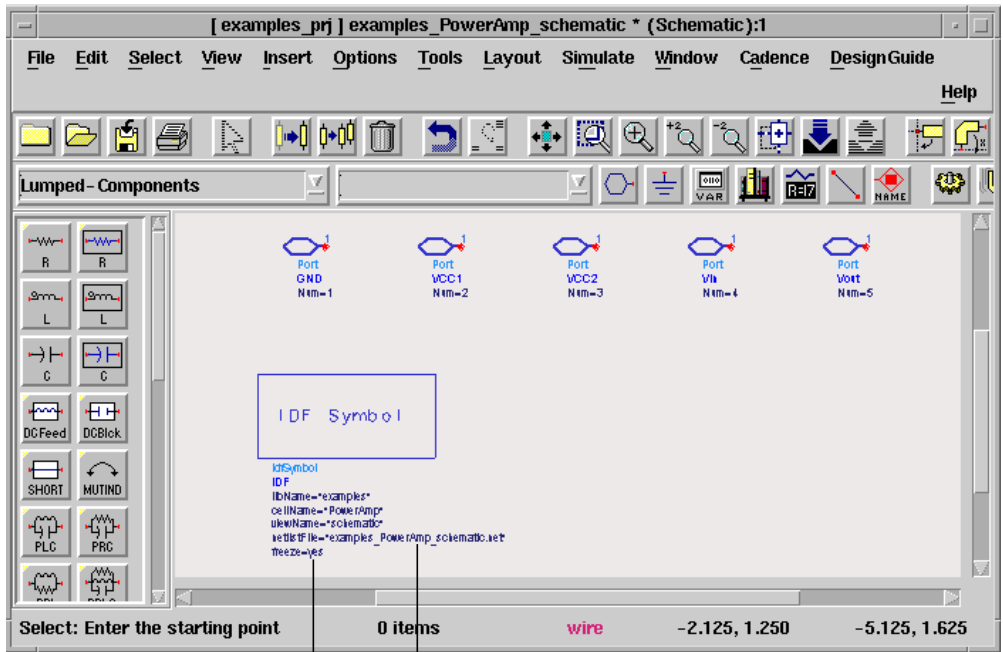
You may want to use the *modelLibraryFiles* parameter for all of your files so that you can put corner case statements into all of your model files.

Freezing Selected Subcircuits

The RFIC Dynamic Link *Freeze* mode enables you to keep Cadence from generating a new netlist each time you simulate in Advanced Design System. This helps to avoid unnecessary time-outs caused by re-netlisting a large Cadence subcircuit.

Setting the Freeze Parameter

If you want to keep a Cadence Cellview from being netlisted and a netlist already exists, edit the ADS dummy design for that Cellview which would have a name of the form `<lib>_<cell>_<view>.dsn` and set the *Freeze* parameter to “yes” or “TRUE”. Refer to the example schematic in [Figure 9-2](#) for the location of the *Freeze* parameter setting. To turn the Freeze parameter off, set the parameter to “no” or “FALSE”. The default value for the Freeze parameter is “FALSE”.



freeze Parameter
 netlistFile Parameter

Figure 9-2. Defining the Freeze Parameter

To freeze all Cadence subcircuits, see [“Modifying the Configuration File” on page 2-9](#).

Generating a Cadence Subcircuit Netlist

In order to run Dynamic Link in Freeze mode, the Cadence subcircuit netlist must exist. If a Cadence subcircuit netlist does not exist, you can generate a new Cadence subcircuit netlist before running your ADS Simulation. Choose **ArtistUtilities > Subcircuit Netlist** from your Cadence Schematic window.

Note If you do not see the *ArtistUtilities* pull down menu in the Cadence tool bar, select **Tools > ADS > Turn On ArtistUtilities** from the Cadence CIW.

This generates the Cadence subcircuit netlist. You can now run your ADS Simulation in *Freeze* mode.

Note If you set the *Freeze* parameter to *TRUE* but the Cadence subcircuit has never been netlisted, the Cadence subcircuit will automatically be netlisted the first time an ADS Simulation is attempted.

Setting the netlistFile Parameter

The *netlistFile* parameter is used to specify the location of the ADS netlist of a frozen Cadence Cellview. On the dummy (placeholder) design, set the *netlistFile* parameter to point to the appropriate netlist file. For example:

```
netlistFile="examples_PowerAmp_schematic.net"
```

The default location for storing the ADS netlist of a frozen Cadence Cellview is:

```
<current_ADS_project_directory>/networks/
```

If you want to copy the netlist file elsewhere, set the *netlistFile* parameter to point to the full path and file name of the new location. For example:

```
netlistFile="/tmp/my_design.net"
```

Refer to the example schematic in [Figure 9-2](#) for the location of the *netlistFile* parameter setting.

Using “Freeze” Mode to Simulate a Design in ADS Standalone

You can run Advanced Design System standalone (without Cadence DFII or RFIC Dynamic Link) using a frozen netlist from an earlier RFIC Dynamic Link session (see [“Freezing Selected Subcircuits” on page 9-10](#)). The parameter *Freeze=TRUE* must be set on all dummy designs in ADS that represent Cadence cellviews.


While RFIC Dynamic Link is not designed to operate on a PC, you can take an ADS netlist (created using Dynamic Link on your UNIX workstation) of your Cadence cellview and copy it to your PC for an ADS simulation. This type of operation is typically done for board design. Once you have done some minor configuration, you can then add simulation and control components externally to your design and resimulate on the PC.

To setup and simulate your Cadence cellview in Advanced Design System on a standalone Windows NT machine:

1. Install RFIC Dynamic Link on your PC.
2. For each Cadence CellView, copy the netlist, AEL and design files into your working project directory's networks subdirectory. For instance, copy the following example PowerAmp schematic files into your project directory:

```
examples_PowerAmp_Schematic.net  
examples_PowerAmp_Schematic.ael  
examples_PowerAmp_Schematic.dsn
```

3. If the Cadence CellView contains design variables, you will need to manually enter them into a VAR block in the top level ADS schematic. To do this:

- Choose *Data Items* from the Component Pallet.
- Click the *Var Eqn* block to add the component and use the cursor to place an instance on the schematic. You may continue placing more instances of the *Var Eqn* block, or choose the *Cancel Command And Return To Select Mode* icon to proceed with the next step. 
- Enter the appropriate values into *Var Eqn* block.

It is important to note that any changes made to the design on your standalone machine will not be reflected in your original Cadence cellview. While you may add simulation and control elements externally, the fundamental design should not be changed if you want it to match your original Cadence design.

Compatibility between Advanced Design System and Cadence Tools

Some of the features provided by the RFIC Dynamic Link include support for compatibility issues related to differences between Advanced Design System and the various Cadence tools. This section addresses several of these compatibility issues.

Support for Duplicate Pin Names

It is typical for the top (chip) level schematic to have multiple pins for the same signal, usually power and ground connections. The netlister lists duplicate I/O ports only once in the subnetwork definition and likewise for the nets connected to an instance of the subnetwork. However, the netlister in ADS (which does the top-level

netlisting), writes out the multiple connections to ports with the same name, causing a conflict to be reported by the ADS simulator while parsing the final netlist. To eliminate this conflict, when the symbol generator encounters duplicate pin names, it draws only one pin with a given name and issues a warning message. However, duplicate pins at lower levels in the Cadence schematic hierarchy are allowed, because no ADS symbol is involved.

Using Buses

For an example on using buses in RFIC Dynamic Link, refer to [Chapter 3, Getting Started Tutorial](#) and use the *PowerAmp2* Cadence cellview in place of the *PowerAmp* cellview. Then use the *PowerAmp2_test* ADS design in place of the *PowerAmp_test* design. The *PowerAmp2* example is the same as the *PowerAmp* example except that it uses bus wires.

For details on creating buses in ADS, refer to Chapter 3 of the *ADS User's Guide*. For details on using buses within Cadence Design Framework II, refer to your Cadence documentation.

Setting up Unnamed Nets

In ADS, unnamed nets begin with an *_net* prefix followed by an integer. All other *net* value are written out to the output dataset during simulation. By default, Cadence tools use the prefix *net* followed by an integer. By default, the dataset can get very large. To avoid this, set the *Net Name Prefix* in the Cadence schematic to *_net* instead of *net*.

To set the default *Net Name Prefix* in the Cadence schematic:

- Choose **Options > Editor** (Cadence 4.4.6) or **Design > Options > Editor** (Cadence 4.4.3)
- In the *Editor Options* dialog box, enter *_net* in the *Net Name Prefix* field
- Click **OK**.
- Choose **Design > Check and Save** to save each related Cadence schematic.

Support for pPar and iPar

This section describes the general use of parent parameters (pPar) and instance parameters (iPar).

pPar()

Figure 9-3 shows an example of an inverter design that contains two CMOS transistors (M0 & M1).

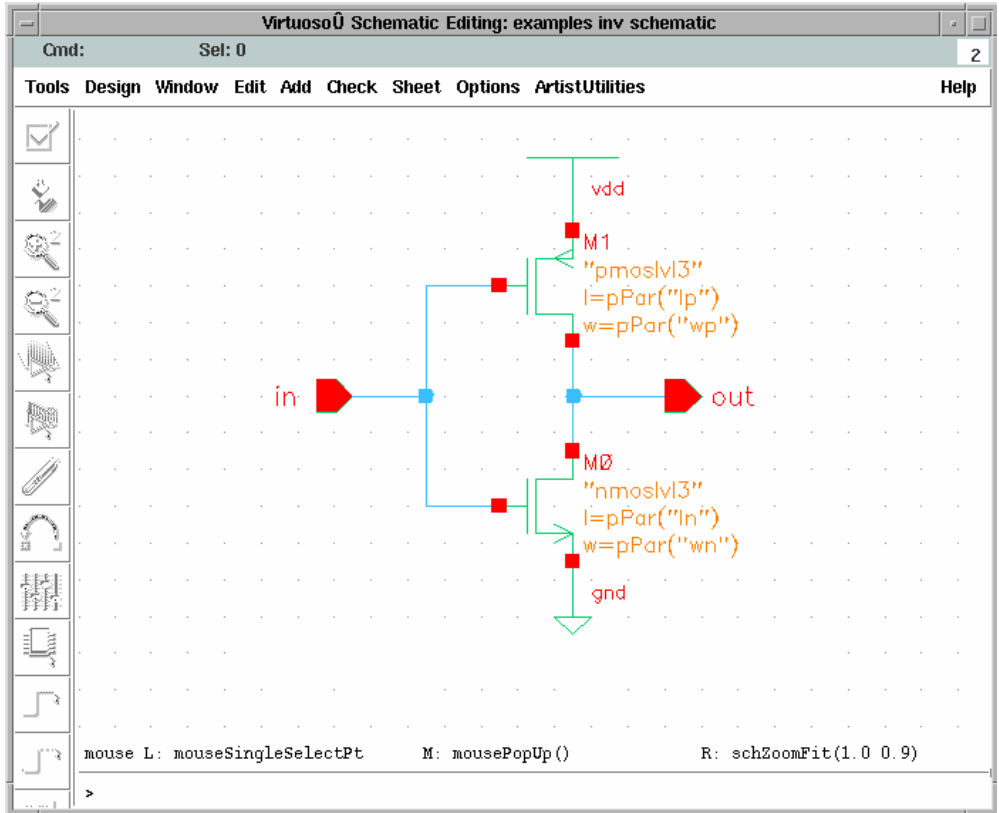


Figure 9-3. Composer Schematic showing use of *pPar()*

This Composer circuit contains instances whose parameters are defined in terms of parent parameter values using *pPar()*. The parameters in this case are defined as,

$$l = pPar("ln")$$

$$w = pPar("wn")$$

for M0, and

```
l = pPar("lp")  
w = pPar("wp")
```

for M1.

This inverter circuit also has an associated symbol view in Cadence Composer. The symbol view shown in Figure 9-4 is equivalent to a black box that displays the input, output and instance properties for the circuit in Figure 9-3.

The default values of *wn*, *ln*, *wp* and *lp* are displayed in the symbol view along with the associated symbol for the device.

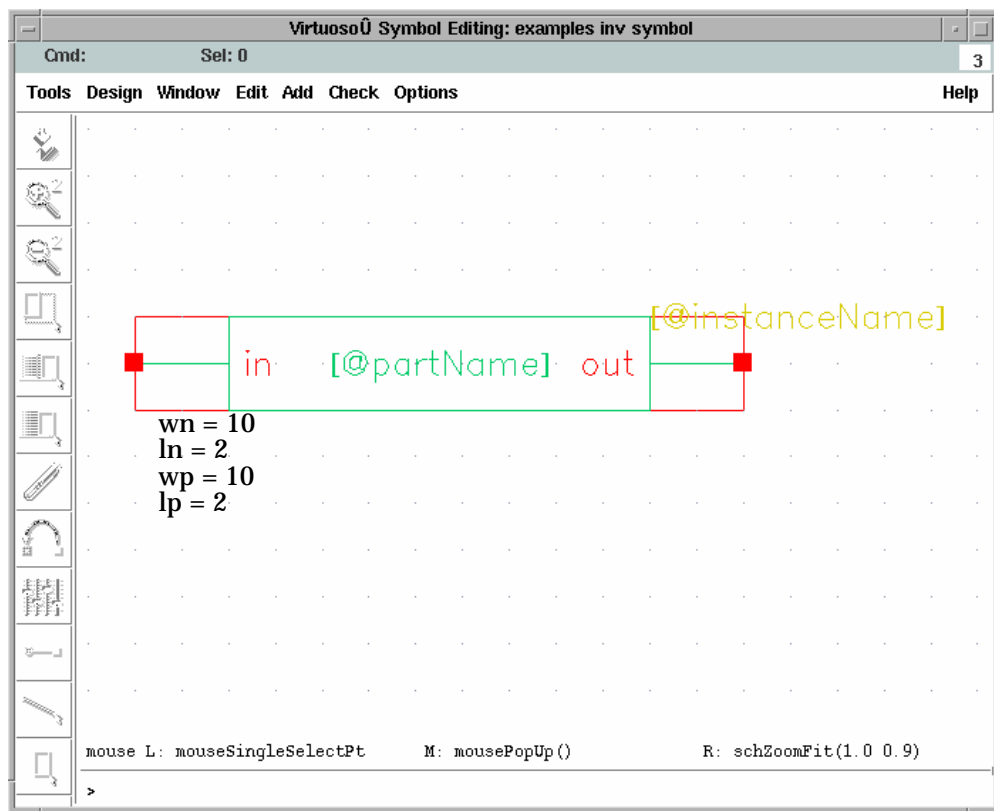


Figure 9-4. Composer Symbol View showing default values for *pPar()*

The Composer symbol is instantiated in ADS via the Dynamic Link where the instance properties also appear in ADS Schematic (see [Figure 9-5](#)).

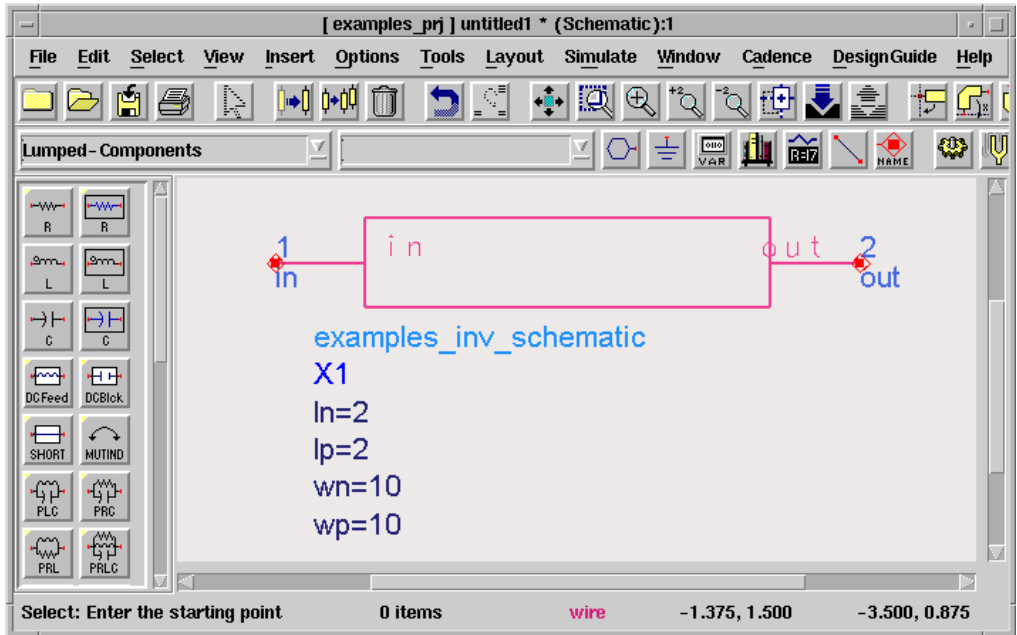


Figure 9-5. Inherited Symbol in ADS

The parameter values are reflected in the netlist that is sent to the simulator. These values can be viewed and edited using the Cadence *Edit Component CDF* form (see [Figure 9-6](#)).

If the Cadence menu option **Design > Create Cellview > From Cellview** is used, the CDF for the schematic will be set up automatically. The Cadence software will traverse the hierarchy looking for *pPar* statements and automatically generate parameters. It will also set up netlisting data for known simulators.

If you are modifying an existing schematic and you have already generated a symbol for your schematic, it may be necessary to manually add the netlisting data for ADS. If this is the case, do the following:

1. Go to the *Simulation Information* section of the *Edit Component CDF* form (see [Figure 9-6](#)) and click **Edit**.

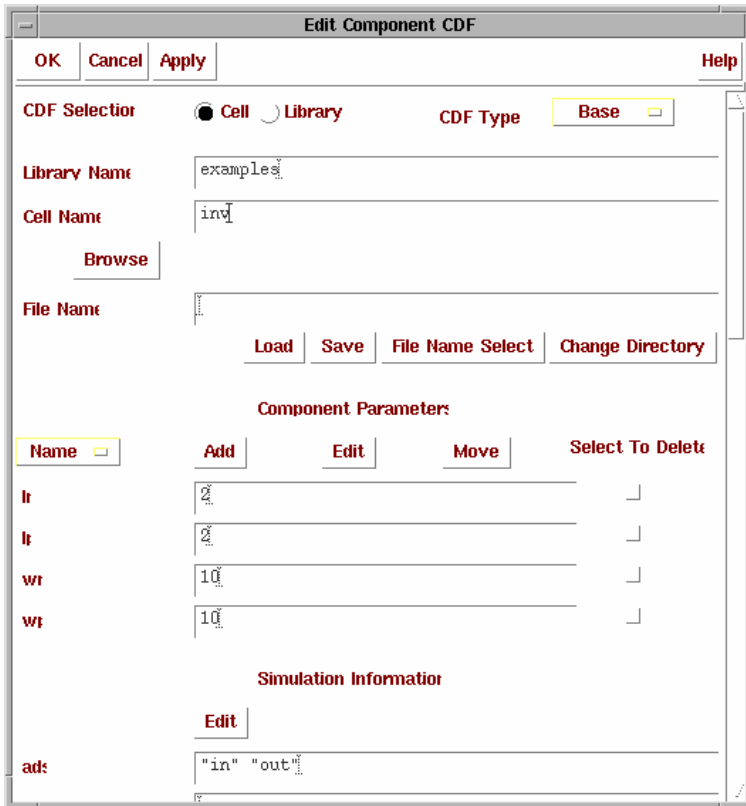


Figure 9-6. Edit Component CDF Form

2. When the *Edit Simulation Information* dialog (see [Figure 9-7](#)) appears, change the *Choose Simulator* setting to “ads”.

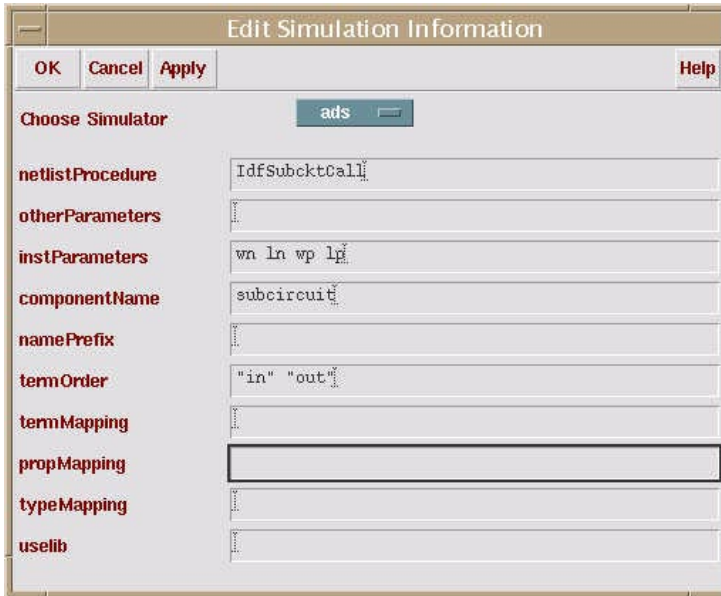


Figure 9-7. Edit Simulation Information Dialog

3. In the *netlistProcedure* field enter “IdfSubcktCall”.
4. In the *instParameters* field enter the parameters you wish to have netlisted for ADS. For the inv circuit, this means entering “wn ln wp lp”. The parameter order does not matter.
5. If you are using Cadence 4.4.3, you also need to set the *macroArguments* field. Change the entry set so that it is identical to the *instParameters* field. For the inv circuit, enter “wn ln wp lp”. Note that, in Cadence versions after 4.4.3, the *macroArguments* field is no longer used and is not displayed in the dialog box.
6. Change the *componentName* field entry to “subcircuit” or leave it blank.
7. Set the *termOrder* field to the order you wish to netlist the terminals in for ADS. You should have one entry for each terminal on your design. For the inv circuit, this is set to “in out”. You can also quote the names, but it is not necessary.
8. If you wish to back annotate currents, make the appropriate entry in the *termMapping* field. This is done by specifying the name of a terminal, followed

by the ADS pin number it will be. For the *inv* example, termMapping entry would be “nil in “:P1” out “:P2””.

For more information on editing simulator information, refer to “*Modifying the Component Description Format*” in Chapter 3 of the “*RFIC Dynamic Link Library Guide*”.

iPar()

Similarly, the Dynamic Link supports the use of *iPar*: For any given instance, you can define an instance parameter as a function of another parameter of the same instance. For example, if the parameter *w* in [Figure 9-3](#) were defined as $w=2*iPar("l")$, then if $l=10$, then $w=20$.

Using Inherited Connections

You can use inherited connections in Dynamic Link. It is recommended that you only use inherited connections with Cadence 4.4.5 and up however; they have been validated to work with Cadence 4.4.3 as well. In Cadence 4.4.3, programmable nodes may also be used, and are recommended in preference to inherited connections.

Inherited connections used in Dynamic Link must all be resolved within the Cadence hierarchy. For example, if you create a schematic in Cadence called *test*, that contains instances that have inherited connections with them, such as *nmos* in *analogLib*, the default connectivity is used in *test* if no netset properties have been placed on instances in the hierarchy of the top level circuit. If you have hierarchy above the top level Cadence schematic placed in the ADS design environment, you cannot place netset properties on those instances.

Using S-parameter File Devices from analogLib

Cadence provides four S-parameter file components in the analogLib library:

- n1port
- n2port
- n3port
- n4port

These devices are supported in ADS with the *rficdl.library* file. In order to use the devices:

1. Make sure that the *rficdl.library* file is included as the IDF library in the file *ADSLibconfig*. This file is found in `$HPEESOF_DIR/circuit/config`. Ensure that the following line appears in the file:

```
IDF $HPEESOF_DIR/idf/components/rficdl.library
```

The installation procedure should add the proper line in automatically.

2. Cadence's spectre and ADS do not use the same format for S-parameter files. However, there is a single parameter that is used to designate the S-parameter file name for both simulators. While you can use a full path name in the file parameter, it is recommended that you input the file name, and then set up paths that will point to different directories where the files with the proper format can be found. In Cadence Affirma, the search path for the S-parameter files is the same as the search path for the model files. Within ADS, the search path is based on the data file search path. By default, the data file search path in ADS is `./data`. If you have a process kit that contains the S-Parameter files, it will not be convenient to copy the files into your working project. To avoid this, you can add the configuration variable `DATA_FILES=` to your `de_sim.cfg` file (this can be in `$HPEESOF_DIR/config`, `$HPEESOF_DIR/custom/config`, or `$HOME/hpeesof/config`). Each path should be delimited with a semicolon character. It is recommended that you keep `./data` within the path for compatibility.

Cadence provides a utility program that can translate Cadence spectre S-parameter files to ADS touchstone S-parameter files. The program is called *sptr* (S-parameter translation). With no options specified, it will convert a spectre format file to an ADS file (e.g. `sptr spar.s2p spar.ads.s2p`). The program can also convert ADS format files to Cadence format. The program is in the same directory as the spectre executable, and should install with the spectre package.

Chapter 10: Using Switch Views, Stop Views and the Hierarchy Editor

This chapter provides information on using *switch views*, *stop views* and the *Hierarchy Editor*. A switch view is a list that describes the views to use and their priority. A stop view is a list that designates when to stop moving down in hierarchy. In other words, if a view is in the switch view list, and also in the stop view list, it won't traverse any lower in hierarchy.

RFIC Dynamic Link supports the concept of using *switch views* and *stop views*. Dynamic Link also supports Cadence's *Hierarchy Editor* tool, which enables more detailed specification of switch views and stop views than the standard Artist forms. Switch views and stop views are utilized by the netlister to expand hierarchy. The Hierarchy Editor enables you to override the switch view list and stop view list for each instance in a schematic's hierarchy.

Note The information provided in this chapter refers to using the Cadence Hierarchy Editor tool with the RFIC Dynamic Link and Advanced Design System. For more detailed information on using the Cadence Hierarchy Editor tool exclusively, refer to your *Affirma Analog Circuit Design Environment User Guide*. For more detailed information on expanding hierarchy in the Cadence environment, refer to the section on “*How the Netlister Expands Hierarchy*” in your Cadence documentation.

Expanding Hierarchy with the Dynamic Link Netlister

The flowchart shown in [Figure 10-1](#) describes the general process used by the RFIC Dynamic Link netlister to move through and expand the hierarchy in Dynamic Link.

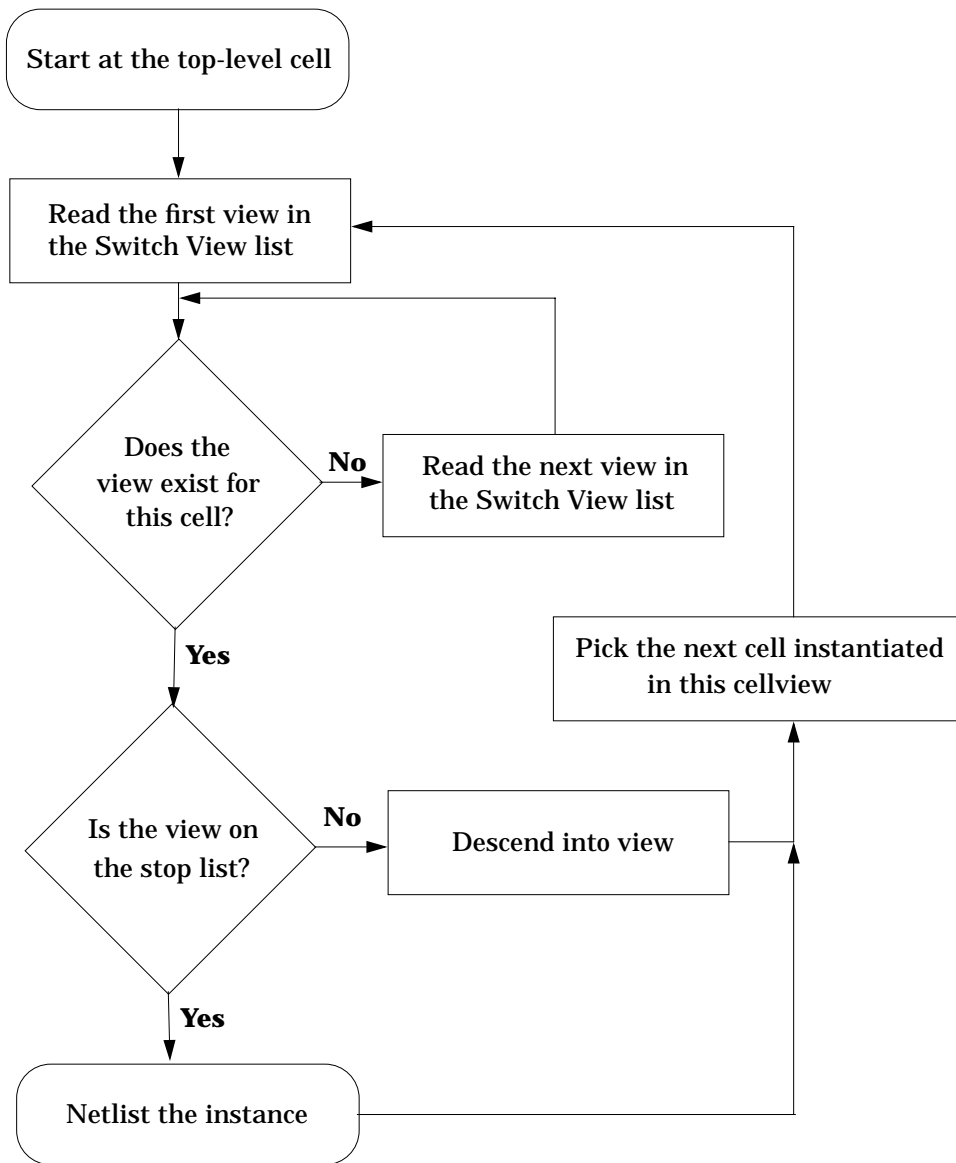


Figure 10-1. Netlist Hierarchy Expansion

In the Cadence schematic window, choose the **ArtistUtilities > Setup Options** menu item to access the *Setup Options* form. The Setup Options form enables you to designate a switch view list, and a stop view list.

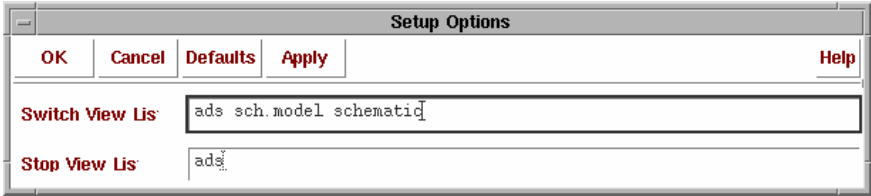


Figure 10-2. Setting a Switch View List and a Stop View List.

Each instance in a hierarchy has a master cell, that contains a number of views for that cell. Typically views are schematic, symbol, layout, extracted, etc. The switch view list is used to enable you to designate the priority of each view for hierarchical netlisting. The stop view is used to designate whether a view in the switch view list should be traversed for hierarchy. A stop view is implied to have no hierarchy. For an example of this, refer to the design *hierarchy_bottom* in the library *examples_lib* that is provided with the Dynamic Link installation.

In this example, the cell *hierarchy_bottom* has two alternate schematic views, *schematic_ideal* and *schematic_parasitics* (see [Figure 10-3](#) and [Figure 10-4](#) respectively).

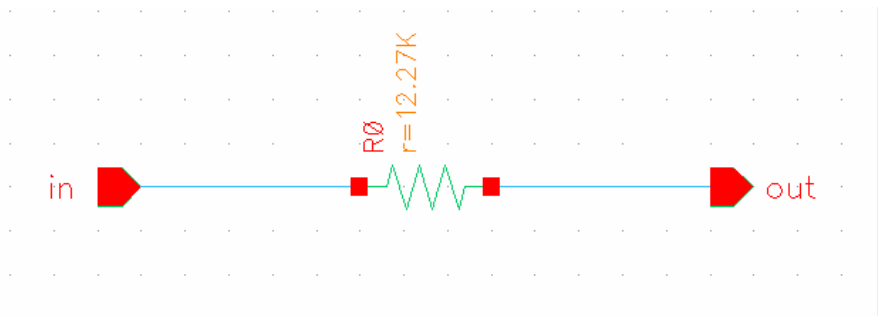


Figure 10-3. Alternate schematic view *schematic_ideal* for the example cell *hierarchy_bottom*.

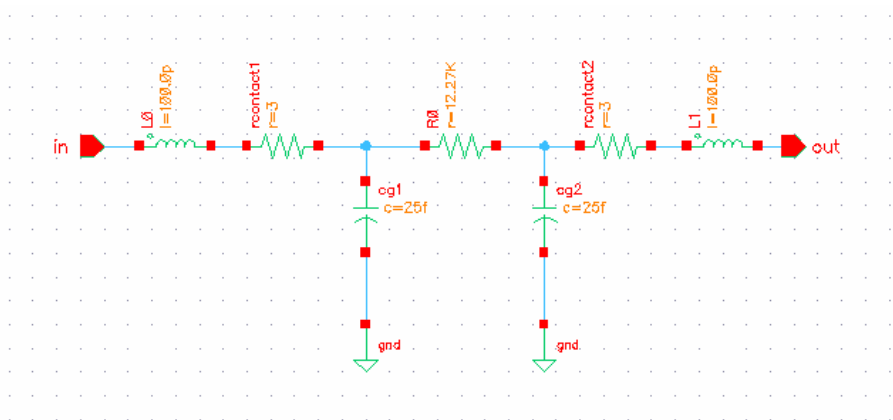


Figure 10-4. Alternate schematic view *schematic_parasitics* for the example cell *hierarchy_bottom*.

Two instances of the cell *hierarchy_bottom* have been placed in the schematic *hierarchy_top* (see [Figure 10-5](#)) in the *examples_lib*.

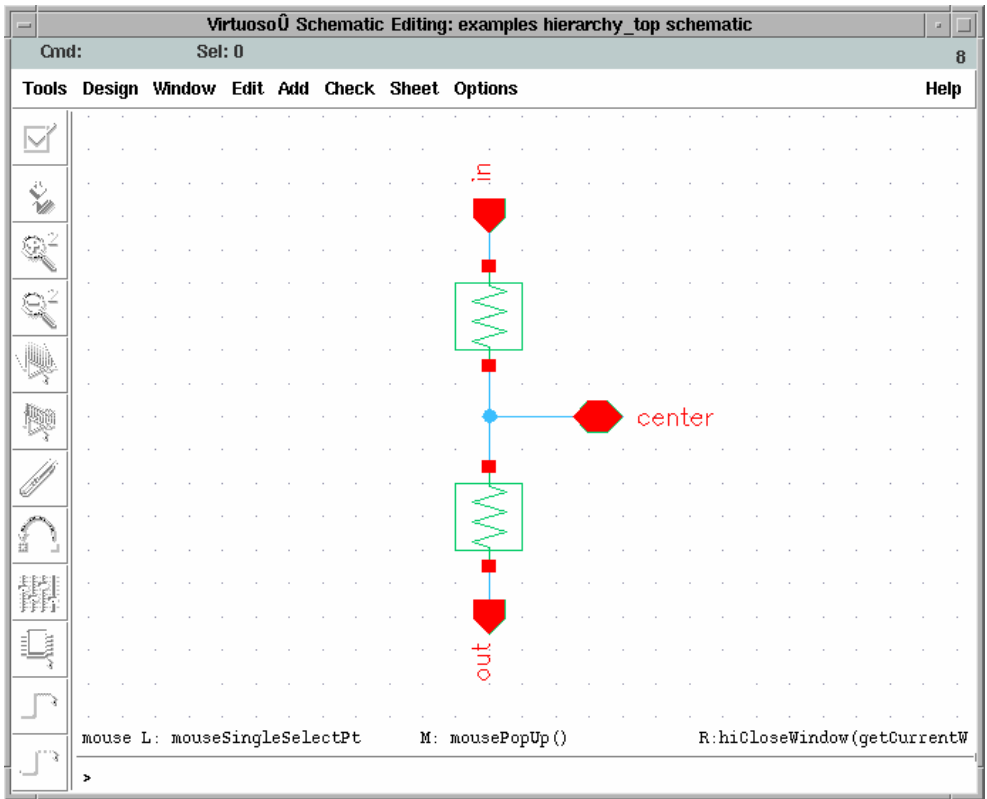


Figure 10-5. The schematic view for the example cell *hierarchy_top*

This cell is in turn placed in the ADS example project design *hierarchyTest*, with the schematic view chosen as the Cadence view (see [Figure 10-6](#)). Because there are two alternate schematics, it becomes necessary to tell the netlister which one to use when you netlist the design *hierarchy_top*. The switch view list is used to tell the netlister which one you want to use.

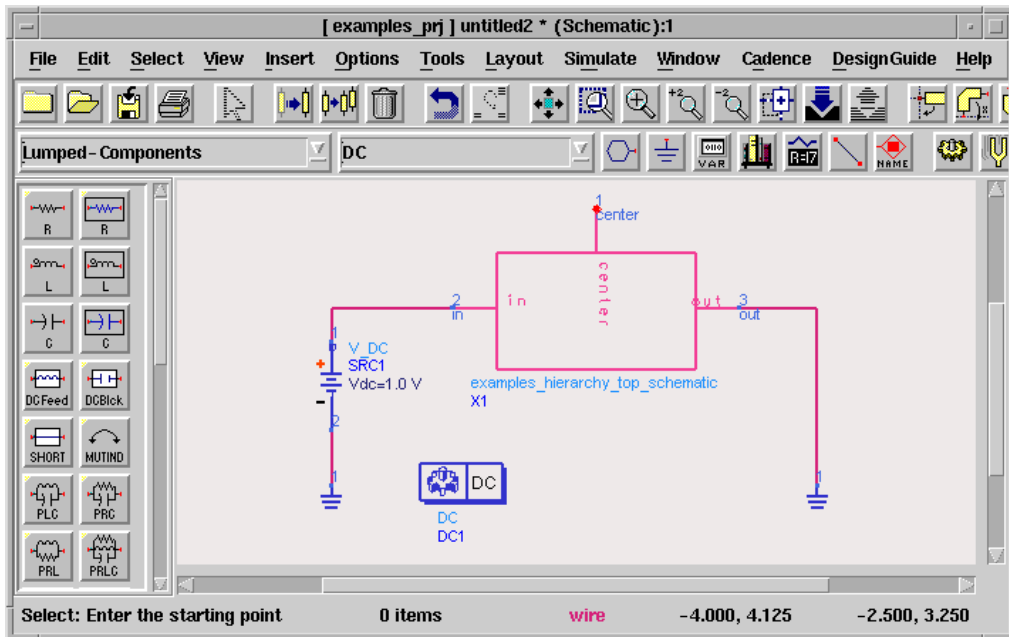


Figure 10-6. The cadence cell *hierarchy_top* placed in ADS

Figure 10-7 shows how the setup options have been changed to designate that *schematic_ideal* takes priority.

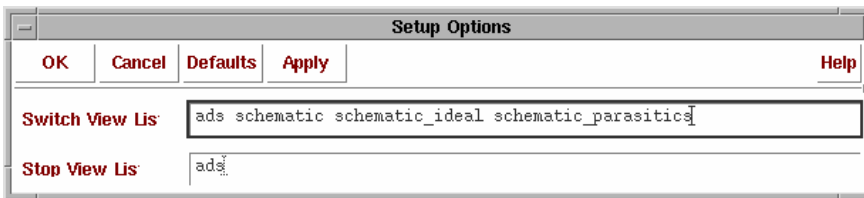


Figure 10-7. Switch View List with *schematic_ideal* taking priority

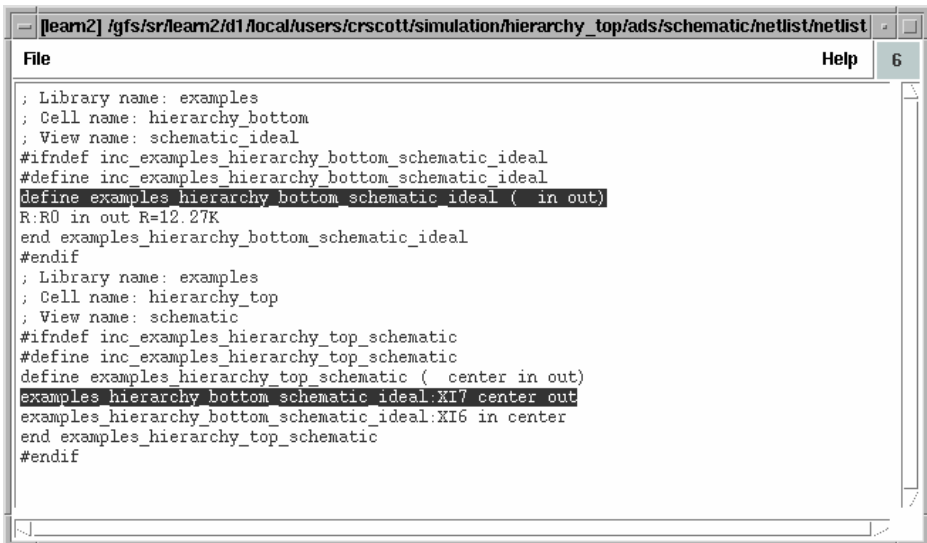
Referring to the Hierarchy Expansion flowchart in Figure 10-1, when an instance of *hierarchy_bottom* is encountered, the following occurs:

1. There is no *ads* view, so the next switch view is looked at.
2. There is no *schematic* view, so the code continues to the next switch view.

3. There is a *schematic_ideal* view. Since *schematic_ideal* is not in the stop view list, it is opened, so that a subcircuit can be netlisted for it.
4. With *hierarchy_bottom* as the top cell, now the first instance, and only instance encountered, is the analogLib *res* component. The switch view list is consulted.
 - Is there an *ads* view? Yes, there is.
 - Is the *ads* view a stop view? Yes, it is.

This means that the *res* component will not be opened as a schematic with hierarchy. Instead, it is meant to be a simulator primitive. Either a built in simulator component exists, which will be used, or a subcircuit definition has been included into the simulator that defines what the component is. A single component line is output for it, and the netlister continues on.

Figure 10-8 shows the resulting netlist from netlisting *hierarchy_top* with *schematic_ideal* having higher priority than *schematic_parasitics*. Note that the instances and subcircuit definition have been highlighted.



```
[learn2] /gfs/sr/learn2/d1/local/users/crscott/simulation/hierarchy_top/ads/schematic/netlist/netlist
File Help 6
; Library name: examples
; Cell name: hierarchy_bottom
; View name: schematic_ideal
#ifndef inc_examples_hierarchy_bottom_schematic_ideal
#define inc_examples_hierarchy_bottom_schematic_ideal
define examples_hierarchy_bottom_schematic_ideal ( in out)
R:R0 in out R=12.27K
end examples_hierarchy_bottom_schematic_ideal
#endif
; Library name: examples
; Cell name: hierarchy_top
; View name: schematic
#ifndef inc_examples_hierarchy_top_schematic
#define inc_examples_hierarchy_top_schematic
define examples_hierarchy_top_schematic ( center in out)
examples_hierarchy_bottom_schematic_ideal:XI7 center out
examples_hierarchy_bottom_schematic_ideal:XI6 in center
end examples_hierarchy_top_schematic
#endif
```

Figure 10-8. Netlist of the example cell *hierarchy_top* with *schematic_ideal* taking priority

If you prefer to use the *schematic_parasitics* schematic, the *hierarchy_top* schematic does not need to be changed. Instead, in the setup options form, you change the switch view list so that *schematic_parasitics* comes before *schematic_ideal* as shown in [Figure 10-9](#).

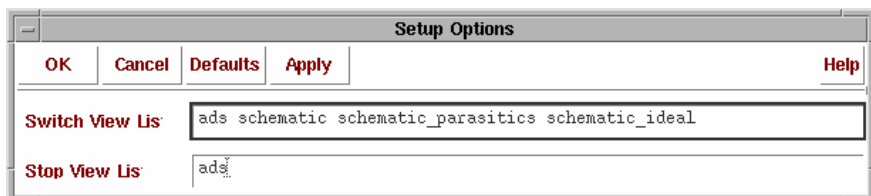
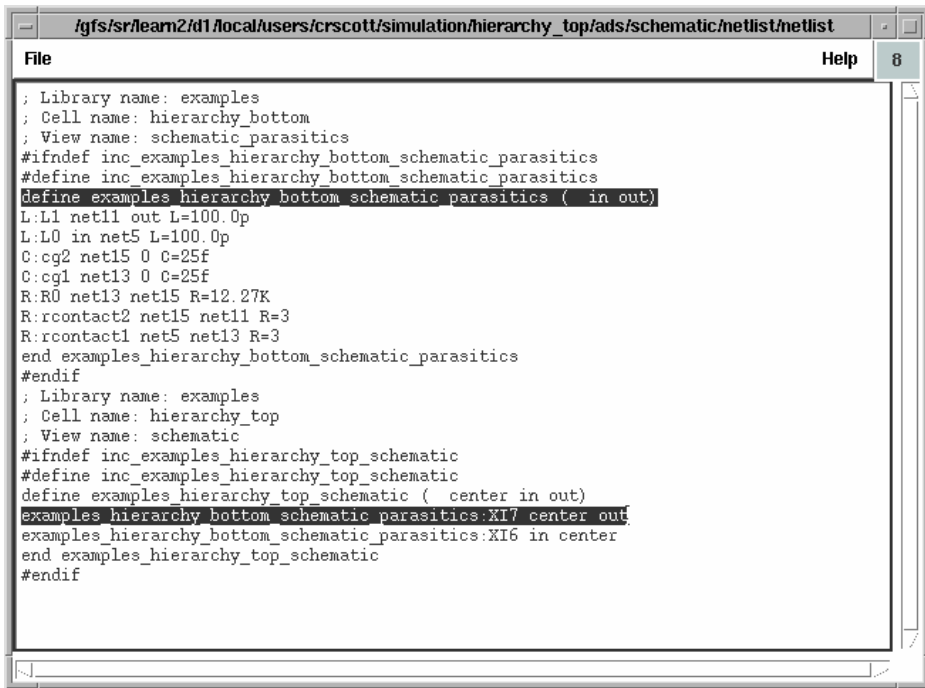


Figure 10-9. Switch View List with *schematic_parasitics* taking priority over *schematic_ideal*

Referring to the flowchart in [Figure 10-1](#) again, when an instance of *hierarchy_bottom* is encountered, the switch view list is checked. There is no *ads* view or *schematic* view. There is a *schematic_parasitics* view, which is not a part of the stop view list. This results in the netlister opening the *schematic_parasitics* view, which is netlisted as a subcircuit. The netlister then goes on to the next instance, without ever checking for a *schematic_ideal* view (in point of fact, it is not necessary to put *schematic_ideal* in the switch view list, since it will not be used). [Figure 10-10](#) shows the resulting netlist with the setup options from [Figure 10-9](#).

A screenshot of a text editor window showing a netlist file. The window title is "/tgs/sr/learn2/d1/local/users/crscott/simulation/hierarchy_top/ads/schematic/netlist/netlist". The text inside the editor is a netlist for a circuit with two hierarchical levels. The first level, "examples_hierarchy_bottom_schematic_parasitics", defines components like inductors (L:L1, L:L0), capacitors (C:cq2, C:cq1), and resistors (R:R0, R:rcontact2, R:rcontact1). The second level, "examples_hierarchy_top_schematic", defines a sub-circuit "examples_hierarchy_bottom_schematic_parasitics:XI7" and connects it to a center node "examples_hierarchy_bottom_schematic_parasitics:XI6".

```
; Library name: examples
; Cell name: hierarchy_bottom
; View name: schematic_parasitics
#ifndef inc_examples_hierarchy_bottom_schematic_parasitics
#define inc_examples_hierarchy_bottom_schematic_parasitics
define examples_hierarchy_bottom_schematic_parasitics ( in out)
L:L1 net11 out L=100.0p
L:L0 in net5 L=100.0p
C:cq2 net15 0 C=25f
C:cq1 net13 0 C=25f
R:R0 net13 net15 R=12.27K
R:rcontact2 net15 net11 R=3
R:rcontact1 net5 net13 R=3
end_examples_hierarchy_bottom_schematic_parasitics
#endif
; Library name: examples
; Cell name: hierarchy_top
; View name: schematic
#ifndef inc_examples_hierarchy_top_schematic
#define inc_examples_hierarchy_top_schematic
define examples_hierarchy_top_schematic ( center in out)
examples_hierarchy_bottom_schematic_parasitics:XI7 center out
examples_hierarchy_bottom_schematic_parasitics:XI6 in center
end_examples_hierarchy_top_schematic
#endif
```

Figure 10-10. Netlist of example *hierarchy_top* with *schematic_parasitics* taking priority

In all of the above examples, the stop view list was set to *ads*. This is the recommended stop view list to use for the RFIC Dynamic Link netlister. If you consult the netlist flowchart in [Figure 10-1](#), you will notice that it is not necessary for the stop view to be *ads*, any view can be used to designate that a part is a primitive. In the future, the RFIC Dynamic Link netlister will be expanded, so that alternate simulation definitions can be used based on which stop view is encountered (e.g. *ads* vs. *ads_ptolemy*). At present, the netlister always uses the simulation definition defined for *ads* in the cell's CDF, no matter which stop view is encountered.

Using the Hierarchy Editor with RFIC Dynamic Link

The Hierarchy Editor is a stand alone Cadence tool that enables switch views and stop views to be designated for each instance in a schematic hierarchy. For information regarding the hierarchy editor in general, consult your Cadence “*Hierarchy Editor User Guide*”.

When the Hierarchy Editor is used, a cell view specific to that tool is generated. The view itself is actually a text file, and cannot be opened directly in anything other than the Hierarchy Editor. The default view name for the Hierarchy Editor is *config*. A *config* view has a top cell view that it points to. All other information regarding switch views and stop views is then created based on the top cell view that is being pointed to. It is not necessary for the Hierarchy Editor view to be a part of the cell that it is pointing to, although in most cases this will be the simplest way to organize things.

Figure 10-11 displays the *Create New File* form used to create a Hierarchy Editor view. In this case, a Hierarchy Editor view is being made for the example cell *hierarchy_top* (see Figure 10-5). The goal is to set up this config view so that, during netlisting, it is possible to specify that one instance of *hierarchy_bottom* should use the *schematic_ideal* view, while the other instance uses the *schematic_parasitics* view.

Figure 10-11. Creating a Hierarchy-Editor view

After the new view is created, the Hierarchy Editor tool is started. For a new view, the *New Configuration* form appears as shown in Figure 10-12.

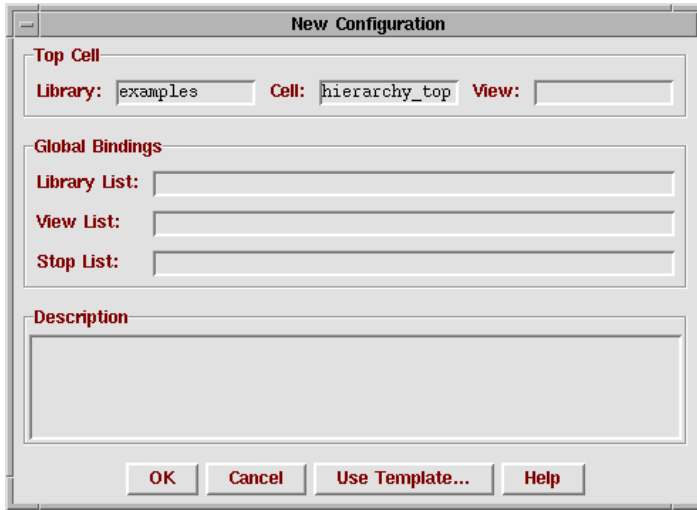


Figure 10-12. New Hierarchy Editor Configuration

Notice that all fields are left blank initially, except the *Library* and *Cell* fields which were specified in the *Create New File* form shown in [Figure 10-11](#). At this point, it is necessary to designate the top cell view, as well as the switch view list and stop view list. The *Library List* can normally be left blank. For more information on the *Library List* field, refer to your Cadence “*Hierarchy Editor User’s Guide*”.

It is necessary to fill in the Top Cell view. The top cell view should be a standard CDB view. From the standpoint of RFIC Dynamic Link, this means either a view that is edited with Virtuoso-schematic, or Virtuoso-layout. If you use a layout view, it should be an extracted view of some sort (i.e. the layout will contain connectivity and instances that can be traced back to schematic equivalents). Typically, you will put in either schematic or extracted as the view name.

The *Library List*, *View List*, and *Stop List* can be filled in by using a template. Templates contain default settings for particular simulators. To select a template, click the **Use Template** button in the *New Configuration* form. The *Use Template* form appears.



In [Figure 10-13](#), the ads template was selected, which has filled the View List in as *ads schematic extracted*, and the stop list as *ads*. These names represent the default view names for the tools that the Dynamic Link netlister supports. During netlisting, the view list and stop list will override the switch view list and stop view list that is specified using the Dynamic Link setup options dialog. Also, the Hierarchy Editor view will not consult the Dynamic Link options dialog to fill in the switch view list or stop view list, it is a completely separate tool, and must be set up independently.

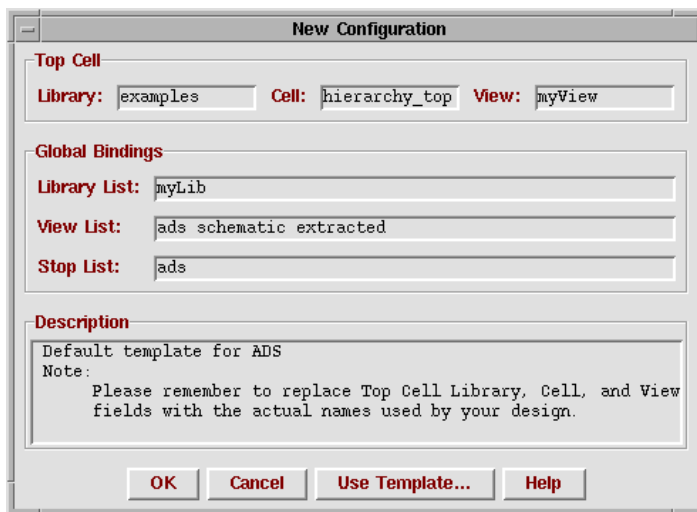


Figure 10-13. New Hierarchy Editor Configuration, using ADS template

When the new configuration is accepted, the main Hierarchy Editor window will be filled out. The main window shows the current Top Cell setting, as well as the global Library List, View List, and Stop List. In addition, the Hierarchy Editor will expand the hierarchy of the top cell, and show which views will be used for each instance within the top cell's hierarchy. When nothing has been overridden, the expansion will

follow the flow chart shown in [Figure 10-1](#). [Figure 10-14](#) shows how the example *hierarchy_top* schematic view was expanded using the global bindings. The Cell Bindings shows the two cells that were found, *hierarchy_top* and *hierarchy_bottom*. You may need to select **View > Update** to see the *hierarchy_bottom* Cell in the Cell Bindings list. The view found for *hierarchy_top* was schematic, this is a special case. Because it is the top cell, the library, cell, and view found will always be the same as what is specified for the top cell, regardless of the view list and stop list. The cell *hierarchy_bottom* says that the view found was ****NONE****. Because *hierarchy_bottom* has the views *schematic_ideal*, *schematic_parasitics*, and *symbol*, this is accurate. None of *hierarchy_bottom*'s views are in the global view list. If netlisting were attempted at this point, an error would result. The cell bindings give a visual indication of this.

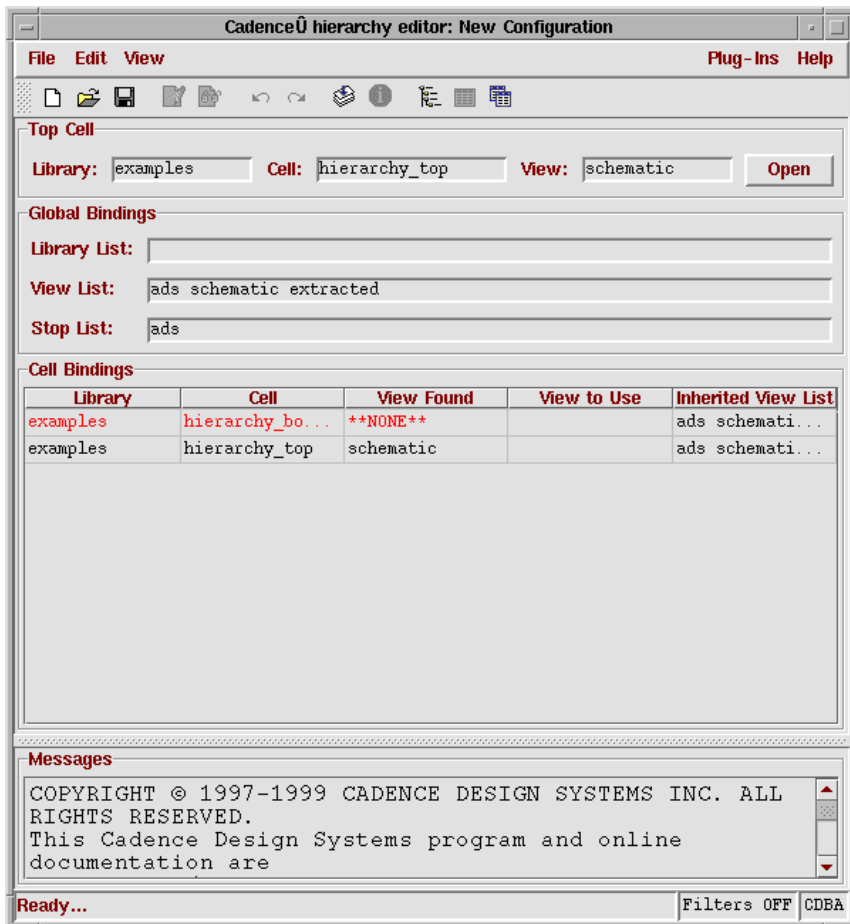


Figure 10-14. Initial Hierarchy Editor Main Window

In [Figure 10-15](#), the global view list has been changed, so that it now includes *schematic_ideal* and *schematic_parasitics*. The cell bindings are now updated to reflect the new hierarchy expansion. Because *schematic_ideal* precedes *schematic_parasitics* in the global view list, the expansion now says that, for *hierarchy_bottom*, *schematic_ideal* is the view found. Also, the analogLib *res* cell has been found, because *hierarchy_bottom* was expanded. This expansion happens because *schematic_ideal* is not listed in the global stop view list. Since *res* has an *ads* view, the expander decides that *ads* will be the view used for the *res* cell.

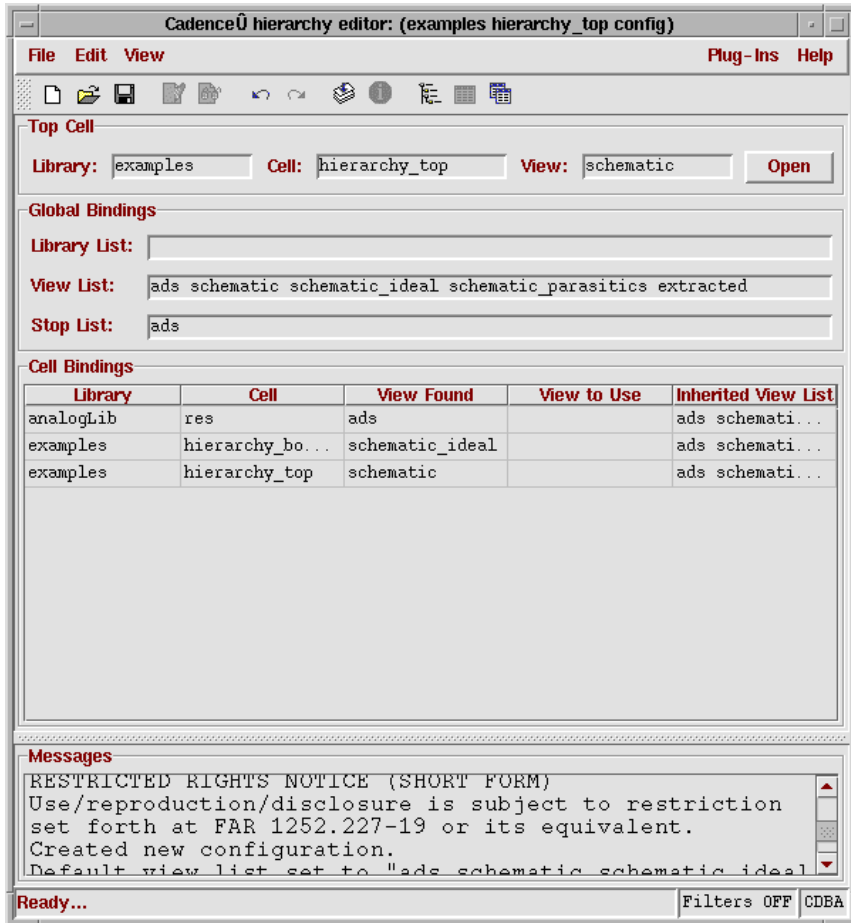


Figure 10-15. Hierarchy Editor with schematic_ideal in Global View List

So far, all of this demonstrates is that the Hierarchy Editor can be used to see how expansion would occur for a specified view list and stop list. That's nice, but it doesn't add any functionality over what was provided by the Setup Options form. What is needed is a way of overriding the view list. As it happens, the Hierarchy Editor is capable of doing just that.

In [Figure 10-16](#), the switch view list has been changed, so that *schematic_parasitics* is now first in the switch view list.

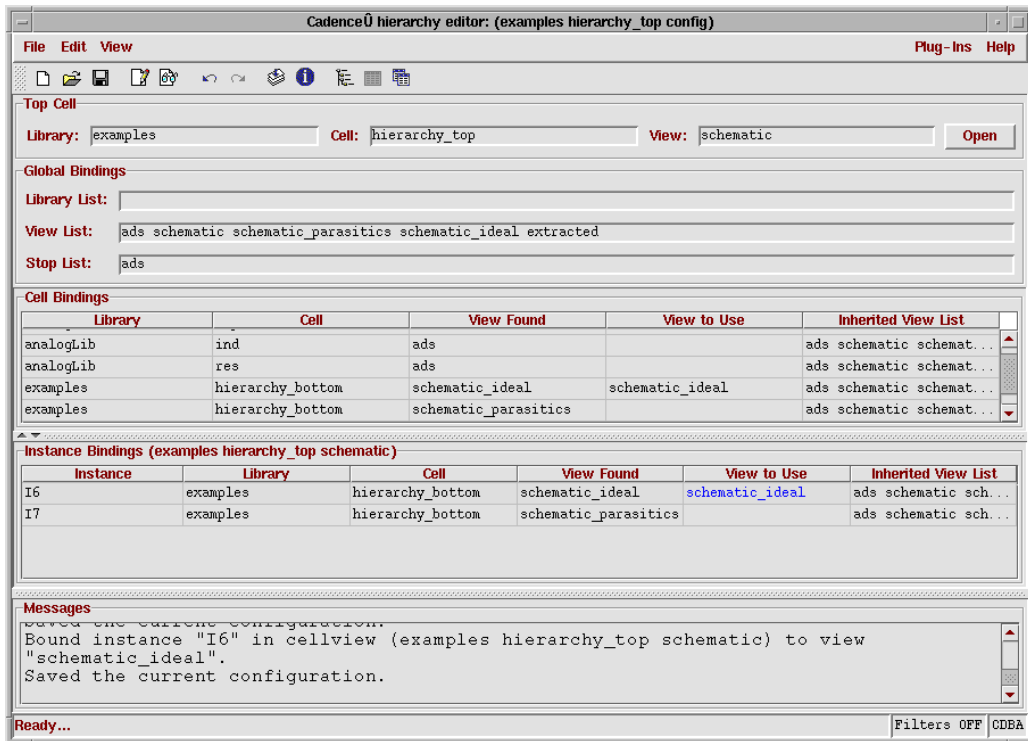


Figure 10-16. Hierarchy Editor with an instance view overridden

Notice that the instance bindings table is now displayed. When *hierarchy_top* is selected, the instance table shows all of the instances in the schematic. As it happens, *hierarchy_top* contains two instances of *hierarchy_bottom*, *I2* and *I3* (see [Figure 10-5](#)). If the global switch view list is used, both instances will expand to use *schematic_parasitics*. In this case, we have chosen to change the default behavior. In the view to use field, *schematic_ideal* has been specified. The view found field now specifies *schematic_ideal* instead of *schematic_parasitics*. Now, when the cell *hierarchy_top* is netlisted with this configuration, it will be necessary to expand the hierarchy for both *schematic_parasitics* and *schematic_ideal*. The netlister keeps track of the proper component name for the netlist.

Figure 10-17 shows the resultant netlist that is created using this configuration.

The image shows a screenshot of a text editor window titled "[beam2] /gfs/sr/beam2/d1/local/users/crscott/simulation/hierarchy_top/ads/config/netlist/netlist". The window has a "File" menu on the left and a "Help" button and the number "6" on the right. The main area contains a netlist file with the following content:

```
; Library name: examples
; Cell name: hierarchy_bottom
; View name: schematic_parasitics
; Inherited view list: ads schematic schematic_parasitics schematic_ideal
;extracted
#ifndef inc_examples_hierarchy_bottom_schematic_parasitics
#define inc_examples_hierarchy_bottom_schematic_parasitics
define examples_hierarchy_bottom_schematic_parasitics ( in out)
L:L1 net11 out L=100.0p
L:L0 in net5 L=100.0p
C:cg2 net15 0 C=25f
C:cgl net13 0 C=25f
R:R0 net13 net15 R=12.27K
R:rcontact2 net15 net11 R=3
R:rcontact1 net5 net13 R=3
end examples_hierarchy_bottom_schematic_parasitics
#endif
; Library name: examples
; Cell name: hierarchy_bottom
; View name: schematic_ideal
; Inherited view list: ads schematic schematic_parasitics schematic_ideal
;extracted
#ifndef inc_examples_hierarchy_bottom_schematic_ideal
#define inc_examples_hierarchy_bottom_schematic_ideal
define examples_hierarchy_bottom_schematic_ideal ( in out)
R:R0 in out R=12.27K
end examples_hierarchy_bottom_schematic_ideal
#endif
; Library name: examples
; Cell name: hierarchy_top
; View name: schematic
; Inherited view list: ads schematic schematic_parasitics schematic_ideal
;extracted
#ifndef inc_examples_hierarchy_top_config
#define inc_examples_hierarchy_top_config
define examples_hierarchy_top_config ( center in out)
examples_hierarchy_bottom_schematic_parasitics.XI7 center out
examples_hierarchy_bottom_schematic_ideal.XI6 in center
end examples_hierarchy_top_config
#endif
```

Figure 10-17. Netlist using configuration detailed in figure 10-15.

An alternate way of viewing the overridden expansion is to look at the tree view, as opposed to the cell and instance binding tables. This is shown in Figure 10-18.

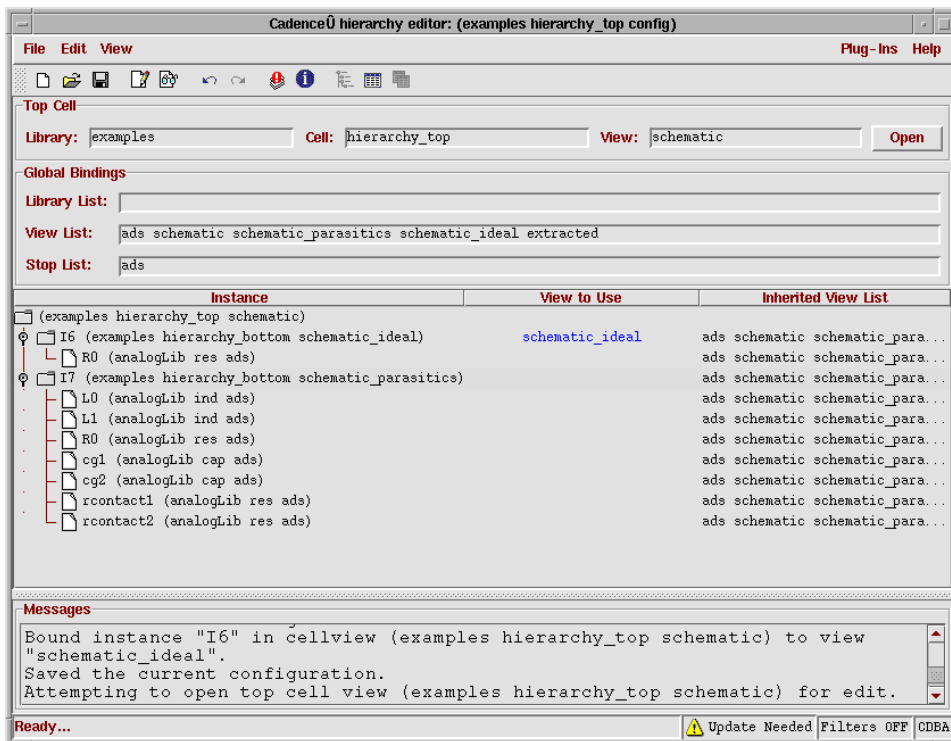


Figure 10-18. Tree view of hierarchy_top configuration with I2 set to use schematic_ideal.

The Hierarchy Editor makes it possible to override the hierarchy expansion on any instance. It is also possible to override the global switch view list on any instance. This allows you to change the expansion options for one tree of a hierarchy, while leaving all other trees intact.

Placing the config view in ADS

In order for a Hierarchy Editor configuration to be used during netlisting, it is necessary to place the Hierarchy Editor view in the ADS schematic. This is done by selecting the **Cadence** menu option. When the *Select Design* dialog appears, set the view name to the Hierarchy Editor view. In the *hierarchy_top* example, this is done by selecting the view name *config* (see Figure 10-19).

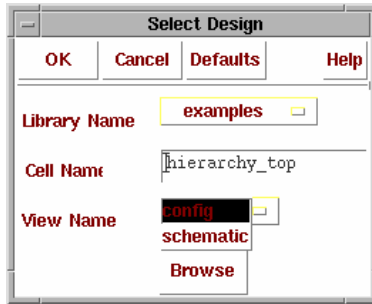


Figure 10-19. Selecting the Hierarchy Editor view for placement in ADS.

A new symbol is generated for the config view, and you are then able to place it. In the *hierarchy_top* example, the symbol graphics are inherited from the *hierarchy_top* cell. This results in a symbol that looks identical to the *hierarchy_top* schematic symbol for ADS. However, this symbol is now linked to the Hierarchy Editor view, as opposed to being linked to the schematic view.

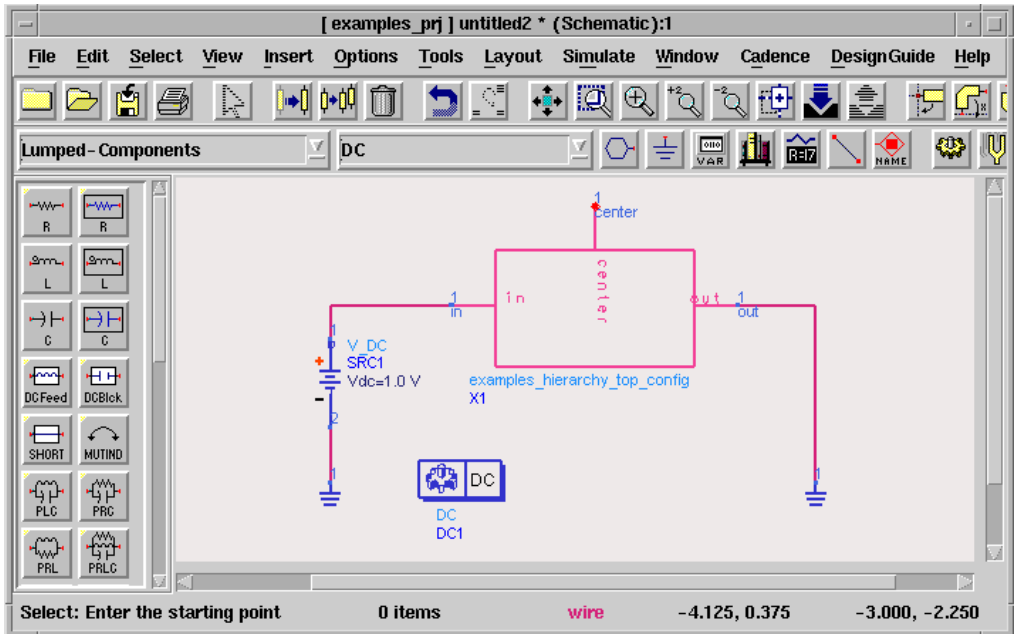


Figure 10-20. Hierarchy Editor test bench with config view placed

Figure 10-20 shows the new test bench where the config view is used instead of the schematic view. If the Cadence instance is selected, and you descend into it's hierarchy, the top cell view for the configuration will be opened. The configuration being used will be indicated in the title of the schematic/layout window that is opened. **Figure 10-21** shows the *hierarchy_top* schematic that will be opened. Note that the title indicates that the Config in use is *examples_lib hierarchy_top config*. This is important, as it means that hierarchy expansion will obey that particular configuration.

Note also that the top cell view does not need to remain constant. Thus, if you wish to do a simulation where the top cell schematic is used, and then do another simulation wherein an extracted view is used, you do not need to make multiple ADS symbols and then swap them. You can make a single Hierarchy Editor configuration, and swap the top cell view name. Once the test bench is set up, you do not need to modify anything in ADS to change your simulation. For the *hierarchy_top* example, a simulation would result in the netlist shown in **Figure 10-17**, based on the configuration shown in **Figure 10-16**.

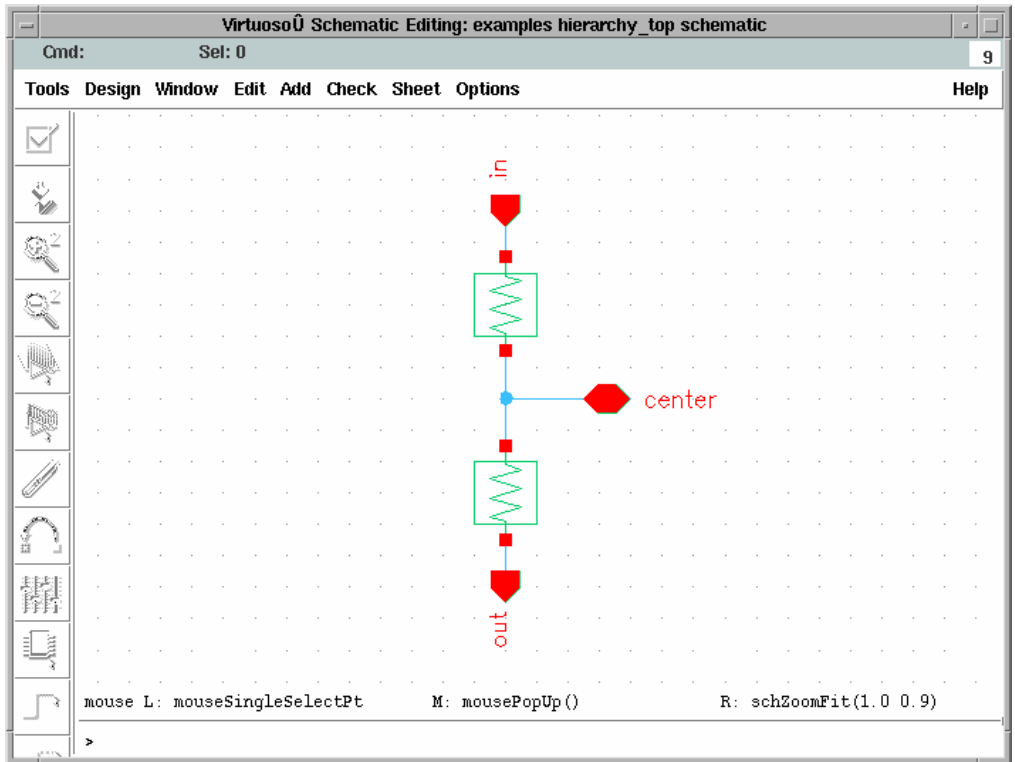


Figure 10-21. Schematic window attached to a Hierarchy Editor configuration

Appendix A: Command Reference and Troubleshooting

This appendix describes the function of each menu selection provided in Advanced Design System (Cadence Menu), the Cadence Schematic window (ArtistUtilities Menu) and the Cadence CIW (ADS Menu) while using the RFIC Dynamic Link.

Troubleshooting information that can help you resolve common problems is also provided in this appendix.

Cadence Menu

ADS Schematic window

Instance > Add Instance of Cellview...

Add a symbol of a Cadence design to the current Advanced Design System Schematic window. For example, see [“Adding a Symbol of the Cadence Cellview” on page 3-7](#) in Chapter 3 of the RFIC Dynamic Link User’s Guide.

Instance > Update Instance of Cellview

Discards `<ads_prj>/networks/<lib>_<cell>_<view>.dsn` and `<ads_prj>/networks/<lib>_<cell>_<view>.ael`, regenerates an ADS symbol from the Cadence symbol, and deletes the `$HOME/simulation/<cell>` directory. Deleting the `$HOME/simulation/<cell>` directory forces ADS to recreate a new netlist the next time ADS requires a netlist for a Cadence subcircuit.

Design Variables > Get Design Variables

Add design variables from a Cadence Cellview to an Advanced Design System Schematic window. For example, see [“Adding Design Variables” on page 3-9](#) in Chapter 3 of the RFIC Dynamic Link User’s Guide.

Design Variables > Update Design Variables to Cellview

Update the value of the Advanced Design System schematic design variables to the Cadence Cellview. For example, see [“Updating Cadence Design Variables” on page 6-2](#) in Chapter 6 of the RFIC Dynamic Link User’s Guide.

Add Netlist File Include

Enables duplication of the Definition, Stimulus, and Model Library File include used in Cadence/Affirma. For more information, refer to [“Using the Netlist File Include Component” on page 9-1](#).

Top-level Design Netlist

Generate and display the Advanced Design System subnetwork netlist for the Cadence design displayed in a particular Composer window. For example, see [“Viewing Netlists from Advanced Design System” on page 5-1](#) in Chapter 5 of the RFIC Dynamic Link User’s Guide.

Annotate > Annotate DC Solution to Selected Cellview

Display the DC node voltages, generated by the ADS simulator, on the Cadence Schematic Window. For example, see [“Performing a DC Simulation” on page 3-13](#) in Chapter 3 of the RFIC Dynamic Link User’s Guide.

Annotate > Annotate Operating Points to Selected Cellview

Display the DC operating points, generated by the ADS simulator, on the Cadence Schematic Window. For example, see [“Annotating DC Operating Points to a Selected Cellview” on page 3-15](#) in Chapter 3 of the RFIC Dynamic Link User’s Guide.

Close Connection

Closes ADS and terminates the link between Cadence and ADS. For example, see [“Ending the Session” on page 3-33](#) in Chapter 3 of the RFIC Dynamic Link User’s Guide.

ArtistUtilities Menu

Cadence Schematic Window

Setup Options ...

Set parameters such as Model Search Path (Cadence versions 4.4.2 and 4.4.3 only), Switch View List and Stop View List. The default values for this dialog are obtained from the configuration file. For more information, consult your Cadence documentation.

New Design ...

Select a new design to include in the Cadence Cellview. Set parameters such as Library Name, Cell Name and View Name. The dialog also offers a browse feature to search for existing designs. For more information, consult your Cadence documentation.

Design Variables ...

Modify the Component Description Format (CDF) information for a component so that it works with ADS. For example, see [“Adding and Editing Design Variables” on page 6-1](#) in Chapter 6 of the RFIC Dynamic Link User’s Guide. For more information, consult your Cadence documentation.

Save State ...

Save a state of a simulation. The simulation name, variables, model path, outputs and environment options are all available selections. For more information, consult your Cadence documentation.

Load State ...

Load an Analog Artist state. For more information, consult your Cadence documentation.

Subcircuit Netlist

Generate and display the Advanced Design System subnetwork netlist for the Cadence design displayed in a particular Cadence Schematic window. For example, see [“Viewing Netlists from the Cadence Schematic Window” on page 5-2](#) in Chapter 5 of the RFIC Dynamic Link User’s Guide. For more information, consult your Cadence documentation.

Note If the *ArtistUtilities* pull-down menu does not appear in the Cadence Virtuoso Schematic window, choose **Tools > ADS** in the Cadence schematic window.

ADS Menu

Cadence Command Interpreter Window (CIW)

There are two additional ADS menu items accessible from the Cadence CIW.

Start ADS

Launch Advanced Design System without opening a Cadence cellview.

Turn On ArtistUtilities

Adds the *ArtistUtilities* menu item to the Cadence schematic window with only the *Subcircuit Netlist* menu item enabled. This option enables you to create a Cadence subcircuit netlist without launching Advanced Design System.

Troubleshooting

All errors, warnings, and other messages are directed to the Cadence CIW. When a new message is written to the CIW, the window is raised to the top of your window stack so that new messages are always visible. Error messages may also be logged in a file, *idf.log*.

Some known problems and solutions are listed in the following section. You may find this information helpful in determining how to resolve a particular problem however, if you're unable to resolve a problem with the RFIC Dynamic Link using the information provided, contact Agilent EEsof-EDA customer support.

Known Problems and Solutions

Problem: By default, ADS does not create its own private color map, which may lead to unpredictable color behavior and/or menu buttons in place of icons.

Solution: Try one or more of the following:

- Set `HPEESOF_COLORMAP = private` in the ADS configuration file `$HOME/hpeesof/config/hpeesof.cfg` or `$HPEESOF_DIR/config/hpeesof.cfg`.
- Set `CDS_NUM_USER_COLORS = 16` in your `.profile` or `.cshrc` file.
- Restart Dynamic Link after exiting all other color-intensive applications.

Problem: When you remake a symbol, even when the old symbol is deleted, ADS does not allow you to create another symbol with the same name.

Solution: Click the instance in the ADS schematic window, then choose **Cadence > Instance > Update Instance of Cellview**.

Problem: There is no distinction between a design variable *X* from *Design A* and a design variable *X* from a different *Design B*.

Solution: Use unique design variable names for different designs, unless you really intend them to be the same variable, in which case there's no problem.

Problem: Symbol generation via Cadence symbol duplication does not reproduce arcs.

Solution: Use line segments instead of arcs.

Problem: *Could not spawn master program*. This message appears in your parent terminal window upon attempting to use the ADS link.

Solution: Ensure that `$HPEESOF_DIR/bin` is in your `PATH` and that `$HPEESOF_DIR/bin/idfmp` is a valid executable. If this does not work, ask your UNIX System Administrator to reboot your system or otherwise determine if a socket address is in use.

Problem: **Error* Could not find 'nlpglobals/ads' in library 'basic'. The nlpglobals' cell view is required. Netlisting aborted*. This error occurs either while netlisting in Analog Artist or after clicking *Simulate* in the ADS.

Solution: Copy the spectre view in the nlpglobals cell to create an ads view.

Problem: The UNIX environment does not set up properly when an in-house script for DFII is used.

Solution: Starting DFII using an in-house script may not set up the UNIX environment properly for RFIC Dynamic link. Work with your System Administrator to ensure that you understand what environment variables need to be set in the in-house script and modify your script accordingly.

Installation and Use Checklist

This section provides a checklist that can be used to help you resolve problems with RFIC Dyanmic Link. You can use the questions below to help determine what the cause of a particular problem might be.

- Is RFIC Dynamic Link installed? If yes, the following files/directories should exist:

```
$HPEESOF_DIR/bin/idf
$HPEESOF_DIR/bin/idfmp
$HPEESOF_DIR/circuit/symbols/idfSymbol.dsn
$HPEESOF_DIR/idf/ael/
$HPEESOF_DIR/idf/skill/4.4.6/ads.ini
$HPEESOF_DIR/idf/skill/4.4.6/ads.al
$HPEESOF_DIR/idf/skill/4.4.6/ads.cxt
$HPEESOF_DIR/idf/skill/4.4.5/ads.ini
$HPEESOF_DIR/idf/skill/4.4.5/ads.al
$HPEESOF_DIR/idf/skill/4.4.5/ads.cxt
$HPEESOF_DIR/idf/skill/4.4.3/ads.ini
$HPEESOF_DIR/idf/skill/4.4.3/ads.al
$HPEESOF_DIR/idf/skill/4.4.3/ads.cxt
$HPEESOF_DIR/idf/config/
$HPEESOF_DIR/idf/cdslib/4.4.6/analogLib/
$HPEESOF_DIR/idf/cdslib/4.4.5/analogLib/
$HPEESOF_DIR/idf/cdslib/4.4.3/analogLib/
$HPEESOF_DIR/idf/cdslib/4.4.6/basic/
$HPEESOF_DIR/idf/cdslib/4.4.5/basic/
$HPEESOF_DIR/idf/cdslib/4.4.3/basic/
$HPEESOF_DIR/idf/examples/
```

- Is Cadence configured for RFIC Dynamic Link? Run *idfConfigCadence -h* (RFIC Dynamic Link 2002) or *\$HPEESOF_DIR/idf/config/configCadence -h* (RFIC Dynamic Link 2001).

- Is `HPEESOF_DIR` set? Run *idfenv* (RFIC Dynamic Link 2002).
- Is `$HPEESOF_DIR/bin` in your `$PATH`?
- Is *icms* in your `$PATH`?
- Are *cds_root* and *cdsd* both in your `$PATH`?
- Is `PATH` and `LM_LICENSE_FILE` set in `~/.cshrc`, `~/.profile`, `~/.kshrc`, or other scripts sourced by these scripts? Execute *echo \$SHELL* or *finger* to display your login shell.
- Is your ADS *license.dat* file in the default `$HPEESOF_DIR/licenses/` directory or is it set in `$LM_LICENSE_FILE`?
- Is there a *trans_idf* (RFIC Dynamic Link 2002) or *Idf_c_interface* (previous versions of RFIC Dynamic Link) feature in the ADS *license.dat* file? Has this feature expired?
- Which item in the following list defines the Cadence license file?

`$CDS_LIC_FILE`

`<Cadence_install_dir>/share/license/clients`

`$LM_LICENSE_FILE`

`<Cadence_install_dir>/share/license/license.dat`

- Does the Cadence license contain the valid features listed below:

`OASIS_Simulation_Interface`

`34510`

`300 (only if using layout)`

- Are all seats of the above ADS and Cadence licenses taken?
- Does `$HPEESOF_DIR/idf/config/.cdsinit` exist?
- Is `$HPEESOF_DIR/idf/config/.cdsinit` loaded in `<Cadence_installation_directory>/tools/dfII/local/.cdsinit`?
- If the above file does not exist, is `$HPEESOF_DIR/idf/config/.cdsinit` loaded in `./cdsinit`?

- If the above files do not exist, is `$HPEESOF_DIR/idf/config/.cdsinit` loaded in `$HOME/.cdsinit`?
- If none of the `.cdsinit` files exists, `cp $HPEESOF_DIR/idf/examples/.cdsinit ./`
- Does the `.cdsinit` file that loads RFIC Dynamic Link's `.cdsinit` file change `PATH`, `CDS_INST_DIR`, `LM_LICENSE_FILE`?
- Does the `icms`, `icfb`, or `msfb` script change your `PATH` or `LM_LICENSE_FILE`? Run `idfenv` at the UNIX prompt, start Cadence, enter `system("idfenv")` in the Cadence CIW, and then compare the output of `idfenv` before and after the Cadence script is executed.
- What is in the `./cds.lib`? Do all of the libraries appear in blue in Cadence Library Path Editor form? If you are using the Agilent Technologies version of `analogLib` and `analogLib` appears in red, check to see if `IDF_CDS_VERSION` is used in `cds.lib` and it is not defined in the effective `.cdsinit` file.
- Does an `ads` view exist for the components in use? Use the Cadence Library Manager to check to see if the `analogLib/cap` cell contains an `ads` view.
- Does the `ads simInfo` section exist in the Cadence Component Description Format (CDF) for the components in use? Is all of the information in the `ads simInfo` correct? Use the Edit Component CDF form (**Tools > CDF > Edit**) or `cdfDump("<lib>" "/tmp/<cell>.cdf" ?cellName "<cell>" ?edit t)` function to see if the `ads simInfo` section exists in a cellview (leave out the `'?cellName "<cell>"` for dumping the library CDF instead of the cell CDF).
- Does the `ads` view exist for the global cellview `nlpglobals` under the `basic` library? If not, open the Cadence Library Manager, select `symbol` or `spectre` view, press middle mouse button, choose **Copy** from the popup list, and copy the `symbol` or `spectre` view to the `ads` view.
- If RFIC Dynamic Link failed to create an ADS netlist, can the Cadence netlister create a `spectre` netlist? Select **Tools > Analog Environment** from Cadence schematic window, select **Setup > Simulator/Directory/Host** from the Affirma DE window and ensure that `spectre` is selected as the Simulator, then select **Simulation > Netlist > Create** to generate a `spectre` netlist.
- Has the Cadence symbol changed since the instance was placed in ADS? Select the instance in the ADS 2002 schematic window, then choose **Cadence > Instance > Update Instance to Cellview** to recreate ADS symbol and remove the `~/simulation/<cell>/` directory to force Dynamic Link to regenerate the netlist. With RFIC Dynamic Link 2001, remove `<lib>_<cell>_<view>.*` under

<ads_prj>/networks/, delete ~/simulation/<cell>, then delete the instance in the ADS schematic and replace the instance.

- Is the *de_sim.cfg* file under the ADS project directory corrupted?
- Is there any local ADS customization in \$HOME/hpeesof/config/de_sim.cfg? Search for *AEL*.
- Is there site-wide ADS customization in \$HPEESOF_DIR/config/de_sim.cfg or custom/config/? For example, the line that reads "+ dynamic_link" in the *de_sim.cfg* file includes the file *dynamic_link.cfg*.
- If some components require models, are the associated ADS model files included with the *modelLibraryFiles* parameter in the *idfInclude* component?
- For debugging purposes, if the *./idf.log* file appears to be shorter than expected for debugging, is IDF_DEBUG_MODE set to TRUE in the UNIX environment?
- If the current GMT in *./idf.log* and *~/CDS.log* do not match, is there a lock on *~/CDS.log*? Is it *~/CDS.log.1* or *~/CDS.log.2* that is the Cadence log file for this session?
- Are there any suspicious options set in any of the following four Cadence *.cdsenv* files?
 - <cds>/tools/dfII/etc/tools/<application>/.cdsenv
 - <cds>/tools/dfII/local/.cdsenv
 - \$HOME/.cdsenv
 - ./cdsenv

The above files contain user preference options.

Glossary

ADS (Advanced Design System)

Advanced Design System is an EDA System for high-frequency circuit and system design.

AEL (Cadence Analog Expression Language)

In the Cadence context, AEL is the syntax and API (available in Skill or C) to support full or partial expression evaluation for repetitious circuit simulation.

AEL (ADS Application Extension Language)

This is a C-like interpretive programming language to configure, customize and enhance the Advanced Design System design environment.

Affirma Analog Circuit Design Environment

Cadence's interface for analog circuit design and analysis in versions 4.4.5 and 4.4.6.

bindkeys

Settings used to map individual keystrokes to a particular function within the software.

callback

A function or expression that gets evaluated when certain events occur; for example, clicking on a menu item.

CDF (Component Description Format)

The CDF is Cadence's mechanism to interactively define and evaluate parameters and attributes for individual components and designs.

CIW (Command Interpreter Window)

The CIW is Cadence's command window.

colormap

Indexed color table where each entry is a combination of R, G, and B pixel intensity values for UNIX X-windows display. Table size (number of colors) per software application is limited by the number of display bits per pixel, commonly eight.

DFII (Design Framework II)

Cadence's overall IC design environment.

EDA (Electronic Design Automation)

Software and services that give customers a distinct advantage by improving time-to-market, quality and productivity in the design of electronic products.

GUI (Graphical User Interface)

The interface between the user and the application.

HB (Harmonic Balance Simulation)

An iterative method of analysis that is based on the assumption that for a given sinusoidal excitation, there exists a steady-state solution that can be approximated to satisfactory accuracy using a finite Fourier series.

iPar()

The function used in an AEL expression for a parameter which is a function of another parameter of the same instance. For example, for MOSFET instances we might use $AD=iPar("w")*5u$.

IPC (Inter-Process Communication)

The protocol for passing messages between two or more processes.

OASIS

Open Analog Simulation Integration Socket. The procedural interface for simulator integration into the Cadence simulation environment.

optimization

Mechanism by which a simulator finds the optimal value of a global parameter within a user-supplied range of values.

OS (Operating system)

Such as, HP-UX, Solaris, AIX, Win95.

pPar()

The function used in a Cadence AEL expression for a parameter which is a function of some parameter of the parent instance. For example, for CMOS inverters we might use $W=pPar(wp)$ (where wp is a parent instance parameter) on

one of the pull-up FETs, enabling use of the same inverter symbol for different size inverters.

PSF

Parameter Storage Format. This is a Cadence-defined file format for storing complex structured data.

RFIC Dynamic Link (Dynamic Link)

RFIC Dynamic Link (Dynamic Link) for Cadence is an EDA framework integration software product. The product enables both *tops-down* and *bottoms-up* design and simulation in *Advanced Design System(ADS)* using IC designs from the Cadence database. RFIC Dynamic Link is based on IPC rather than data file translation maximizing data integrity and ease of use.

SKILL

Cadence's C/lisp-like interpretive programming language for framework and database integration.

testbench

Top-level schematic used to analyze a sub-circuit using a circuit simulator.

tuning

Mechanism by which a simulator can quickly re-simulate a circuit using new values for a number of parameters without having to re-input the netlist and recreate its data structures.

Index

A

- activating components, 3-13, 3-28
- adding
 - instance of cellview, 3-7, 4-2
 - netlist file include, 3-9
- ADS, 1-1, 1-2, 1-3
 - Data Display, 2-3, 3-14, 3-17, 3-22, 3-23, 3-25, 3-30, 5-1, 7-2
 - installation directory, 2-6
 - Main window, 2-12, 3-4, 4-1
 - Node Probing Setup, 3-20
 - project directory, 9-13
 - pull-down menu, 4-2
 - simulator, 9-14
- AEL, 9-13
- Agilent EEsof Installation Manager, 2-3
- Analog Artist, 1-2, 2-6, 2-10, 2-12, 3-29, 6-1
- analogLib library, 1-3, 3-1, 9-21
- annotating, 1-3, 3-15, 8-1
 - DC solution to selected cellview, 3-14
 - operating points to selected cellview, 3-17
- ArtistUtilities, 1-2, 4-1, A-3, A-4
 - menu, A-1

B

- bindkey settings, 2-12
- bundles, 1-3
- buses, 1-3, 9-14
- bus-ports, 1-3

C

- Cadence
 - AEL, 5-4
 - Analog Artist, 1-2, 2-6
 - ArtistUtilities, 1-2, 1-3, A-4
 - CDF, 9-18
 - cellviews, 1-2, 4-3, 6-2, 7-11, 9-10, 9-12, 9-13, 10-3
 - CIW, 3-1, 3-5, 5-2, 7-8, 9-11
 - design editor, 4-3
 - design environment, 3-4
 - design hierarchy, 7-8
 - design variables, 6-1, 7-6, 7-8
 - DFII, 1-2, 2-1, 2-11, 3-2, 3-6, 9-1
 - expressions, 5-4

- hierarchy editor, 1-3
 - install directory, 2-5
 - licenses, 2-2
 - menu, 1-3, 10-18, A-1
 - new versions, 2-7
 - schematic window, 1-3
 - simulation information, 9-19
 - Spectre, 9-21
 - subcircuit netlist, 9-11
 - Tool Filter, 2-5
 - Virtuoso Schematic Composer, 1-1, 1-2
 - cdsTerm, 8-6
 - Check and Save, 3-18
 - Close Cadence Connection, 3-33, 4-3
 - colors
 - allocating, A-4
 - commands
 - Cancel, 3-9
 - IdfMpsTuneEnd, 7-8
 - compatibility issues, 9-13
 - components
 - activating, 3-13
 - component options, 9-3
 - deactivating, 3-24
 - Goal, 3-28, 3-32, 7-9
 - Netlist File Include, 3-9, 9-1
 - Nominal Optimization, 3-28, 7-9
 - parameters, displaying on schematic, 9-3
 - VarEqn, 3-9, 3-28, 6-1
 - configuration, 1-3, 2-5
 - constants, 5-4
 - control elements, 9-13
 - corner analysis, 9-8
 - currents, 3-15
 - displaying, 8-5
- ## D
- data
 - file translation, 1-2
 - dataset, 9-14
 - DC
 - annotating currents, 8-2
 - annotating voltages, 8-1
 - simulation, 3-13
 - deactivating components, 3-24

- design, 1-2
 - environment, 1-1, 2-3
 - files, 9-13
 - selecting, 3-8
 - variables, 1-3, 3-31, 5-4, 6-1, 7-1, 9-13
- device operating point level, 3-16
- direct toolkit, 2-6
- directories
 - ael, 2-8
 - cdslib, 2-8
 - components, 2-8
 - config, 2-8, 2-9
 - Data, 8-6
 - directory names, 9-4
 - examples, 2-8, 3-1
 - idf, 2-8
 - model path, 2-9
 - models, 3-11
 - networks, 9-13
 - project, 2-10, 2-12
 - psf, 8-6
 - skill, 2-8
 - symbols, 2-8
- duplicate pin names, 9-13

E

- environment variables, 2-8, 2-9
 - IDF_ADS_PROJ_DIR, 2-8, 2-12
 - IDF_CADENCE_SYMBOL, 4-2
 - IDF_CONFIG_FILE, 2-8, 2-9
 - IDF_DEBUG_MODE, 2-8
 - IDF_EXPR_MAP, 5-5
 - IDF_LOG_FILE, 2-8
- equations, 9-4
- error messages, 3-29, 5-1, 9-2, A-4
- exclamation point suffix, 5-6
- expressions
 - variable, 6-2
- extracted views, 10-3, 10-20

F

- files
 - .cdsenv, 2-5, 2-7
 - .cdsinit, 2-9, 3-1
 - ads, 2-5, 2-7
 - ads.hierEd, 2-7
 - ads.ile, 2-5, 2-7
 - ads.ini, 2-12

- adsCdsenvFile, 2-7
- ADSLibconfig, 2-8, 9-21
- AEL, 2-11
- bindKeys.il, 2-12
- cds.lib, 3-1
- configuration, 2-9
- de_sim.cfg, 9-21
- emx.log, 2-8
- idf, 2-8
- idf.cfg, 2-8, 2-9, 2-12
- idf.log, 4-2
- idfConfigCadence, 2-8
- idfmp, 2-8
- idfSymbol.dsn, 2-8
- model suffix, 2-10
- mps.log, 2-8
- netlist.log, 5-1
- rficdl.library, 9-21
- freezing subcircuit netlists, 1-3, 2-11, 9-10
- functions, 5-4
 - bindkey, 3-24, 7-8
 - function names, 5-4
 - IdfSetBindKeys(), 2-12
 - non-mapping, 5-5

G

- Get Design Variables, 3-9
- global nodes, 5-6
- ground, 5-6

H

- Hewlett Packard, 2-1
- hierarchy, 4-1, 9-14, 9-20
 - editor, 1-3, 10-1, 10-10, 10-18, 10-21
 - netlist hierarchy expansion, 10-2
 - new configuration, 10-11
 - Push Into Hierarchy, 4-3
- HPEESOF_DIR, 2-6

I

- IBM, 2-1
- idfConfigCadence, 2-4, 2-6, 2-7
 - options, 2-6
- includePath, 9-4
- inherited connections, 9-20
- installation, 1-3, 2-3, 2-5, 9-21
 - Complete, 2-3
 - Custom, 2-3

- Typical, 2-3
- instance
 - bindings, 10-16
 - parameters, 7-1, 7-3, 7-8
 - properties, 9-17
- inter-process communication, 1-2
- iPar, 1-3, 9-14, 9-20

K
key mappings, 2-12

L
layout, 10-3
libraries, 1-2, 8-4

- analogLib, 1-3, 3-1, 9-21

licenses, 2-2
linearized device parameters, 3-15

M
mapping

- expressions, 2-11
- keys, 2-12

master cell, 10-3
menus

- ArtistUtilities, A-1
- Cadence, 3-5, 10-18, A-1
- Tools, 3-4
- tuning pop-up, 7-4, 7-8

messages

- error, A-4
- information, 3-11
- timeout, 2-11

model files, 9-21
modelLibraryFiles, 3-10, 9-6, 9-10

N
name mapping, 1-3, 5-1

- net and instance, 5-3

name space, 5-4
named nodes, 3-17
Net Name Prefix, 3-18, 9-14
Netlist File Include, 3-9, 9-9
netlisting, 1-3, 3-29, 5-1, 5-3, 9-14

- netlist, 9-13
- netlist filter, 2-10
- netlist include component, 9-1
- netlist suffix, 2-10
- netlistFile parameter, 9-12

- viewing netlists, 5-2
- node probing, 3-17

O
open

- design, 3-6
- file, 3-2

operators, 5-5
optimization, 1-3, 3-27, 3-32, 6-1, 6-2, 7-1

- using Optim component, 7-8

options

- site-specific, 2-9
- special, 2-11
- user-specific, 2-9

P
parameters

- displaying on schematic, 9-3
- instance parameters(iPar), 9-14
- parent parameters(pPar), 9-14

PATH, 2-6, 2-8

- prefixes, 9-4

PC, 9-12
pin names

- duplicate, 1-3

power, 3-15, 5-6
pPar, 1-3, 9-14
pre-processor directives, 9-8
programmable nodes, 9-20
projects, 1-2

R
redefinition errors, 9-2
RFIC Dynamic Link

- licenses, 2-2

S
schematic, 10-3

- ideal, 10-4
- parasitics, 10-4

scripts

- customized start-up, 3-2
- idfConfigCadence, 2-4, 2-6, 2-7

search path, 9-21
Section designator, 9-8
settings

- bindkey, 2-12

SETUP utility, 2-3

- shell variables
 - Korn shell, 1-2
- simulation, 1-1, 1-3, 3-14, 5-1, 9-13, 9-14
 - controller, 3-18
 - DC, 3-13
 - operating point, 3-15
 - displaying voltages or currents, 8-5
 - S-parameter, 3-24, 3-27, 3-32, 7-1
 - status window, 3-29

SKILL, 3-24, 7-8

S-parameter, 3-24, 3-27, 3-32, 7-1, 9-21

special options, 2-11

sptr, 9-21

statements

- #define, 9-8

- #endif, 9-8

- #ifdef, 9-8, 9-9

- #include, 9-8, 9-9

statistical analysis, 6-1

stderr, 4-2

stop views, 1-3, 10-1

subcircuit definition, 10-7

suffixes, 5-4

Sun, 2-1

supported platforms, 2-1

sweeps, 6-2

switch views, 1-3, 10-1, 10-3

symbols, 10-3, 10-19

- generating, 2-11, 8-7, A-5

- geometry, 4-2

- inherited, 9-17

- usage, A-5

system requirements, 1-3, 2-1

T

terminal output, 4-2

Tool Filter, 2-5

touchstone, 9-21

troubleshooting

- bundles, 9-14

- buses, 9-14

- bus-ports, 9-14

- debug mode, 2-10

- error messages, A-4

tuning, 1-3, 7-1, 7-2

- pop-up menu, 7-4, 7-8

- tune control, 7-3, 7-5

U

unnamed nets, 1-3, 9-14

updating

- cellviews, 7-11

- design variables, 6-2

UsePreprocessor, 9-8

utilities

- SETUP, 2-3

V

VarEqn component, 3-9, 3-28, 6-1, 9-13

variables, 7-11, 9-4

- design, 1-3, 6-1, A-5

- environment, 2-8

- shell, 1-2

- variable expressions, 6-2

views

- stop view list, 2-10

- switch view list, 2-10

Virtuoso Schematic Composer, 1-1, 1-2

voltages, 3-15, 3-17

- displaying, 8-5

W

warning messages, 9-14

Windows NT, 9-13