PrimeRail
User Guide

Version D-2009.12, December 2009

SYNOPSYS®

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

> "This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, CHIPit, Design Compiler, DesignWare, Formality, HDL Analyst, HSIM, HSPICE, Identify, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Physical Compiler, PrimeTime, SCOPE, Simply Better Results, SiVL, SNUG, SolvNet, Syndicated, Synplicity, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, the Synplicity logo, UMRBus, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Confirma, Cosmos, CosmosLE, CosmosScope, CRITIC, CustomExplorer, CustomSim, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, DFTMAX, Direct Silicon Access, Discovery, Eclypse, Encore, EPIC, Galaxy, Galaxy Custom Designer, HANEX, HAPS, HapsTrak, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance ASIC Prototyping System, HSIM^{plus}, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, StarRC, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (^{SM})

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.
ARM and AMBA are registered trademarks of ARM Limited.
Saber is a registered trademark of SabreMark Limited Partnership and is used under license.
All other product or company names may be trademarks of their respective owners.

Contents

12. Advanced Analysis

Preface

This preface includes the following sections:

What's New in This Release

Information about new features, enhancements, and changes; known problems and limitations; and resolved Synopsys Technical Action Requests (STARs) is available in the *PrimeRail Release Notes* in SolvNet.

To see the *PrimeRail Release Notes*,

1. Go to

   https://solvnet.synopsys.com/ReleaseNotes

   If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.
2. Select PrimeRail, then select a release in the list that appears at the bottom.

About This Guide

This document describes how to use the PrimeRail analysis capability. This guide is intended to be used in conjunction with PrimeRail online Help in which you can find detailed descriptions of all PrimeRail commands.

Audience

This manual is intended for integrated circuit designers who are concerned about voltage drop and electromigration problems during the physical design process and who have knowledge of UNIX and high-level design techniques.

Related Publications

For additional information about PrimeRail, see Documentation on the Web, which is available through SolvNet at

https://solvnet.synopsys.com/DocsOnWeb

Because PrimeRail uses some Milkyway commands and is tightly integrated with other Synopsys tools, like IC Compiler or PrimeTime PX, you might want to refer to the documentation for the following Synopsys products:

- Hercules
- HSIM^{plus}

- HSPICE
- IC Compiler
- Milkyway
- NanoSim
- PrimeTime PX
- PrimeTime SI
- StarRC

Conventions

The following conventions are used in Synopsys documentation.

Table 1 Conventions Used in Synopsys Documentation

Convention	Description
`Courier`	Indicates command syntax.
`Courier italic`	Indicates a user-defined value in Synopsys syntax, such as *`object_name`*. (A user-defined value that is not Synopsys syntax, such as a user-defined value in a Verilog or VHDL statement, is indicated by regular text font italic.)
`Courier bold`	Indicates user input—text you type verbatim—in Synopsys syntax and examples. (User input that is not Synopsys syntax, such as a user name or password you enter in a GUI, is indicated by regular text font bold.)
[ ]	Denotes optional parameters, such as *`pin1`* [*`pin2 ... pinN`*]
|	Indicates a choice among alternatives, such as `low | medium | high` (This example indicates that you can enter one of three possible values for an option: low, medium, or high.)
_	Connects terms that are read as a single term by the system, such as `set_annotated_delay`
Control-c	Indicates a keyboard combination, such as holding down the Control key and pressing c.
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services including software downloads, documentation on the Web, and “Enter a Call to the Support Center.”

To access SolvNet,

1. Go to the SolvNet Web page at

   https://solvnet.synopsys.com

2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.)

If you need help using SolvNet, click HELP in the top-right menu bar or in the footer.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a call to your local support center from the Web by going to https://solvnet.synopsys.com (Synopsys user name and password required), and then clicking “Enter a Call to the Support Center.”
- Send an e-mail message to your local support center.
  - E-mail support_center@synopsys.com from within North America.
  - Find other local support center e-mail addresses at http://www.synopsys.com/Support/GlobalSupportCenters/Pages
- Telephone your local support center.
  - Call (800) 245-8005 from within the continental United States.
  - Call (650) 584-4200 from Canada.
  - Find other local support center telephone numbers at http://www.synopsys.com/Support/GlobalSupportCenters/Pages

1

Introduction to PrimeRail

The PrimeRail tool is the Synopsys next generation reliability tool that provides both static and dynamic analysis capabilities in analyzing voltage (IR) drop or electromigration problems of the design. The PrimeRail tool enables you to locate and eliminate possible voltage drop and electromigration violations for system-on-chip (SoC) designs. No matter what type of design it is, PrimeRail can provide accurate power rail analysis results with sign-off quality.

This chapter offers an overview of PrimeRail analysis capabilities, including the following sections:

Features and Benefits

As technology feature sizes shrink, conductors get smaller, and supply voltages are reduced, the corresponding current per scaled feature size increases exponentially. These changes cause power-rail voltage drop and electromigration effects that significantly degrade performance and might cause the circuit to malfunction. To avoid such problems, PrimeRail provides a design-stage comprehensive sign-off solution for power consumption, voltage drop, and electromigration analysis for advanced technology designs.

Built on StarRC, HSPICE, NanoSim, HSIM^{plus}, and PrimeTime PX sign-off technologies, PrimeRail offers gate-level and transistor-level time-average and dynamic voltage drop and electromigration analyses with on-chip decoupling capacitance and full-chip sign-off with package parasitics. The PrimeRail integration with the Galaxy Design Platform allows designers to achieve large design convergence and a predictable path to sign-off.

PrimeRail is a full-chip solution for time-average and dynamic voltage drop and electromigration analysis. It extends the Synopsys sign-off solution in the Galaxy Design Platform to power networks. With PrimeRail, the Galaxy Design Platform now provides a comprehensive solution for timing, signal integrity, and power network sign-off.

When analysis is complete, color highlighting shows the problem areas in an easy-to-use graphical user interface. The interactive what-if analysis feature enables you to optimize IR drop and electromigration effects by inserting user-defined taps or nonideal voltage sources and parasitics to the design. Early rail analysis can be done before the design is complete—for example, when only the power routing is done or right after the placement stage.

The PrimeRail matrix solver employs the Dynamic-Macromodeling technology to rapidly analyze complex, hierarchical, multimillion-transistor, full-chip designs with less memory usage and improved turnaround time without sacrificing accuracy.

Additionally, PrimeRail does the following:

Hybrid Technology

An innovative engine delivers gate-level and transistor-level time-average and dynamic voltage drop and electromigration analyses within 1 to 2 percent of HSPICE accuracy. It provides full-chip performance and capacity for arbitrary resistance-inductance-capacitance (RLC) networks. Its high-speed and memory-efficient architecture enables analysis of 20M gates in 8 hours with minimized memory usage. It supports design formats in Milkyway, LEF/DEF, or GDSII.

Accurate Dynamic Modeling of Memories and Analog

The PrimeRail simulation and modeling capability delivers accurate memory and analog models with HSPICE correlation. Memories in GDSII format are simulated using NanoSim or HSIM^{plus}; StarRC is used for power ground and signal parasitic extraction to provide highest accuracy.

Built on the Milkyway database

PrimeRail is built to use the Milkyway design database—it can directly access critical data without time-consuming and error-prone translation steps, significantly reducing design time. The Milkyway database facilitates convergence of critical design properties, including timing, area, power, and signal integrity.

The Synopsys Milkyway database is a widely used database for IC design implementation and advanced technologies extending to 90 nm and below.

Flexible Design Database Interface

The PrimeRail design database accepts various industry-standard formats, including Milkyway, Library Exchange Format (LEF), and Design Exchange Format (DEF). GDSII is also acceptable for hard macros and pad cells.

Strongly Integrated with All ASIC Design Flows and Other Synopsys Tools

The PrimeRail tool is tightly integrated with other Synopsys products, such as IC Compiler, StarRC, PrimeTime PX, PrimeTime SI, and Hercules tools. PrimeRail also accepts tool data from third-party vendor products as inputs and calculates power consumption or rail violations in a design.

PrimeRail, IC Compiler, Hercules, and StarRC can all read and write to the common Milkyway database for tight flow integration and delivery of high performance and accuracy. Besides from using PrimeRail as a standalone tool, you can invoke PrimeRail within IC Compiler for cell-level static and dynamic rail analysis.

Flexible Postanalysis Features

PrimeRail enables you to view analysis results graphically in the voltage drop and electromigration maps. Different colors can be assigned to indicate various step levels of voltage rise and drop or current density limits. You can control design hierarchy if the design consists of overlapping layouts.

The visibility control over layout layers and design levels enables you to easily investigate possible violation problems and improves the map drawing speed.

Scheme Programming Language Support

Scheme is a high-level programming language that provides an extensive set of data types and flexible control structures to be used as an interface for PrimeRail. It supports operations for the following types of data:

- Characters
- Lists or vectors of objects

- Numbers
- Strings
- Vectors

You can develop scripts with multiple Scheme functions and commands to accomplish a series of sequential tasks. For more information, see “Using Command Files” on page 2-8.

PrimeRail and Other Synopsys Tools

The PrimeRail analysis feature interacts with the following Synopsys tools:

IC Compiler

IC Compiler provides the capability of analyzing cell-level reliability effects, displaying maps and browsing error cells without leaving an IC Compiler session. This allows you to find and fix possible reliability problems at an early place and route stage without leaving the IC Compiler environment. You can also generate the PrimeRail scripts and use them to perform rail analysis in the standalone version of PrimeRail.

PrimeRail is also able to read in the rail setup file generated by the IC Compiler `create_rail_setup` command, and it verifies the accuracy and consistency of input and design data before performing power and ground net extraction, power analysis, or rail analysis in the standalone version of PrimeRail.

HSIM^{plus}

PrimeRail uses HSIM^{plus} to perform transistor-level circuit simulation on memories and analog blocks for creating dynamic models.

HSPICE

PrimeRail adopts the HSPICE technology to generate current waveform power libraries and intrinsic parasitics libraries automatically during library characterization.

Hercules

PrimeRail uses Hercules to create the CONN view for hard macros, improving data preparation performance.

NanoSim

PrimeRail uses NanoSim to perform transistor-level circuit simulation on memories and analog blocks for creating dynamic models.

PrimeTime PX

PrimeRail adopts the PrimeTime PX technology to generate the necessary power and timing information.

PrimeTime SI

PrimeTime SI supports the cell-instance-based voltage drop values that PrimeRail generates for IR-drop derated timing analysis.

StarRC

PrimeRail uses the StarRC extraction engine to generate signal net parasitics for power analysis and the parasitics of power and ground nets for rail analysis.

Supported Platforms

The PrimeRail software runs on various UNIX and Linux platforms. For detailed information about supported platforms, see the information listed at the following web address:

http://www.synopsys.com/products/sw_platform.html

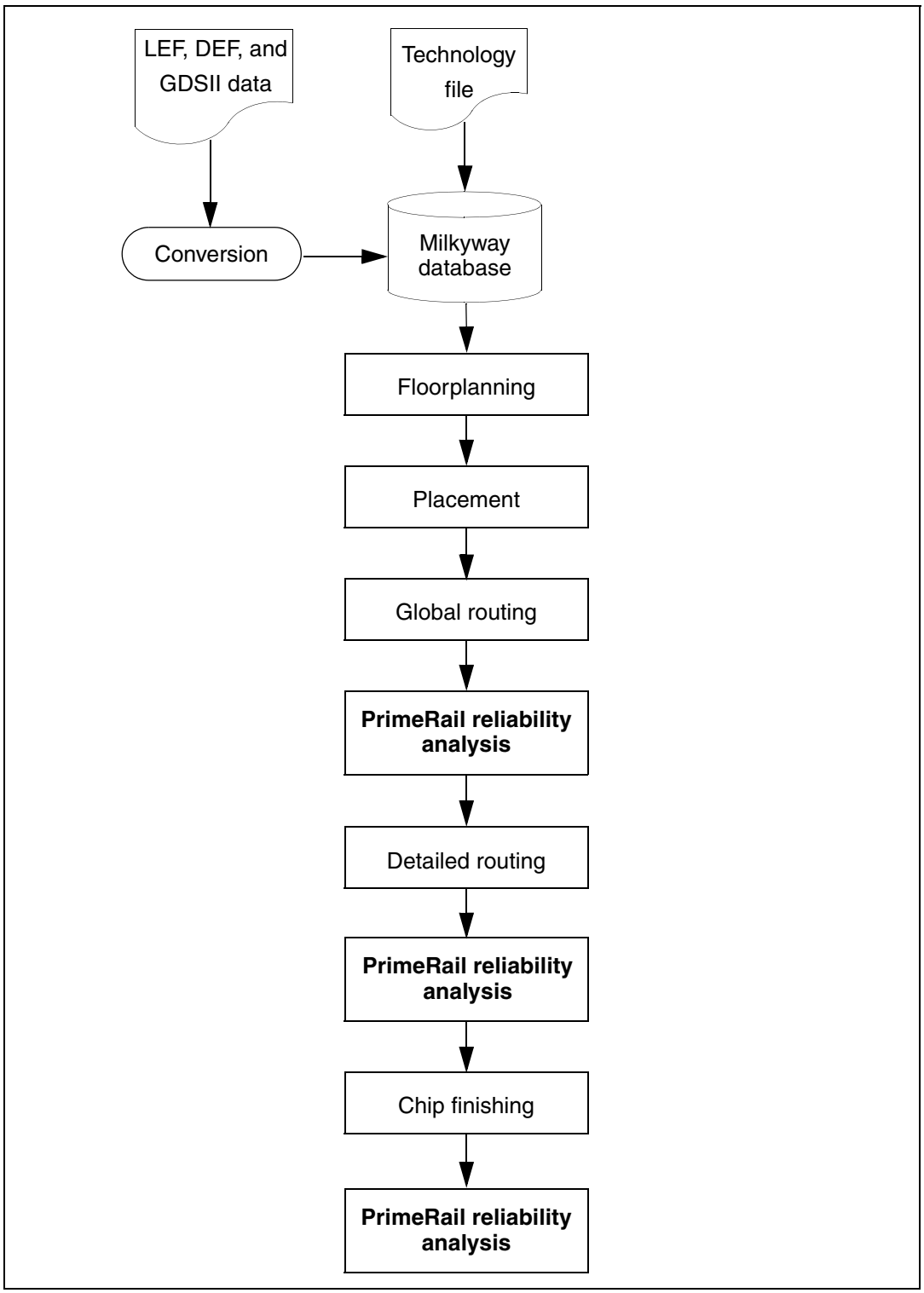
Using PrimeRail in the Design Flow

You can use PrimeRail at different points in the design flow after global routing is performed. When global routing is completed, use PrimeRail to verify the power and ground network robustness, to calculate total power consumption and to check for voltage drop and electromigration violations on a circuit. This capability enables you to detect potential violations before you perform detailed placement and routing and thus significantly reduces turnaround time later in the design cycle.

Alternatively, you can check the reliability effects of the design after detailed routing or chip finishing. Use PrimeRail to verify the reliability of your design for sign-off.

Figure 1-1 shows when to use PrimeRail in a circuit's design flow.

Figure 1-1 Using PrimeRail in the Design Flow



In-Design Rail Analysis in IC Compiler

The PrimeRail tool is tightly integrated with IC Compiler; you can run rail integrity checking, voltage drop analysis, and electromigration analysis in the IC Compiler place and route environment. Advanced analysis features, such as decoupling capacitor analysis and in-rush analysis, are also supported in the IC Compiler environment. When checking and analysis processes are finished, you can display the generated resistivity map, voltage drop map, and electromigration map to find the problematic areas and perform implementation correlation without leaving the current session in IC Compiler. Error cells are also available for display through the IC Compiler error browser.

IC Compiler and PrimeRail provide the `analyze_rail` command for invoking PrimeRail and for performing PrimeRail rail analysis within IC Compiler.

The following examples show how to use the PrimeRail checking and analysis commands within IC Compiler:

- Execute the `analyze_rail -integrity` command to perform rail checking on the current editing design. Possible missing vias and connections will be reported in the report file. You can fix the problems based on the generated reports and recommendations.
- Execute the `analyze_rail -voltage_drop` command to perform voltage drop analysis on the design and to post-process the results for generating a voltage drop map. When analysis is complete, display the generated map from the IC Compiler graphical user interface, check the highlighted violations in the map, and fix the design, based on the recommendations.
- Execute the `analyze_rail -electromigration` command to perform rail analysis on the design and to post-process the results for generating an electromigration map. Violations and possible suggestions for fixes will also be reported. When analysis is complete, display the generated maps from the IC Compiler graphical user interface, check the highlighted violations in the map, and fix the design, based on the suggestions.
- Execute the `analyze_rail -decap` command to perform decoupling capacitor insertion analysis on the specified net. Execute this option when you want to solve power supply noises by adding decoupling capacitance (decap) cells in the affected areas.
- Execute the `analyze_rail -inrush` command to perform in-rush analysis with power management cell implementations in the IC Compiler environment.

For more information, see Chapter 3, “In-Design Rail Analysis in IC Compiler,” in this user guide.

PrimeRail Analysis Flows

The PrimeRail technology consists of various analysis flows, as discussed in the following sections:

- Cell-Level Analysis
- Cell-Level Advanced Analysis
- Transistor-Level Dynamic Analysis
- Transistor-Level Signal Net Electromigration Analysis

Cell-Level Analysis

In PrimeRail, the cell-level analysis is based on the PrimeTime and PrimeTime PX sign-off technology. PrimeRail uses its built-in power and ground net extraction engine to extract the parasitic information of the power and ground network. The rail analysis engine analyzes voltage drop and electromigration violations of the design. The cell-level analysis in PrimeRail supports both time-average and dynamic analyses.

For sign-off accurate dynamic analysis results, you need to have either Composite Current Source Power (CCS Power) or built-in library characterization for standard cells and dynamic macro models (DWMs) for memory and analog blocks.

The built-in library characterization is done with industry standard HSPICE simulator and does not require information about the design. You run the characterization once as the part of reference library creation. The results are accessible through an ASCII interface, and simulation runtime is modest.

For more information about performing library characterization and cell-level power and rail analysis, see Chapter 6, “Library Characterization for Cell-Level Dynamic Analysis,” and Chapter 7, “Performing Cell-Level Analysis,” in this user guide.

Cell-Level Advanced Analysis

In addition to the basic time-average and dynamic voltage drop and electromigration analysis, PrimeRail provides several advanced analysis capabilities for both block-level and full-chip designs, including multithreshold-CMOS design analysis, in-rush analysis, decoupling capacitor insertion, power and ground core model generation, what-if analysis, and so on.

For a detailed description about conducting advanced analyses, see Chapter 12, “Advanced Analysis,” in this user guide.

Transistor-Level Dynamic Analysis

The dynamic transistor-level analysis in PrimeRail utilizes the established Synopsys dynamic verification products. The flow integrates StarRC for power ground network parasitic extraction and NanoSim (and HSIM^{plus}) for current waveform generation of transistors. PrimeRail provides the `poTxPowerAnalysis` command that enables you to configure options for extraction and simulation processes in a single user interface.

The next step is to run the `poRailAnalysis` command that checks peak voltage drops by simulating the RLC (resistance-inductance-capacitance) power supply network. This is to simulate the node voltages and branch currents in the power net and generate dynamic voltage waveforms. With the generated voltage waveforms, you are able to investigate where a peak voltage drop or rise occurs in a device terminal.

For a detailed description of performing the transistor-level dynamic analysis, see Chapter 9, “Performing Transistor-Level Dynamic Analysis,” in this user guide.

Transistor-Level Signal Net Electromigration Analysis

The transistor-level signal net electromigration analysis flow in PrimeRail also utilizes the technologies of the StarRC and NanoSim tools. In PrimeRail, the `poSigDynamicAnalysis` command automatically invokes StarRC and NanoSim for extraction and simulation. You do not need to run each of the tools individually.

For a comprehensive, detailed data flow for conducting a signal net electromigration analysis at the transistor level, see Chapter 13, “Performing Transistor-Level Signal Net Electromigration Analysis,” in this user guide.

2

Working With PrimeRail

This chapter provides information about the concepts and tasks involved when working with PrimeRail.

This chapter contains the following sections:

Starting PrimeRail

You can access PrimeRail from the UNIX command line.

To facilitate opening libraries, start the application in the directory containing your libraries. PrimeRail provides sample Milkyway libraries, which you can find in the directory where PrimeRail is installed.

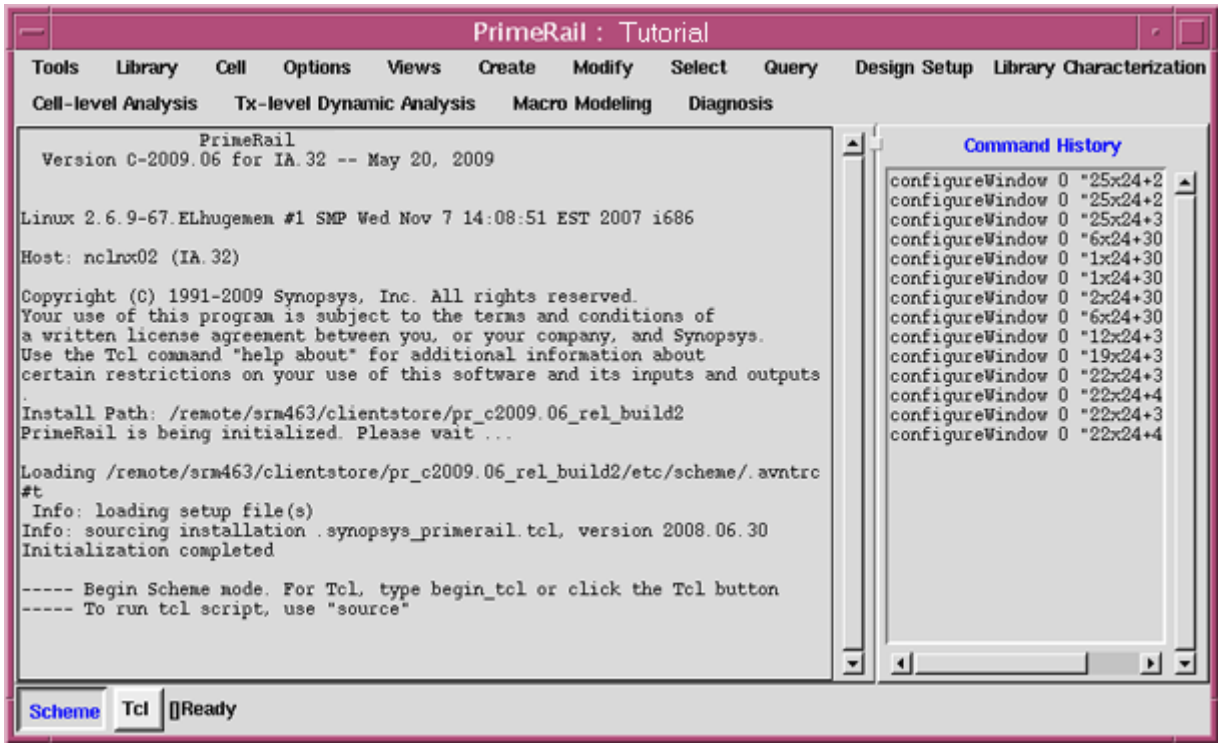
Starting the PrimeRail GUI

To start the PrimeRail application, enter the following at the UNIX prompt:

```
%PrimeRail
```

The PrimeRail window appears.

Figure 2-1 Invoking PrimeRail



Command Options

You can use command options when you invoke the PrimeRail application.

To see the command options PrimeRail provides, enter the following at the UNIX prompt:

```
%PrimeRail -help
```

The following information is displayed:

```
Usage:  PrimeRail[-tcl] [-log <filename>] [-logd <name>]
          [-cmd <filename>]
          [-cmdd <name>] [-load <scm_file>] [-replay <scm_file>]
          [-file <tcl_file>] [-iconic] [-nullDisplay] [-nogui]
          [-caseSensitive] [-version] [-release <key...>]
          [-freeAllOptionKeys] [-no_init] [-static] [-AstroRail]
          [-help]

Options/Arguments:
   -tcl                 Startup in Tcl mode
   -log filename        Name of the log file
   -logd name           Use <name>.log.mm_dd_hh_mm as log file
   -cmd filename        Name of the command log file
   -cmdd name           Use <name>.cmd.mm_dd_hh_mm as command log file
   -load scm_file       Load and run a command file (no command echo)
   -replay scm_file     Replay a command log file (after loaded)
   -file tcl_file       The Tcl script file to be executed after setup
   -iconic              Display window as an icon
   -nullDisplay         Display to a Null X Server
   -nogui               Do not show gui windows and use text-shell
                        initially
   -caseSensitive       Set process in case-sensitive mode
   -version             Display version information
   -release key...      Release license keys
   -freeAllOptionKeys   Release all optional license keys
   -no_init             Don't load user-defined Tcl initialization
                        file (optional)
   -static              Enable the static mode
   -AstroRail           Invoke PrimeRail in Astro-Rail mode
   -help                Display this information
```

The following is a description of how to use the command options and arguments:

`-tcl`

Starts PrimeRail in Tcl mode.

`-log <filename>`

Writes the log to a user-defined file. By default, PrimeRail writes the log to the file called *PrimeRail*.

`-logd <filename>`

Writes the log to a file named name.mo_dd_hh_mm (where mo_dd_hh_mm indicates the month, day, hour, and minute the file was created). By default, PrimeRail writes the log to the file called *PrimeRail.log.mo_dd_hh_mm*.

`-cmd <filename>`

Writes the commands executed in the current session to a file. If you do not specify a file name, PrimeRail writes the commands to the default file named *PrimeRail.cmd*. You can then use this command file to replay your session.

`-cmdd <name>`

Writes the command file to a file named name.cmd.mo_dd_hh_mm (where mo_dd_hh_mm indicates the month, day, hour, and minute the file was created).

`-load <scm_file>`

Loads and has PrimeRail automate the execution of the commands contained in a command file.

`-replay <scm_file>`

Loads and has PrimeRail automate the execution of the commands contained in a replay file.

By default, PrimeRail creates replay files with the name PrimeRail.cmd.mo_dd_hh_mm. However, you can specify a different name for the replay file when you start PrimeRail by using the `-cmd` option.

`-file <tcl_file>`

Loads and has PrimeRail automate the execution of the commands contained in the Tcl script file.

`-iconic`

Starts the PrimeRail application in batch mode, which means that dialog boxes and windows are displayed as icons.

`-nullDisplay`

Starts the PrimeRail application in a suppressed display mode, which means that dialog boxes and windows are not displayed.

`-nogui`

Starts PrimeRail without the GUI and uses the text shell initially.

`-caseSensitive`

Starts the PrimeRail application in case-sensitive mode.

`-version`

Displays the version of PrimeRail installed in your current path.

`-release <key...>`

Releases the PrimeRail license key.

`-freeAllOptionKeys`

Releases the optional PrimeRail license keys.

`-no_init`

Starts PrimeRail without loading Tcl initialization files (optional).

`-static`

Starts PrimeRail in static mode. When being invoked in static mode, PrimeRail allows users to run only cell-level static analysis flow inside the tool.

`-AstroRail`

Starts PrimeRail in the Astro-Rail mode. When being invoked in Astro-Rail mode, PrimeRail allows users to run Astro-Rail commands and scripts inside the tool.

For more information, see Appendix A, “Backward Compatibility,” in this user guide. If you need a detailed description about running Astro-Rail commands, see *Astro-Rail User Guide*.

`-help`

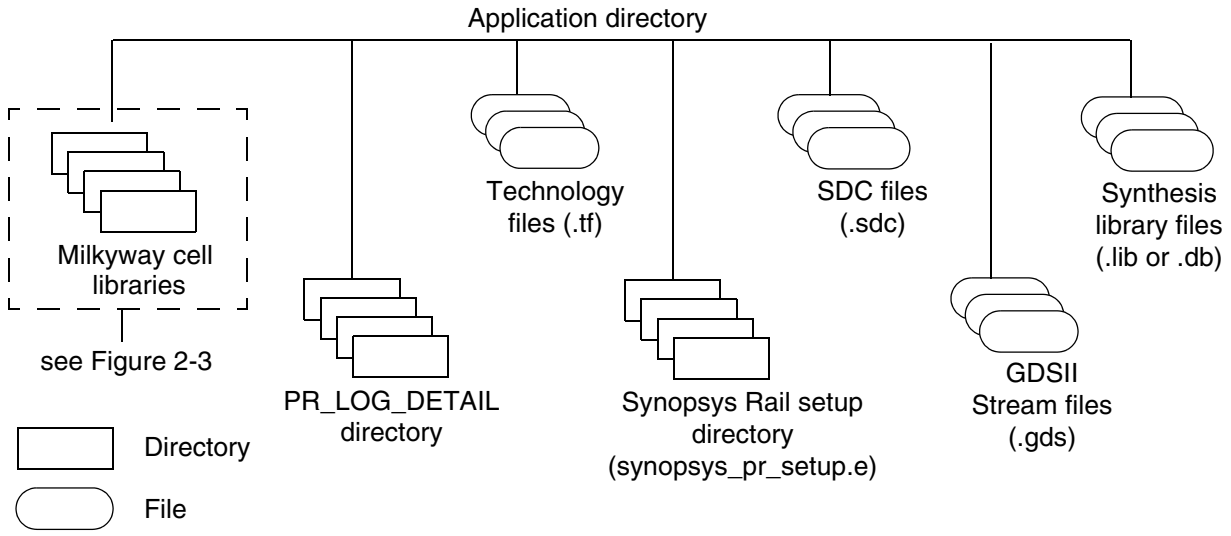
Displays the command option information.

Note:
Replay files should not contain any replay functions.

Application Directory

The application directory is where you install PrimeRail. Figure 2-2 is a graphical representation of the data structures for the PrimeRail application directory.

Figure 2-2 PrimeRail Application Directory



Milkyway Cell Libraries

A Milkyway cell library can contain top-level designs as well as the cells nested inside top-level designs. In certain cases, cells in a library can use cells located in other libraries. Instead of copying cells into multiple libraries, PrimeRail references them from the other libraries, called reference libraries.

PR_LOG_DETAIL Directory

The directory where PrimeRail saves log files, warnings, and error messages from various analysis stages.

Technology Files

The technology file defines the characteristics of a cell library, such as units of measure, graphical specifications, layer and device definitions, and design rules.

Synopsys Rail Setup Directory

The directory where you can find all the rail setup files that are generated by the IC Compiler `create_rail_setup` command. The files include the SDC file, the signal parasitic file (in the SPEF format), the Verilog netlist and the synopsys_pr_setup.e file.

Synopsys Design Constraints Files

PrimeRail supports the Synopsys Design Constraints (SDC) file to obtain clock definition data.

GDSII Files

GDSII is a standard format for transporting physical layout information between different design environments.

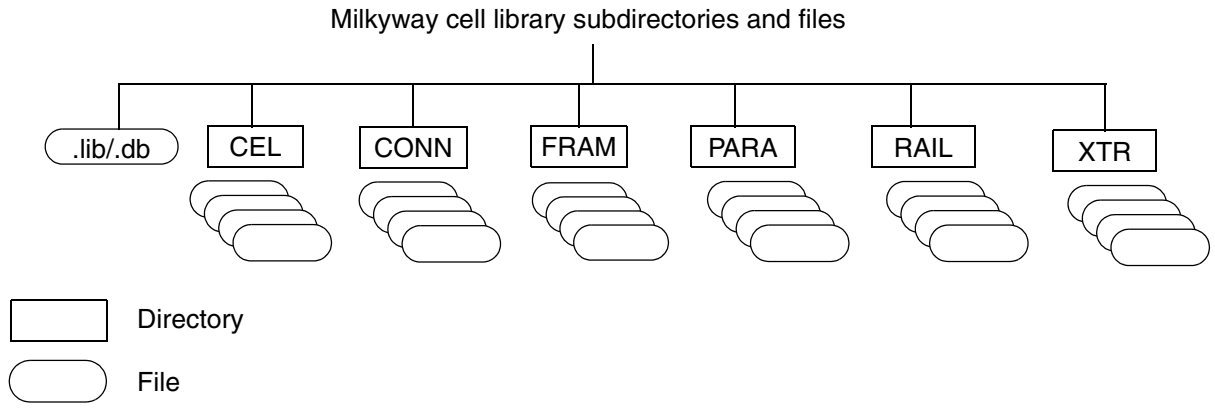
Synthesis Library File

The synthesis library file contains cell timing and functionality information in .lib or .db format.

Milkyway Cell Library Directory

Figure 2-3 shows the structure of the Milkyway cell library directory with views, which are relevant to PrimeRail. To see a complete Milkyway database structure, see the *IC Compiler Data Preparation Using Milkyway User Guide*.

Figure 2-3 Milkyway Cell Library Subdirectories and Files



The Milkyway cell library directory contains the following:

- Library information file (.lib/.db) – A binary file that contains library category and technology information. Binary files cannot be edited, but you can change the technology and CLF files by using Milkyway to unload and reload these files.

  When unloading one of these files, Milkyway places the information in a format that enables you to modify the file with a UNIX text editor.

- Cell views (subdirectories) – Views that contain cells of each type in the library. Table 2-1 lists the views shown in Figure 2-3 and their contents.

Table 2-1 Milkyway Cell Views and Their Contents

Cell view	Contents
CEL	Physical data
CONN	Internal power and ground structure connectivity information
FRAM	Placement and routing abstraction
PARA	Parasitic resistance and capacitance data
RAIL	Reliability information
XTR	Extracted layout data

Using Command Files

A command file is a file that contains commands and can be one of the following types:

- Replay file – Created each time you run PrimeRail. The file contains a list of commands executed during a PrimeRail session.
- Startup file – A special type of replay file used to set up your PrimeRail environment. You can create a startup file by running PrimeRail, then editing the automatically generated replay file.
- Load file – You can create a load file of your own. It generally contains design database functions that provide information for PrimeRail. When you replay or load a command file, PrimeRail reads the file and executes the commands and functions it contains.

Replay Files

Running PrimeRail automatically generates a command file called a replay file, in which all of the commands executed during a session are saved. You can replay a file by opening it automatically. To do this, type the following at the UNIX prompt:

```
%PrimeRail -replay fileName
```

By default, PrimeRail creates replay files with the name *appName*.cmd.mo_dd_hh_mm. To specify a different name for the replay file, use the `-cmd` option. For more information, see "Command Options" on page 2-3.

Note:
Do not include replay functions in the replay files.

Startup Files

A startup file is a special type of replay file used to set up your PrimeRail environment. You can use startup files to set the environment for PrimeRail in general or for a particular library.

You can manually create a startup file by using a UNIX text editor. You can also create a startup file by running the application and editing the automatically generated replay file. For example, you might create a startup file to automatically open the library you typically use when you start PrimeRail or to automatically open the top-level cell when you open a particular library.

Application Startup Files

When you start PrimeRail, the process looks for the .avntrc file in the following directories (in the order given):

1. The local directory (from which you issue the startup command)
2. Your home directory
3. The install directory (where PrimeRail is installed)

When it finds the .avntrc file, PrimeRail stops searching and automatically replays the file.

Library Startup Files

When you open a library, PrimeRail looks for a file named .avntrc-<*fileName*> (where <*fileName*> is the name of the library you are opening) in the local directory.

When it finds the .avntrc file, PrimeRail stops searching and automatically replays the file.

Load Files

Throughout the design flow, you can provide information to PrimeRail by using the load files. These files generally contain database functions. For example, a port information file, which contains `dbSetCellPortTypes` functions, is a load file.

To load a load file, enter the following in the command window of the PrimeRail application window:

```
load "fileName"
```

In this example, *fileName* is the name of the load file.

Using Script Files

A script file is a combination of Scheme functions and command syntax in a text file, which you can use to accomplish a series of tasks in sequence.

Pattern Matching

PrimeRail supports using patterns for matching names of multiple objects. You can build patterns from the following expressions:

- An ordinary character, which matches itself.
- A dot (.), which matches any single character.
- An asterisk (*), which matches occurrences of the character or expression immediately preceding it. For example, n* matches occurrences of the character n, and .* matches occurrences of any character.
- A backslash (\) followed by a special character, which matches the character. This allows you to match a character that would otherwise be considered a special character. For example, \. matches a dot (.) and * matches an asterisk (*).

  Note:
  In Scheme functions, you need to precede the special character with a double backslash (\\). For example, \\. matches a dot (.) and * matches an asterisk (*).
- A set of characters enclosed by square brackets ([]), which matches any character inside the brackets. To include a right square bracket (]) in the set, place it at the beginning of the set—for example, []cdrx].
- A caret (^) and a set of characters enclosed by square brackets ([]), which matches any character not included in the set. For example, [^469] matches any character except 4, 6, and 9. A caret in any other position is interpreted as an ordinary character. To include a right square bracket (]) in the set, place it at the beginning of the set, immediately following the caret—for example, [^]cdrx].
- A minus sign (–) between two characters, which indicates a range of consecutive characters and matches any character in the range. For example, [2–5] matches 2, 3, 4, or 5 and [b–d] matches b, c, or d. To include a minus sign in a set, place it at the beginning or end of the set. For example, [–cdrx] or [cdrx–].

About PrimeRail GUI

The PrimeRail GUI includes an application window and cell editing windows that provide views of a cell (a representation of a design). The windows provide menu commands, and every menu command has an underlying Scheme function.

Figure 2-4 The PrimeRail Application Window

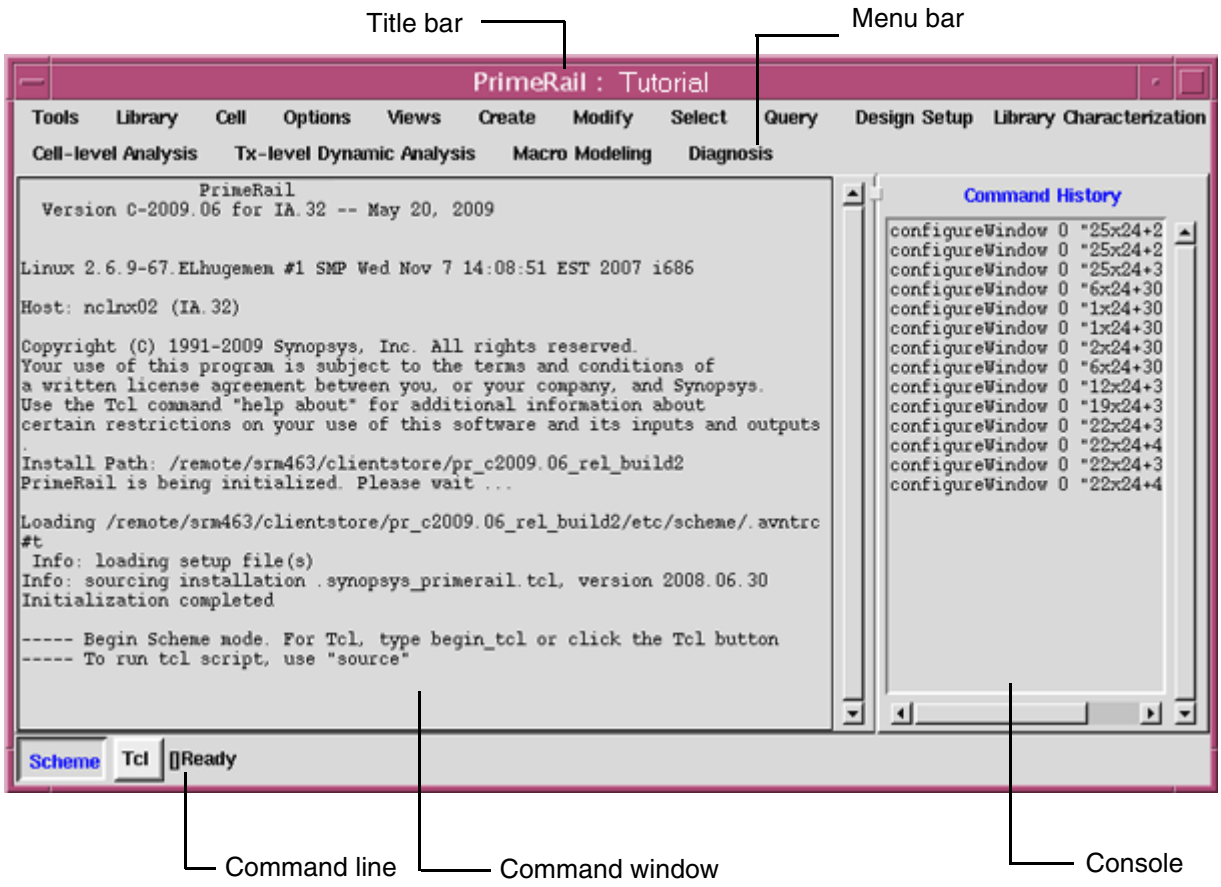


Table 2-2 explains the main areas in the PrimeRail application window.

Table 2-2 PrimeRail Application Window Description

Area	Description
Menu bar	Accesses the PrimeRail commands
Command window	Displays system messages and records all commands executed and points entered

Table 2-2 PrimeRail Application Window Description (Continued)

Area	Description
Command line	Displays the Scheme commands that correspond to the menu commands you select, and prompts you to select objects or specify points required for command execution
Console	Displays the Scheme commands you have used during the current application session

To list command functions in the command window, enter

```
functions "pattern"
```

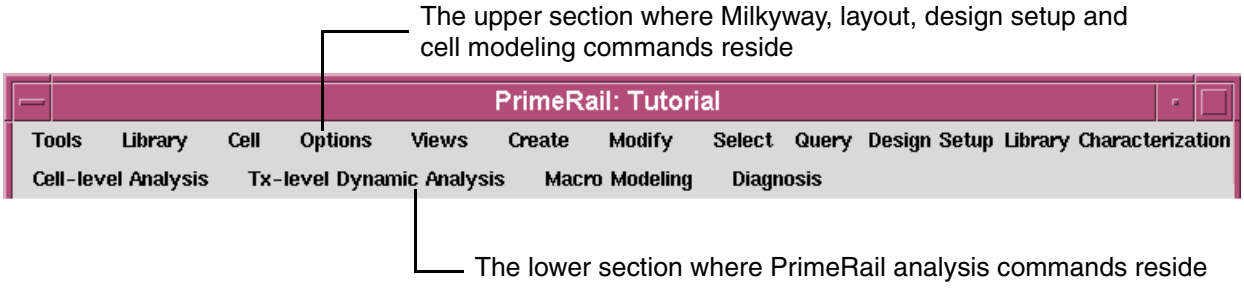
Functions with the pattern (characters specified between the quotation marks) appearing anywhere in their names will be listed. For example, to list all functions having the character string *cell* in them, enter

```
functions "cell"
```

GUI Menu Bar

The menu bar on the PrimeRail application window is divided into upper and lower sections. The upper section of the menu bar contains 11 categories in which the global, point, viewing, design setup, and library characterization commands are located.

The lower section contains 4 categories that include all the PrimeRail commands for various analysis flows.



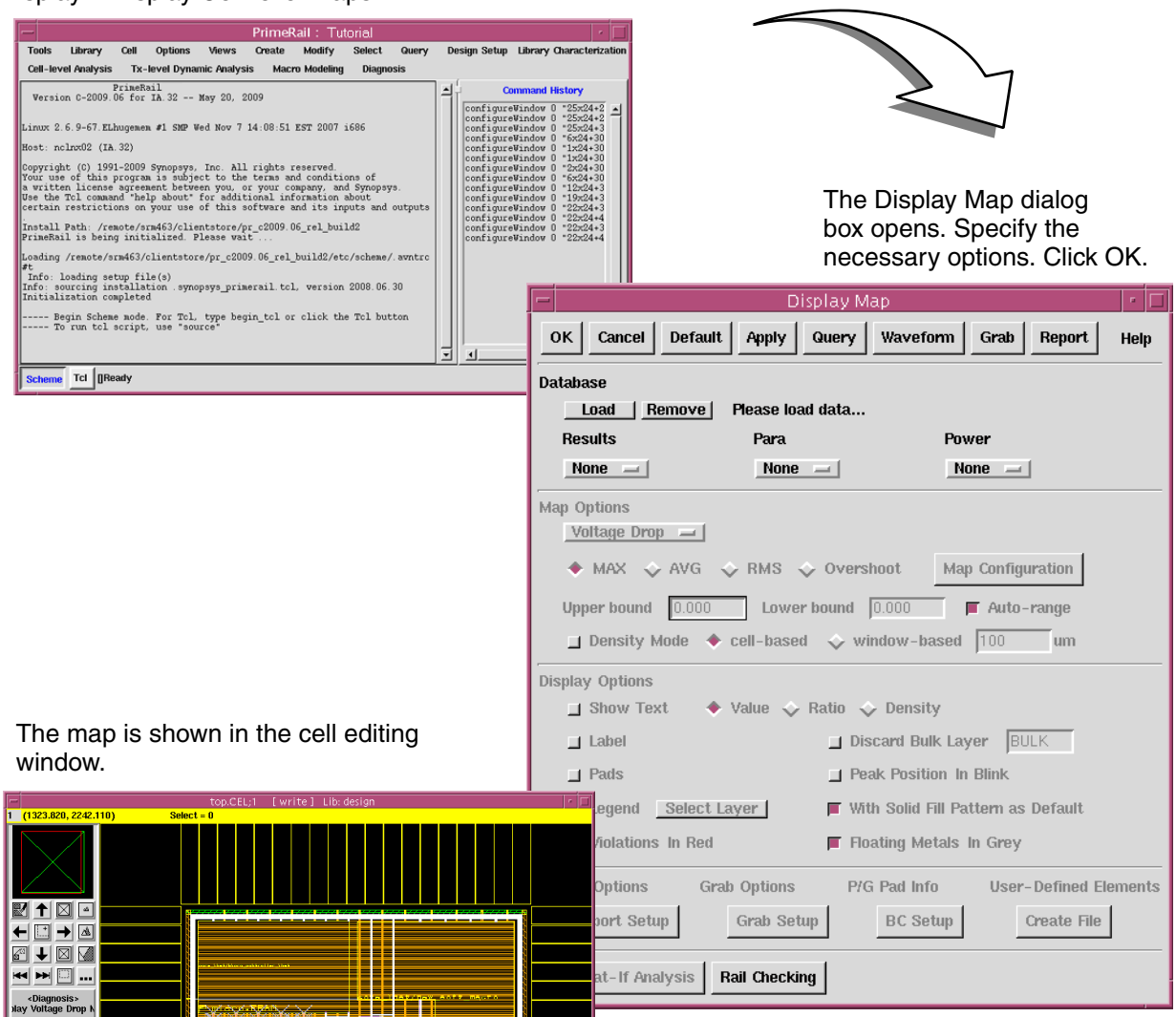
- Cell-level Analysis: Contains commands for cell-level time-average and dynamic analysis, including input data preparation and checking, power and ground net extraction, power and rail analysis, and analysis result display.
- Tx-level Dynamic Analysis: Contains commands for transistor-level dynamic analysis, including power and rail analysis and analysis result display. The command used to run transistor-level signal electromigration analysis can also be found in this category.

- Macro Modeling: Contains commands for creating macro models.
- Diagnosis: Contains commands for generating reports on power information, current sources, and dynamic current waveforms. Commands for viewing error files and setting error types are also available in this category.

Dialog Boxes and Windows

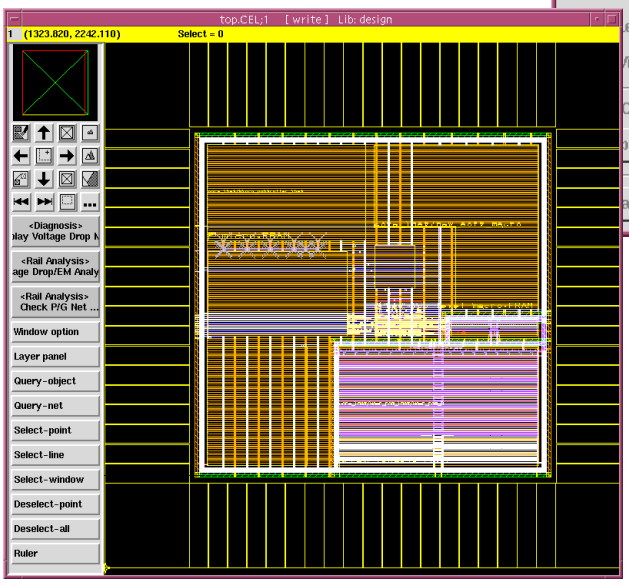
When you execute a command in PrimeRail, by either selecting it from the menu or entering it in the command window, PrimeRail executes the command immediately or displays a dialog box where you specify settings for running this command. The results are displayed in different windows, depending on the command you execute. The following figure shows a sample operation flow of running the `pgMap` command.

Enter `pgMap`, or select Cell-level Analysis > Display – Display Cell-level Maps.



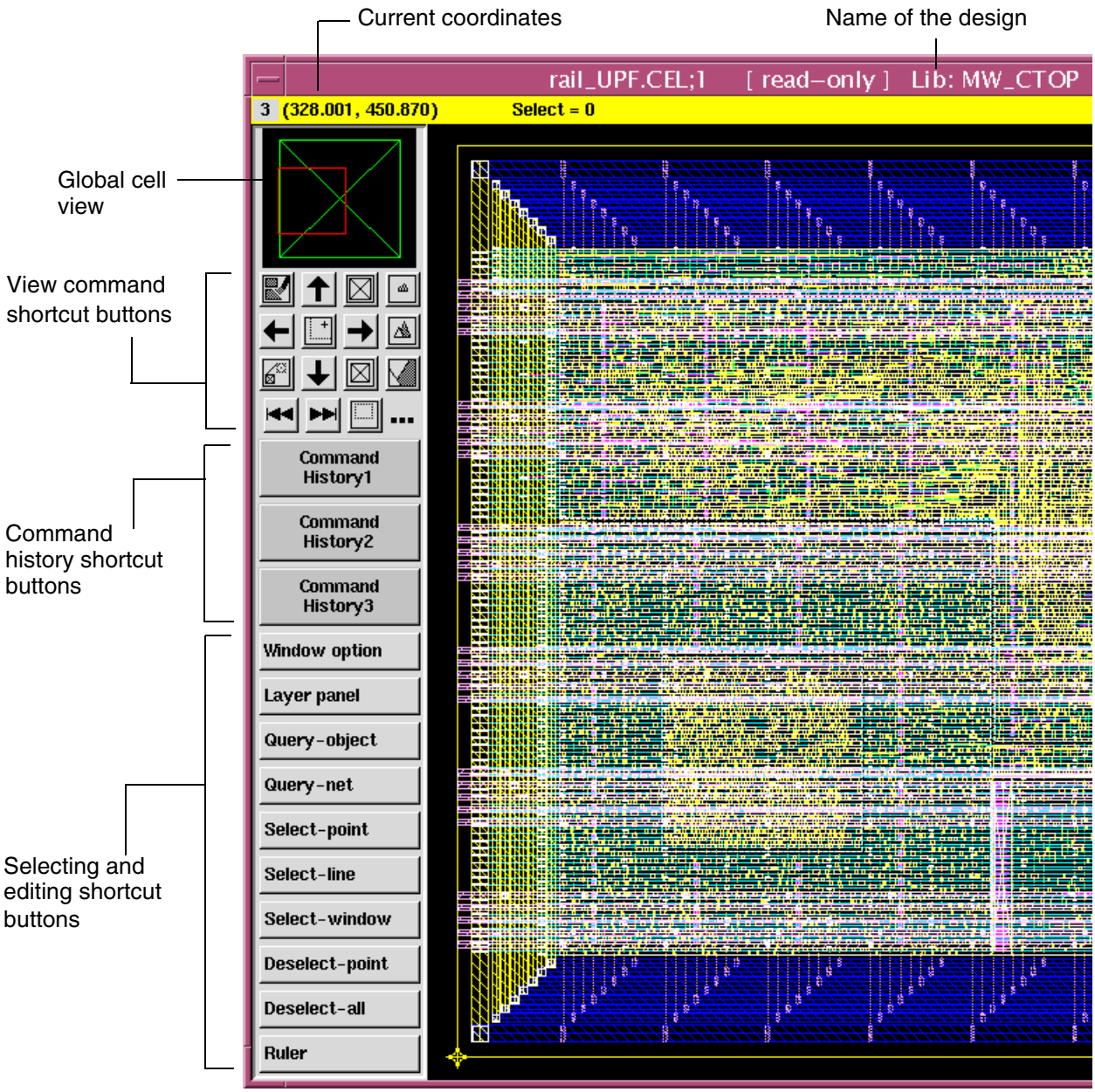
The Display Map dialog box opens. Specify the necessary options. Click OK.

The map is shown in the cell editing window.



Cell Editing Window

When you open a cell, PrimeRail creates and opens the cell editing window, where the cell is represented graphically. You can open multiple cell editing windows, each containing a different cell or the same cell.



In the cell editing window, you can view and analyze the analysis results interactively, including power distributions, voltage drop, electromigration violations, or parasitics.

For a detailed description of working with a cell, see the online Help topics on working with graphics windows. Physical Implementation Online Help describes the basics of working in the Synopsys cell layout design environment, and you receive it when you purchase Milkyway.

Getting Help in PrimeRail

The PrimeRail tool provides an online Help system with information about using the GUI and commands, as described in the following sections:

- Using Online Help
- Viewing Man Pages
- Setting Message Levels

Using Online Help

Access the PrimeRail Help system by clicking the Help button on any application menu bar or in any dialog box or by entering “help” in the command window.

The online Help system is in Hypertext Markup Language (HTML) format and can be viewed with the document browser built into the tool or any browser with HTML capability. You can view PrimeRail Help in any HTML browser, such as Internet Explorer, by opening the <PrimeRail>/help/index.html file.

You can also access online Help within the tool by clicking the Help button in any dialog box or by typing “help *command_name*” in the PrimeRail command window.

Viewing Man Pages

In PrimeRail, warnings and error messages are categorized by the occurrence at the specific stage of the tool. Each of the categories is given an index with the prefix of “RAIL-” and a number. A maximum of five warnings of each type will be printed in the PrimeRail command window or in the main log file. PrimeRail writes the complete list to a log file, named <log_file_name>.<stage_name>, in the PR_LOG_DETAIL directory.

You can check the man page on a particular rail message for explanation of that message and for recommended areas to look for a solution.

Example:

While running the `poTransientPowerAnalysis` command in PrimeRail, the following message might appear in the command window or main log file:

```
Warning: Cannot find PG port with voltage 2.5 for cellInst
clk!bufbd7!7!12 in DB. (RAIL-200)
```

There can be multiple RAIL-200 messages printed to the PrimeRail.log.02_08_11_37.transientPA file under the PR_LOG_DETAIL directory.

Now, if you want to check a man page on RAIL-200 to get more information, do the following:

- In the Scheme mode, enter

  ```
  tcl "man RAIL-200"
  ```

- In the Tcl mode, enter

  ```
  man RAIL-200
  ```

The following information is displayed in the command window:

```
NAME
RAIL-200 (warning) Cannot find PG port with voltage %lg for  cellInst %s
in DB.

DESCRIPTION
When trying to assign power values from the PrimeTime PX report file, the
tool is failing to find a PG port on the given cell instance at the
specified
voltage. The switching and short-circuit power contribution from this PG
port port instance will be ignored.

WHAT NEXT
The most common cause of this error is that you have run PrimeTime PX on
a database with a different nom_voltage than  your operating voltage for
the design. Check that the library linked in your PrimeTime PX script
is the correct one. The total power ignored will be reported at the end
of poTransientPowerAnalysis. If it is an insignificant percentage of
the total power in your design, you may ignore these warnings.
```

Setting Message Levels

The Milkyway database provides flexibility in controlling how much information a message prints to the command window. This prevents PrimeRail from printing superfluous information (such as technical information) that might take up many lines at the beginning of the command messages.

Use the `dbSetMsgLevel` command to set the message level in the following format:

```
dbSetMsgLevel "level"
```

Set *level* to high (h), medium (m), or low (l). The higher the message level is, the more information the message contains. The default level is medium.

To view what the current message level is, enter `dbGetMsgLevel` in the command window.

For details, see online Help for the commands or the “Enhanced Milkyway Message Handling” section in the *Milkyway Environment Extension Language Reference Manual*.

Interrupting or Terminating a Job

You can interrupt command processing and remain in the application. To interrupt or terminate a job, press Control-c.

The time the command takes to respond (to stop what it is doing and return to the prompt) depends on the size of the design and the function of the command you are interrupting.

Exiting PrimeRail

To exit the PrimeRail GUI, choose Tools > Quit from the menu bar. Alternatively, you can enter the following in the command window:

```
exit
```

A confirmation dialog box appears. Click OK to exit the PrimeRail application window. If any open cells have been changed, select the cells you want to save from the list of cells in the Save Cells dialog box that appears and click OK.

3

In-Design Rail Analysis in IC Compiler

In addition to running PrimeRail as a standalone tool, you can execute PrimeRail commands within IC Compiler for power network resistivity checking, and for static and dynamic voltage drop analysis and electromigration analysis. Other advanced analysis capabilities, such as decoupling capacitor insertion analysis and in-rush analysis, are also available.

This chapter includes the following sections:

In-Design Rail Analysis Flow

Reliability analysis can be done after various stages of IC Compiler, such as the placement, clock tree synthesis, global routing, detailed routing, or chip finishing stages. Catching and resolving most of the power network issues as early as possible in a design cycle is very important. In order to extend the usability and effectiveness of sign-off rail analysis, the In-Design Rail Analysis feature is implemented in IC Compiler to allow you to invoke PrimeRail in batch mode within IC Compiler if you want to complete power network checking and analysis at any stage within IC Compiler. When analysis is complete, you can check the results in IC Compiler by displaying maps or error cells. If any issues are found, you can debug the issues by looking into the analyze_rail.log file. If you need to perform detailed debugging, you can do it in the standalone version of PrimeRail. This capability of checking and fixing reliability issues without leaving the IC Compiler session greatly reduces the overall design cycle time.

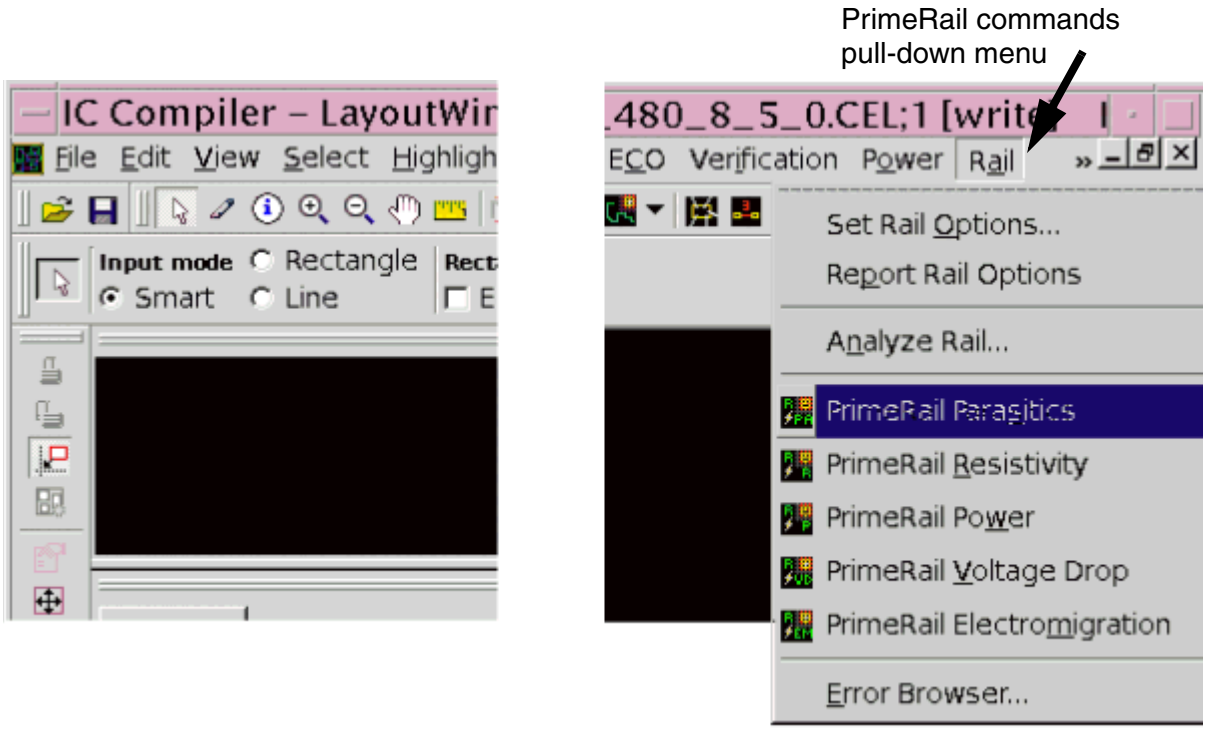
If you want to do an early rail analysis during implementation with early power estimation, such as assigned switching information and instance power, you can assign the switching and instance power information through the `set_rail_options` command. Then, run the `analyze_rail` command to consider the assigned switching and instance power information during the target rail analysis.

If you encounter any issues during IC Compiler rail analysis, you can debug the issues in the standalone version of PrimeRail. When the debugging is complete in PrimeRail, you can export the results and load them back to IC Compiler where you can check the problematic areas and fix them in the design.

Executing PrimeRail Commands in IC Compiler

You can execute PrimeRail commands on the icc_shell command line. If you invoke IC Compiler graphical user interface by executing `gui_start` in icc_shell, all the PrimeRail commands reside under the RAIL category in the IC Compiler GUI, as shown in Figure 3-1.

Figure 3-1 Executing PrimeRail Commands from IC Compiler Graphical User Interface



The following IC Compiler commands are used to complete PrimeRail tasks. For detailed descriptions about the commands and the analysis flow, see the man pages or the *IC Compiler Design Planning User Guide*.

`set_rail_options`

Specifies setup options for the `analyze_rail` command, including the option for setting analysis mode to be static or dynamic.

`report_rail_options`

Reports the current option settings for the `analyze_rail` command.

```
analyze_rail -integrity|voltage_drop|electromigration|decap|inrush|
             primerail_script_file|script_only
```

Performs the specified analysis type and checking on the specified nets. When executed, this command generates the data that is needed and runs PrimeRail in batch mode within the IC Compiler session.

This command supports several types of analyses, including power and ground network integrity analysis, voltage drop analysis, and electromigration analysis and so on. When the analysis is finished, a PrimeRail command script file will be generated. This script file is useful when errors have occurred during analysis. You can modify the generated script

file and load it back to the tool in the subsequent run. For debugging purposes, you can load the generated script file to the standalone version of PrimeRail and complete the rest of the analysis.

For more information on debugging rail analysis, using the standalone version of PrimeRail, see "Debugging In-Design Rail Analysis Flow" on page 3-7.

- `-integrity`

  Enables connectivity checking on the power and ground network of the current design, including missing vias and floating geometry. Execute this option when the design being implemented in IC Compiler has finished power network synthesis, global routing, clock tree synthesis, detailed routing or chip finishing.

- `-voltage_drop`

  Enables voltage drop analysis to make sure a voltage drop threshold is met. Execute this option when the design being implemented in IC Compiler has finished clock tree synthesis, routing, or chip finishing. Recommendations will be provided at the end of the analysis for fixing the problematic areas in the design.

- `-electromigration`

  Enables electromigration analysis on power nets to make sure the current densities are under electromigration thresholds. Execute this option when the design being implemented in IC Compiler has finished clock tree synthesis, routing, or chip finishing.

  When analysis is complete, the command reports the analysis results in the log file. If any electromigration violations are found, recommendations are provided for resolving the problems.

- `-decap`

  Enables decoupling capacitor insertion analysis on the specified net. Execute this option when you want to solve power supply noises by adding decoupling capacitance (decap) cells in the affected areas.

- `-inrush`

  Enables in-rush analysis with power management cell implementations in the IC Compiler environment.

- `-primerail_script_file`

  Generates PrimeRail script files for performing rail analysis. Execute this option when you want to run the script file in the standalone version of PrimeRail for debugging purposes.

- `-script_only`

  Performs PrimeRail analysis with the specified script file.

`read_rail_maps`

Reads analysis results to display the resistivity map, the voltage drop map and the electromigration map in IC Compiler graphical user interface.

`remove_rail_maps`

Deletes the loaded rail analysis results from database.

License Required

To invoke PrimeRail inside IC Compiler, IC Compiler searches for PrimeRail and PrimeTime executables in the UNIX search path, or the binaries that are specified in the `set_rail_options` command.

The following minimum license modules are required for invoking PrimeRail within IC Compiler:

PrimeRail, MDataPrep, Milkyway

Note that if only the cell-level static analysis is to be run, you can use the PrimeRail-static license key instead.

You do not need to have the PrimeTime PX license to invoke PrimeRail inside IC Compiler, because a limited version of PrimeTime PX is invoked within PrimeRail to generate necessary power and timing information without requiring the PrimeTime license key.

In-Design Rail Analysis Command Flows

Figure 3-2 and Figure 3-3 describe the commands used to complete block-level and chip-level rail analysis in IC Compiler.

Figure 3-2 PrimeRail Block-Level Rail Analysis Command Flow Within IC Compiler

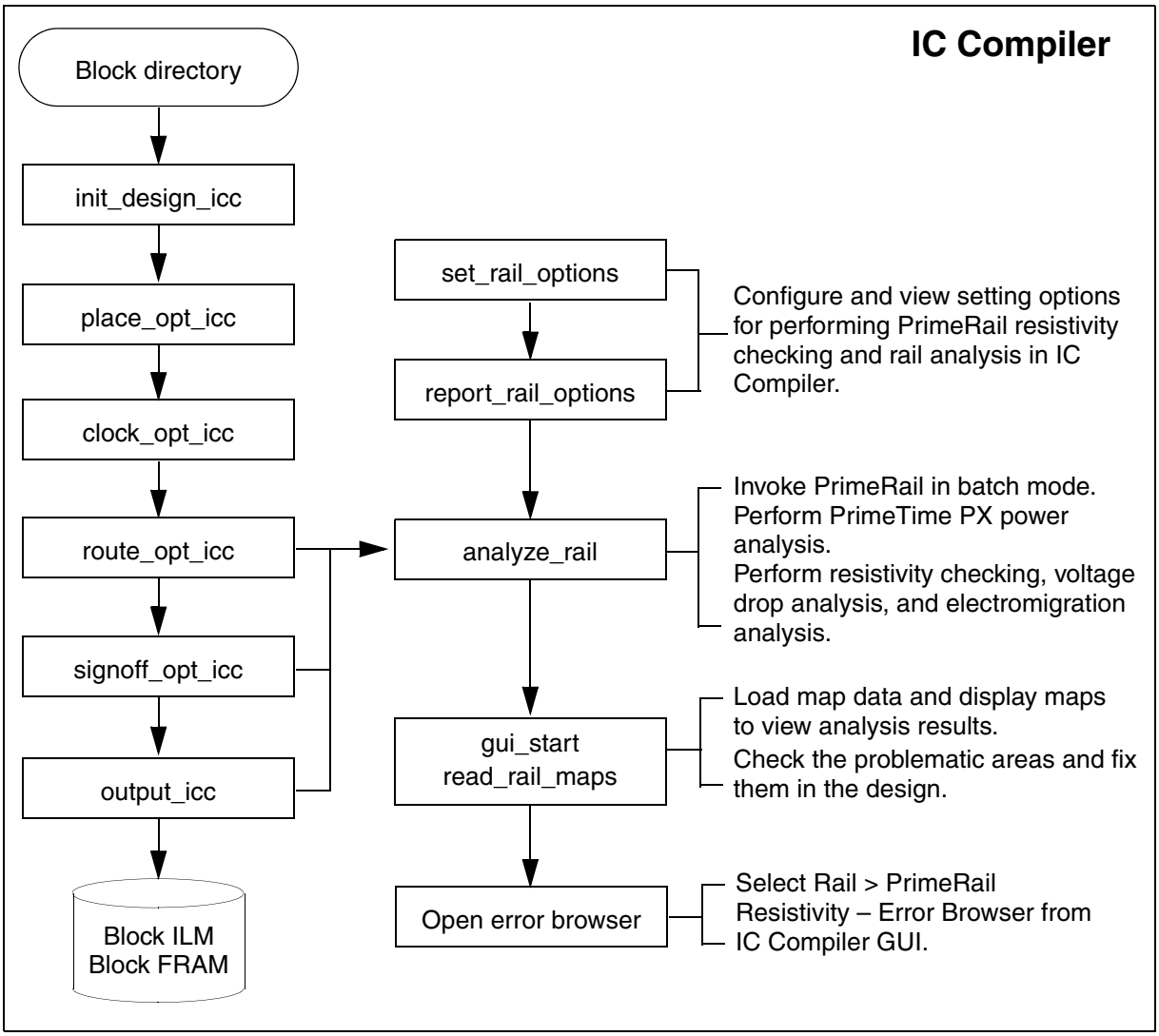
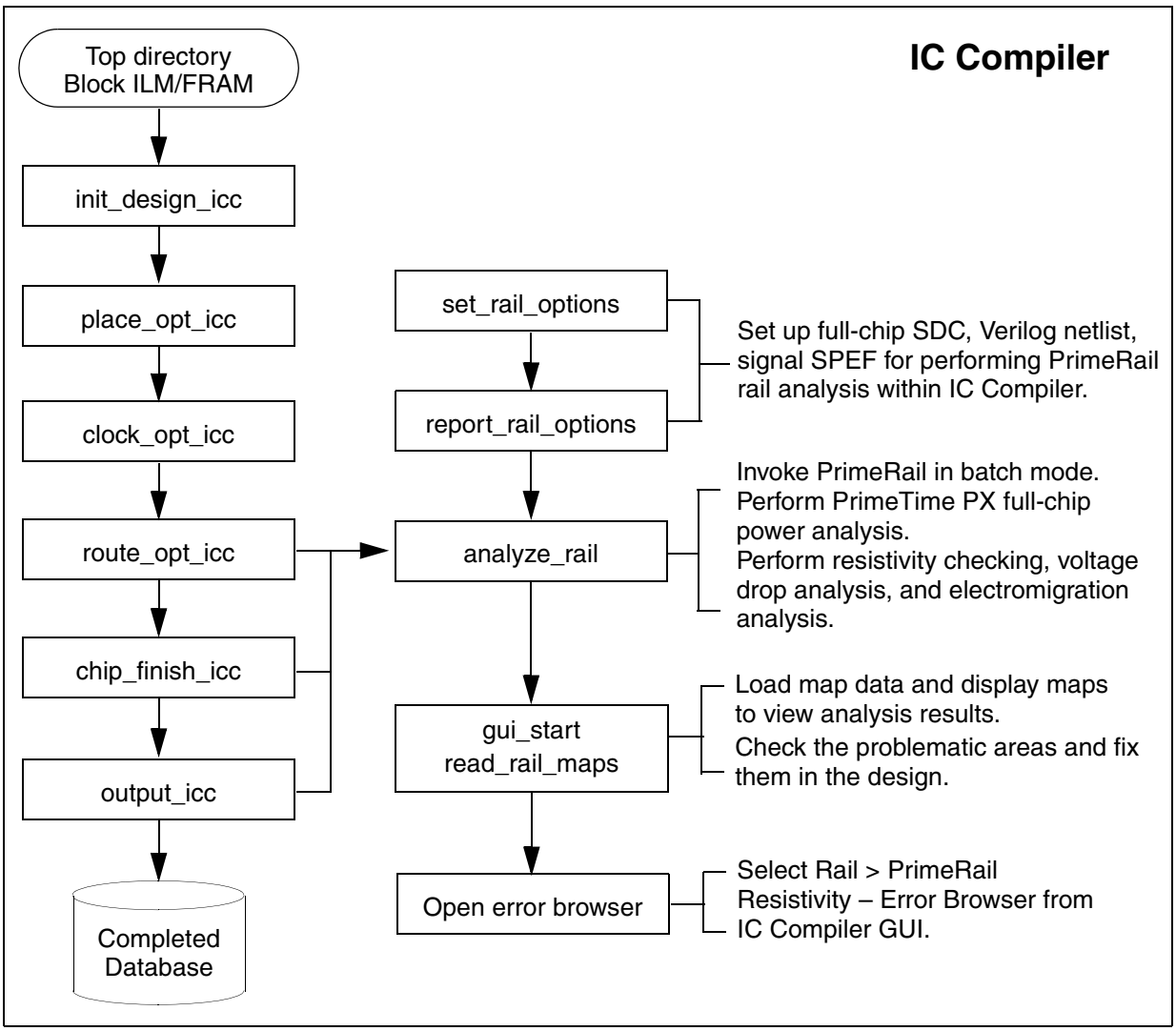


Figure 3-3 PrimeRail Chip-Level Rail Analysis Command Flow Within IC Compiler



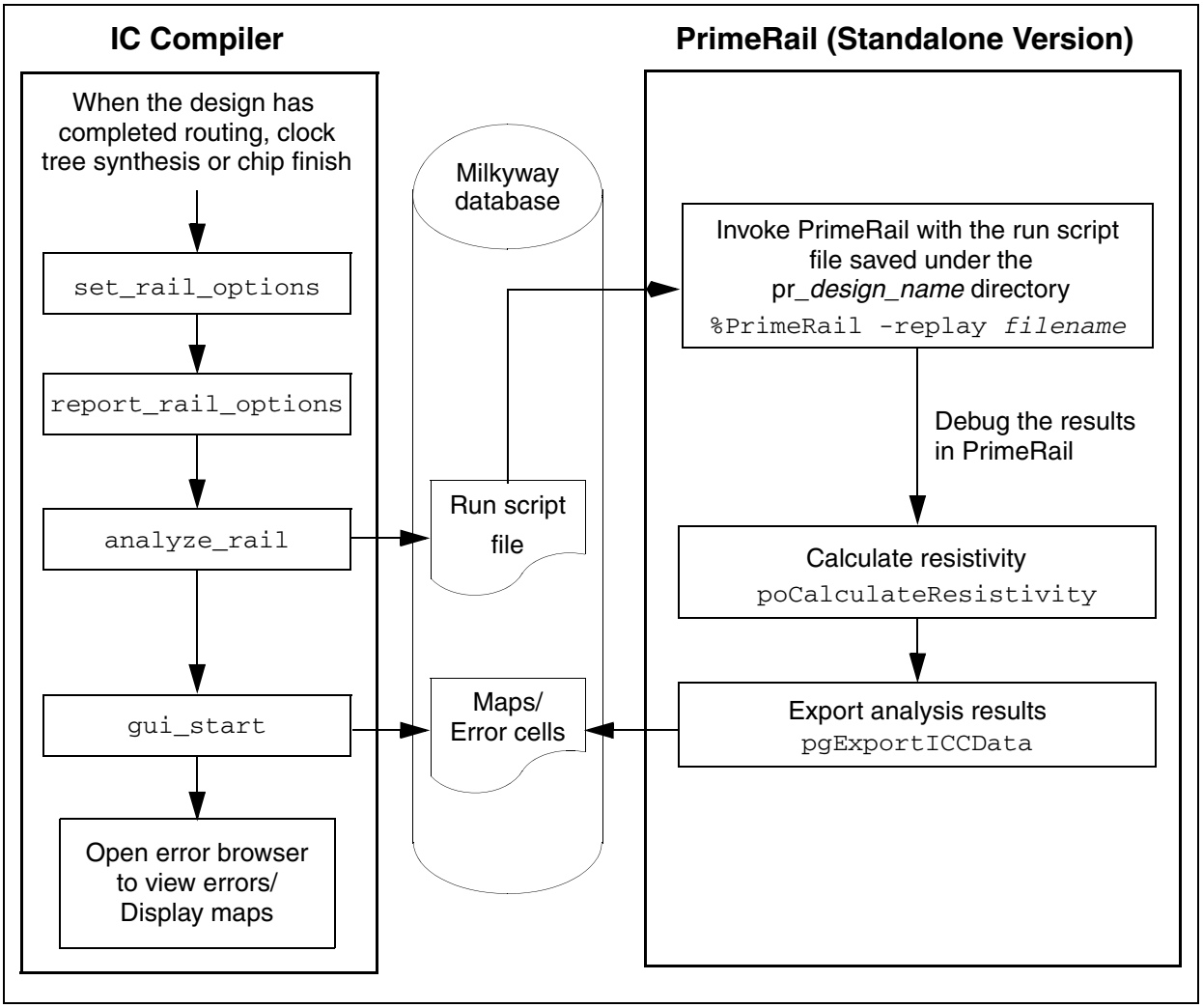
Debugging In-Design Rail Analysis Flow

When errors occur during IC Compiler rail analysis, locate the run script file from the pr_*design_name* directory where IC Compiler saves all the output and log data related to PrimeRail rail analysis. Note that the run script file and log file are rewritten each time the `analyze_rail` command is executed. If you want to keep the script and log files for debugging purposes, be sure to save a copy of the files on your disk.

Invoke the standalone version of PrimeRail with the run script file that IC Compiler generates and debug any reliability issues in PrimeRail. You can then export the analysis results to an output map file and display the results in IC Compiler for map display and error browsing.

Figure 3-4 illustrates the steps for debugging in the IC Compiler and PrimeRail analysis flow.

Figure 3-4 Debugging in the PrimeRail and IC Compiler Flow



Calculating Net Resistivity

If you complete the power and ground parasitic extraction using the `poExtractPGParasitics` command in the standalone version of PrimeRail but want to view the resistivity map in IC Compiler, execute the `poCalculateResistivity` command to calculate resistivity values from the nets in the design. Then run the `pgExportICCData` command to export the calculated resistivity results to an output file for map display in IC Compiler. The calculated resistivity results are saved to the RAIL view.

Note that if you want to view the resistivity information calculated by `poCalculateResistivity` using the `pgMap` command, choose the result name with *Resistivity* as a prefix from the Results pull-down menu in the Display map dialog box. For more information about displaying resistivity maps in the standalone version of PrimeRail, see "Displaying Resistivity Maps" on page 8-9.

Note:

You do not need to execute the `poCalculateResistivity` command if you plan to view the resistivity map using the `pgMap` command in the standalone version of PrimeRail.

Here is the syntax for calculating resistivity information:

```
poCalculateResistivity '("netName1" "netName2") "userDefineTapFileName" \
  useTopLevelDesignPin "pinResistanceFileName" \
  "padFileType=Master|Instance" "padFileNameName"
```

Argument	Description
`userDefineTapFileName`	The name of the tap file that specifies where the ideal voltage is applied by coordinates and layer numbers. For details on using tap files, see "User-Defined Taps" on page 7-25.
`useTopLevelDesignPin`	Specify to use the top-level design pin. The default setting is 1. Otherwise, set the argument to 0.
`pinResistanceFileName`	The name of the pin resistance file.
`padFileType`	Specify the pad file type to be Master or Instance.
`padFileName`	The name of the pad file.

You need to define at least one argument so that PrimeRail can locate boundary conditions for resistivity calculation. Other STRING type arguments can be empty.

Example

If you want to calculate resistivity from the nets VDD, VDDV, and VDDX, and the boundary condition is to use top-level design pins, execute the `poCalculateResistivity` command, using the following syntax:

```
poCalculateResistivity '("VDD" "VDDV" "VDDX") "" 1 "" "" "" ""
```

When your calculation is complete, the tool writes the results in the log, as shown in the following example:

```
;; poCalculateResistivity

---------- Arguments -------------
Net names: [0] VDD
Use top level design pin: TRUE

Loading PG Net VDD ...
Load Parasitic Data  Memory: 268.543 MB  CPU: 0 seconds  Elapse: 0
second

testcell VDD

Reading user defined pad location under VDD ...
Calculating Resistivity Map ...
Calculate resistivity value
  Memory: 268.543 MB
Resistivity Result File: Default:Default:VDD, size: 2306940 Bytes
poCalculateResistivity  Memory: 268.543 MB  CPU: 0 seconds  Elapse: 1

second
```

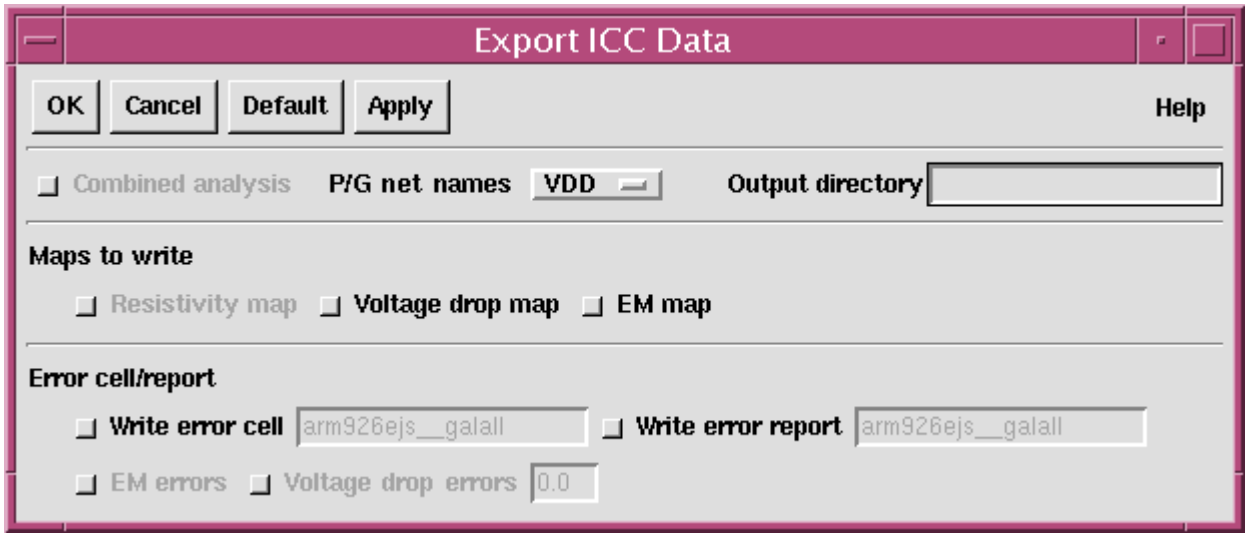
Exporting PrimeRail Analysis Data for IC Compiler

If you finish the rail analysis tasks in the standalone version of PrimeRail and want to view the results in IC Compiler, execute the `pgExportICCData` command to write the calculated analysis data that is saved in RAIL view to an output file for map display in IC Compiler. Error reports and error cells generated from voltage drop and electromigration analyses will also be generated if needed.

Note that if you want to display the resistivity map in IC Compiler, you need to run the `poCalculateResistivity` command before `pgExportICCData`. If you are to display resistivity map with `pgMap` in the standalone version of PrimeRail, running the `pgExportICCData` command is not necessary.

To export analysis data for IC Compiler,

1. Enter `pgExportICCData`. The Export ICC Data dialog box appears.



2. Select “Combined analysis” if you are performing a full-chip analysis.
3. Select the name of the power or ground net whose RAIL view data is to be generated.
4. In the “Output directory” text box, enter the name of the directory where you want to save the output data and error files that are generated.
5. In the “Maps to write” section, choose the type of map data that is to be generated.
6. In the “Error cell/report” section, select “Write error cell” and specify the name to be assigned to the error cell. The default naming convention for error cells is *cell_name_net_name*_export.err. You can open and view the generated error cells in the IC Compiler error browser.

   To generate an error report file, select “Write error report” and specify the file name you want to assign to the error report to be generated. The default name convention for error reports is *cell_name_net_name*_vdlem.txt.

   Select “EM errors” or “Voltage drop errors” to generate errors from electromigration analysis or voltage drop analysis.
7. Click OK.

Sample Log

The following is an example of log messages written by the `pgExportICCData` command:

```
;# pgExportICCData
Loading PG Net VDD ...
Load Parasitic Data  Memory: 268.543 MB  CPU: 0 seconds  Elapse: 0
seconds
```

```
testcell VDD
EM Rule Table:

Scaling Factor: 1.000000

Layer: metal1  Mode: average
  Width      | Max. Current
  -----------+-------------
  1.600e-01  | 6.287e-02
  1.975e-01  | 8.648e-02
  2.655e-01  | 1.292e-01
  4.292e-01  | 2.322e-01
  3.665e+00  | 1.802e+00
  2.787e+01  | 1.245e+01

Building data structure for net VDD  Memory: 268.543 MB  CPU: 1 seconds
Elapse: 1 seconds

Writing map file for net VDD  Memory: 268.543 MB  CPU: 2 seconds  Elapse:
3 seconds

Cell arm926ejs__galall_VDD_export.err existed already. Delete it ...
Report Total EM violations    : 0
Generating EM violations report...
  Memory: 268.543 MB  CPU: 0 seconds  Elapse: 0 seconds
EM report successfully generated
pgExportICCData  Memory: 268.543 MB  CPU: 4 seconds  Elapse: 4 seconds
```

4

Preparing Library and Other Input Data

To analyze voltage drop and current density violations in PrimeRail, the tool requires a certain set of data to be available in the Milkyway technology file, such as the units and precision values of the voltage, current, and power, and the setting of the maximum current density that can flow through each metal and via layer. If you have a design database in the Library Exchange Format/Design Exchange Format (LEF/DEF) format with GDSII data for hard macros or pad cells, you have to import it into the Milkyway database.

PrimeRail also supports the wire and via electromigration rules through an Advanced Library Format (ALF) file. Usually the advanced library file can be obtained from the foundry. If you need an application note on preparing ALF files based on electromigration rules provided by the foundry, contact your local Synopsys application consultant. In addition to these library input data, PrimeRail requires the rail setup information from IC Compiler to be available before performing power and ground net extraction, power analysis and rail analysis. See “Creating Rail Setup Data in IC Compiler” on page 5-3 for more information.

This chapter includes the following sections:

Setting Up the Technology File

To analyze voltage drop and current density violations in PrimeRail, the tool requires a certain set of data to be available in the Milkyway technology file. The Milkyway library technology file defines the characteristics of a cell library, such as units, layers, design rules, and capacitance models. Each technology file contains several sections. You need to modify the technology section and the layer section of the technology file for PrimeRail:

- Technology section
- Layer section

The following is a description of how to set up a technology file for PrimeRail purposes. If you need information about how to create and load a technology file, see the related chapters in the *IC Compiler Data Preparation Using Milkyway User Guide*.

Technology Section

In the technology section of your technology file, define the following power-related units:

- The voltage unit and precision, using `unitVoltageName` and `voltagePrecision`
- The current unit and precision, using `unitCurrentName` and `currentPrecision`
- The power unit and precision, using `unitPowerName` and `powerPrecision`

These fields are in bold in the following example:

Technology Power Settings Example

```
Technology      {
      name                      =    CMOS18FV"
      dielectric                =    3.942e-05
      unitTimeName              =    "ns"
      timePrecision             =    1000
      unitLengthName            =    "micron"
      lengthPrecision           =    1000
      gridResolution            =    20
      unitVoltageName           =    "v"
      voltagePrecision          =    1000
      unitCurrentName           =    "mA"
      currentPrecision          =    10000
      unitPowerName             =    "mW"
      powerPrecision            =    10000
      unitResistanceName        =    "ohm"
      resistancePrecision       =    10000
      unitCapacitanceName       =    "pf"
      capacitancePrecision      =    1000000
      unitInductanceName        =    "nh"
      inductancePrecision       =    1000
}
```

The valid units of voltage, current, and power are volt (V), ampere (A), and watt (W), respectively, with or without any of the following prefixes:

```
f = femto
p = pico
n = nano
u = micro
m = milli
```

Layer Section

In each layer section for metals and vias in your technology file, define the threshold current density that the layer can safely sustain with the maxCurrDensity statement. This information is process-dependent. Consult with your process engineer or documentation to determine the appropriate values.

Specify the threshold current density for each metal and via layer in the technology file if you plan to perform error checking during electromigration diagnosis. When segments of metal rails or via objects are found with a current density higher than the specified value, PrimeRail flags these as errors and reports them to an error cell or to an ASCII log file.

Table 4-1 shows the units and default values of `maxCurrDensity`.

Table 4-1 Units and Default Values of maxCurrDensity

Layer type	Units	Default value
Conducting metal layer	amp/cm	1e+23 amp/cm
Via layer	amp/cm^2	1e+23 amp/cm^2

The following example shows a layer section of the technology file with the `maxCurrDensity` setting in bold:

Layer Power Settings Example

```
Layer "M1"       {
      layerNumber                    =   3
      maskName                       =   "metal1"
      isDefaultLayer                 =   1
      defaultWidth                   =   0.6
      minWidth                       =   0.6
      minSpacing                     =   0.6
      fatThinMinSpacing              =   3
      fatFatMinSpacing               =   6
      pitch                          =   1.4
      fatWireThreshold               =   30
      maxSegLenForRC                 =   1000
      blink                          =   0
      visible                        =   1
      selectable                     =   1
      lineStyle                      =   "solid"
      pattern                        =   "slash"
      color                          =   "cyan"
      fillPattern                    =   "outlineStippleFill"
      unitMinHeightFromSub           =   0.8
      unitNomHeightFromSub           =   0.8
      unitMaxHeightFromSub           =   0.8
      unitMinThickness               =   0.7
      unitNomThickness               =   0.7
      unitMaxThickness               =   0.7
      maxDeltaWidth                  =   0.7
      nomDeltaWidth                  =   0.7
      minDeltaWidth                  =   0.7
      maxCurrDensity                 =   10.00
}
```

Note:

The maximum current density that is specified (indicated in bold in the example) is multiplied by the defined current precision before it is stored in the Milkyway database. If the scaled value is too large because of a very high current `maxCurrDensity` value or a very high `currentPrecision` setting, overflow occurs. This occurs when the combined value exceeds the value of 2^{31} -1.

To solve this problem, reduce the `currentPrecision` setting of the technology section until no overflow occurs.

ContactCode Section

In the ContactCode section, define the upper and lower layers for poly-contact, contact and via layers. The poly-contact layer is the layer between poly and metal. The contact layer is the layer between diffusion and metal. The via layer is between metal_N and metal_N+1.

ContactCode Section Example

```
ContactCode     "contactcode_polycont" {
                contactCodeNumber               = 1
                cutLayer                        = "POLYCONT"
                lowerLayer                      = "POLY1"
                upperLayer                      = "METAL1"
                isDefaultContact                = 1
                cutWidth                        = 0.16
                cutHeight                       = 0.16
                upperLayerEncWidth              = 0
                upperLayerEncHeight             = 0
                lowerLayerEncWidth              = 0.07
                lowerLayerEncHeight             = 0.07
                minCutSpacing                   = 0.18
}
ContactCode     "contactcode_contact" {
                contactCodeNumber               = 2
                cutLayer                        = "CONT"
                lowerLayer                      = "N-Diff"
                upperLayer                      = "METAL1"
                isDefaultContact                = 1
                cutWidth                        = 0.16
                cutHeight                       = 0.16
                upperLayerEncWidth              = 0
                upperLayerEncHeight             = 0
                lowerLayerEncWidth              = 0.07
                lowerLayerEncHeight             = 0.07
                minCutSpacing                   = 0.18
}

ContactCode     "via1" {
                contactCodeNumber               = 3
                cutLayer                        = "VIA12"
```

```
                    lowerLayer                      = "METAL1"
                    upperLayer                      = "METAL2"
                    isDefaultContact                = 1
                    cutWidth                        = 0.19
                    cutHeight                       = 0.19
                    upperLayerEncWidth              = 0.005
                    upperLayerEncHeight             = 0.05
                    lowerLayerEncWidth              = 0.01
                    lowerLayerEncHeight             = 0.05
                    minCutSpacing                   = 0.29
                    unitMinResistance               = 0.00102
                    unitNomResistance               = 0.00102
                    unitMaxResistance               = 0.00102
}
```

LEF/DEF Database

Use the `read_lef` or `read_def` command to create the Milkyway reference library by translating the LEF or DEF files. After the database conversion, it might be necessary to create a technology file and a .db file to ensure that the database includes complete information.

It is recommended that you read in first the Verilog netlists and then the DEF files if you are using the LEF/DEF flow. Otherwise the tool will report errors about missing connectivity information.

This happens because the `read_def` command tries to retrieve connectivity information from Verilog netlists when you are reading in the DEF files. If you do not load the Verilog netlists before reading in the DEF files, set the `PCXGDEF` switch before you run the `read_def` command by entering the following in the command window:

```
define PCXGDEF 0
```

By default, the switch is set to 1.

You do not need to set the switch if you are using version W-2004.12 (and earlier) of Milkyway. The `PCXGDEF` switch has been implemented to prevent flow issues resulting from the optimization of DEF In.

The following are sample DEF In messages that appear when you run different versions of the tool:

- When you run version X-2005.09 and later, this message might appear:

```
Add the new net U_CORE\/bsr_i1/bsc_v_data_i_9\latch_out_2 in
Incremental
netlist input mode.
Add the new net U_CORE\/bsr_i1/si_bsc_v_data_60 in Incremental netlist
```

```
input mode.
```

- When you run version W-2004.12 of Milkyway and earlier, this message might appear:

```
Add the new net U_CORE\/bsr_i1/bsc_v_data_i_9\ / latch_out_2
 in Incremental netlist input mode.
Connect pin instance A in U_CORE\/bsr_i1/BH3_BUF295 to net U_CORE\
 /bsr_i1/bsc_v_data_i_9\/latch_out_2.
Connect pin instance Z in U_CORE\/bsr_i1/BH3_BUF296 to net U_CORE\
 /bsr_i1/bsc_v_data_i_9\/latch_out_2.
```

For a detailed description about importing LEF and DEF files, see the chapter of translating physical design data in the *IC Compiler Data Preparation Using Milkyway User Guide*.

CCS Power Model Support

PrimeRail supports CCS power modeling technology and extends current library models to include current-based waveform data, which provides a complete solution to address time-average power, dynamic power, and IR drop violations. When CCS power .db file is present, PrimeRail automatically reads the library data directly from the CCS power .db file for power and rail analysis. Unlike when you use NLPM libraries, you do not need to perform library characterization if the .db file has already been created with the CCS power library.

PrimeRail supports the full set of information contained in the CCS power library, including

- Intrinsic parasitics for steady state cells
- Current waveforms for modeled switching events
- Gate leakage currents for cells at the steady state
- Current waveforms for macros, such as memory cells
- I/V curves for power management cells
- Leakage currents and parasitics for decoupling capacitor cells

Because the CCS power library stores leakage currents and dynamic current waveforms, you do not need to characterize the library before running cell-level dynamic power and rail analysis; the tool reads data directly from the CCS power library. The steps involved in creating dynamic current waveforms for each cell instance are as follows:

1. Perform gate-level power analysis with PrimeTime PX
2. Perform cell-level dynamic power analysis to generate dynamic current waveforms

If your .db file is generated with the NLPM library, generating dynamic current waveforms for cell-level blocks is more complicated. To allow a viable simulation time for analyzing designs that contain millions of cell instances, generating dynamic current waveforms for each cell instance consists of the these three steps:

1. Perform library characterization
2. Perform gate-level power analysis with PrimeTime PX
3. Perform cell-level dynamic power analysis to generate dynamic current waveforms

Unified Power Format Support

The Unified Power Format (UPF) is an industrial standard for specifying power design intent as an extension to logic specification. UPF provides a consistent format throughout the entire design flow and will be part of the design source. It can also add power-aware functionality to the design, such as level shifters, isolation cells, and retention registers. As part of the Synopsys low power flow, PrimeRail supports UPF Accellera Standard v1.0.

In the Synopsys low power flow, PrimeTime PX supports a subset of UPF commands for both timing and power analysis. The PrimeRail `poCalculatePower` command uses the PrimeTime PX power reports to calculate dynamic current waveforms. No additional step is required in analyzing a UPF design in PrimeRail.

Here are the major concepts defined in UPF:

- Power domains – Logic grouping of one or more hierarchical blocks in a design that share the same power domains
- Supply ports – The ports that provide the supply interface to power domains and switches
- Supply nets – The nets that connect supply ports
- Power switches – The switches that control the supply distribution

Using Electromigration Rules

PrimeRail uses the width- and temperature-dependent current limits from the Advanced Library Format (ALF) file and extracts all other technology and library information (such as constraint information) from the Milkyway technology file. The constraint information is normally loaded into the database together with the technology file during data preparation.

In the cell-level time-average analysis flow, if you want to perform advanced width-dependent electromigration checking for electromigration thresholds, PrimeRail supports defining electromigration thresholds through a Physical Library Database (.pdb)

file or an ALF file. The PDB file contains layer, via, site and macro cell definitions. ALF is a modeling language for library elements used in IC technology. ALF serves as the data specification of library elements for design applications used to implement integrated circuits. The range of abstraction is from the register transfer level (RTL) to the physical implementation level.

In the cell-level dynamic reliability analysis and transistor-level signal net electromigration analysis, PrimeRail supports defining electromigration thresholds through an ALF file for advanced width-dependent electromigration checking for maximum DC current density.

In the ALF file, you need to specify peak, average, and RMS current limits for all the design layers you want to check the electromigration violations. The current limit in the LAYER section can be temperature dependent, width dependent for the routing layer, and area dependent for the cut layer.

Caution:

PrimeRail analyzes voltage drop and electromigration violations only on power and ground nets. Currently, PrimeRail does not support gate-level signal EM analysis, but IC Compiler does. See the *IC Compiler Implementation User Guide* for running gate-level signal EM analysis with IC Compiler.

This section includes the following topics:

Importing a PDB File

Use the `cmPDBIn` command to import a PDB (.pdb) file into a Milkyway library, creating the technology information and cells in the library.

For a detailed description of importing a PDB file, see the online Help of the command.

Loading an ALF File

In an ALF file, the information about current limitations is described in the LAYER section, including average measurement for DC (power route), absolute average measurement for AC (signal route), and peak and RMS measurement.

When the database contains the ALF file, complete the steps as usual for running power analysis, power and ground net extraction, and rail analysis.

Important:

Before loading an ALF file to the database, make sure the current density units defined for each metal and via layer in the file are consistent with those in the database.

All the electromigration thresholds defined in the ALF file should be expressed in terms of current, not current density.

In PrimeRail, loading an ALF file for cell-level analysis and transistor-level signal net electromigration analysis requires different commands.

- Power and Ground Net Electromigration Analysis
- Transistor-Level Signal Net Electromigration Analysis

Power and Ground Net Electromigration Analysis

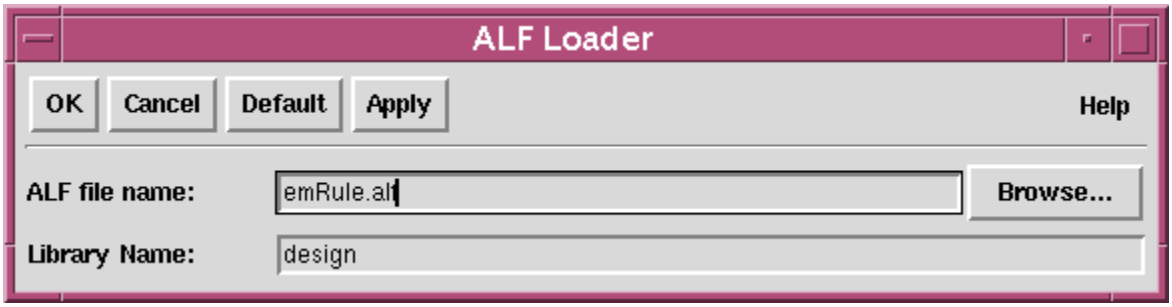
Use the `auAlfToDB` command to load an ALF file to the database if you are conducting a power and ground net electromigration analysis. The command parses the ALF file and reports if any problem occurs during the process in the log file. If you want to verify the loaded results, use the `auDumpALF` command to save the loaded ALF data to an output file (see "Reporting ALF Information" on page 4-15). If you want to remove the loaded ALF data, use the `auPurgeSIOfALF` command (see "Deleting ALF Information" on page 4-15).

To load an ALF file for electromigration thresholds,

1. Enter `auAlfToDB` in the command window.

   The ALF Loader dialog box appears, as shown in Figure 4-1.

Figure 4-1 The ALF Loader Dialog Box



2. Enter the name of the ALF file to be loaded or click Browse to specify one from the list that appears.
3. Enter the name of the library where you want the ALF file loaded.
4. Click OK or Apply.

The tool will report the use of ALF file in the log message when you display analysis results with the `pgMap` command. The following is an example of the log message:

```
EM Rule Table:
Scaling Factor: 1.000000

Layer: metal1      Mode: static
             | 1.050e+02 | 1.100e+02 | 1.250e+02 | Temperature
  -----------+-----------+-----------+-----------+-----------
  1.000e-01  | 2.290e-01 | 1.600e-01 | 5.700e-02 |
  1.000e+00  | 2.811e+00 | 1.960e+00 | 7.020e-01 |
  1.000e+01  | 2.862e+01 | 1.996e+01 | 7.146e+00 |
  -----------+-----------+-----------+-----------+-----------
  Width      |                                   | Max. Current

Layer: metal2      Mode: static
             | 1.050e+02 | 1.100e+02 | 1.250e+02 | Temperature
  -----------+-----------+-----------+-----------+-----------
  1.000e-01  | 2.290e-01 | 1.600e-01 | 5.700e-02 |
  1.000e+00  | 2.811e+00 | 1.960e+00 | 7.020e-01 |
  1.000e+01  | 2.862e+01 | 1.996e+01 | 7.146e+00 |
  -----------+-----------+-----------+-----------+-----------
  Width      |                                   | Max. Current
```

ALF File Example

The following is an example of a typical ALF file, including the LIBRARY and LAYER sections.

```
LIBRARY design {
      TIME { UNIT = 1e-09; }
      FREQUENCY { UNIT = 1e+09; }
      SLEWRATE { UNIT = 1e-09; }
      DELAY { UNIT = 1e-09; }
      HOLD { UNIT = 1e-09; }
      PULSEWIDTH { UNIT = 1e-09; }
      RECOVERY { UNIT = 1e-09; }
      REMOVAL { UNIT = 1e-09; }
      SETUP { UNIT = 1e-09; }
      SKEW { UNIT = 1e-09; }
      CAPACITANCE { UNIT = 1e-12; }
      CURRENT { UNIT = 1; }
      ENERGY { UNIT = 1e-12; }
      WIDTH { UNIT = 1e-06; }
      THICKNESS { UNIT = 1e-06; }
      AREA { UNIT = 1e-12; }
}
LAYER METAL5 {
      PURPOSE = routing;
      LIMIT {
            CURRENT dc_limit {
            UNIT = mAmp ;
            MEASUREMENT = static ;
            HEADER {
                  WIDTH {TABLE { 0.5 1.0 10.0 17.5 25.0 }
                  }
```

```
            }
            TABLE { 300 400 600 800 1000 }
            }
      }
}
```

The current limit in the LAYER section can be temperature-dependent, width-dependent for the routing layer, and area-dependent for the cut layer.

Define the units for CURRENT, WIDTH, AREA in the LIBRARAY section. These units, however, are overwritten under each layer when thresholds are specified. When finished, load the ALF file to the design library with the `auAlfToDB` command. Note that you cannot change any rules during the current PrimeRail session after the file is loaded.

When you open the design after the file is loaded, PrimeRail acknowledges the existence of the ALF file by writing the following in the log file:

```
Found [ALF file] on metal (width vs current) in DB units
      1e3    238.6e3
```

The DB units refer to the units and precision defined in the Milkyway technology file. In the example above, 238.6e3 in DB units means that the number stored in the database is 238,600. If the precision in the technology file is 1e5, then the actual value is 2.386 in mA (because this is the unit of current).

Transistor-Level Signal Net Electromigration Analysis

When performing a transistor-level signal net electromigration analysis, you must load an ALF file using the `poSigDynamicAnalysis` command. Specify the name of the ALF file to be used in the Signal Nets Dynamic Analysis dialog box (see Figure 13-2 on page 13-5). For more information on analyzing signal nets at the transistor level, see Chapter 13, “Performing Transistor-Level Signal Net Electromigration Analysis,” in this user guide.

The tool will report the use of the ALF file in the log message when you display analysis results with the `poSigDisplayElectromigrationMap` command.

The following is a sample ALF file to be used in transistor-level signal net electromigration analysis:

```
LIBRARY mpu{

LAYER metal1{
        PURPOSE = routing;
        LIMIT{
            CURRENT peak_limit{
                MEASUREMENT = peak;
                MAX{ HEADER{
                        WIDTH{ TABLE{0.1 1.0 10}}
                } TABLE{ 4 49 499 }
```

```
            }

        }
        CURRENT rms_limit{
            MEASUREMENT = rms;
            MAX{ HEADER{
                    WIDTH{ TABLE{0.1 1.0 10}}
            } TABLE{ 0.914 5.153 43.966} }

        }
        CURRENT average_limit{
            MEASUREMENT = average;
            MAX{ HEADER{
                    TEMPERATURE{ TABLE{100 110 125}}
                    WIDTH{ TABLE{0.1 1.0 10}}
            } TABLE {0.229 0.160 0.057
                     2.81 1.96 0.7
                     28.5 19.9 7.1}
         }
        }
    }
}

LAYER metal2{
    PURPOSE = routing;
    LIMIT{
        CURRENT peak_limit{
            MEASUREMENT = peak;
            MAX{ HEADER{
                    WIDTH{ TABLE{0.1 1.0 10}}
            } TABLE{ 2.6 31.85 324.35 }
            }

        }
        CURRENT rms_limit{
            MEASUREMENT = rms;
            MAX{ HEADER{
                    WIDTH{ TABLE{0.1 1.0 10}}
            } TABLE{ 0.59 3.08 25.1} }

        }
        CURRENT average_limit{
            MEASUREMENT = average;
            MAX{ HEADER{
                    WIDTH{ TABLE{0.1 1.0 10}}
                    TEMPERATURE{ TABLE{105 110 125}}
            } TABLE {0.298 2.653 37.209
                     0.208 2.548 25.948
                     0.074 0.912 9.289}
           }
        }
    }
}
```

```
LAYER poly{
    PURPOSE = routing;
    LIMIT{
        CURRENT average_limit{
            MEASUREMENT = average;
            MAX{ HEADER{
                    TEMPERATURE{ TABLE{100 110 125}}
            } TABLE {0.386e-3 0.375e-3 0.347e-3}
           }
        }
    }
}

LAYER polyCont{
    PURPOSE = cut;
    LIMIT{
        CURRENT average_limit{
            MEASUREMENT = average;
            MAX{ HEADER{
                    TEMPERATURE{ TABLE{100 110 125}}
                    AREA{ TABLE{0.0144 0.0288}}
            } TABLE {0.421 0.294 0.105
                     0.842 0.588 0.210}
           }
        }
    }
}

LAYER contact{
    PURPOSE = cut;
    LIMIT{
        CURRENT average_limit{
            MEASUREMENT = average;
            MAX{ HEADER{
                    TEMPERATURE{ TABLE{100 110 125}}
                    AREA{ TABLE{0.0144 0.0288}}
            } TABLE {0.421 0.294 0.105
                     0.842 0.588 0.210} }
        }
    }
}

LAYER via1{
    PURPOSE = cut;
    LIMIT{
        CURRENT average_limit{
            MEASUREMENT = average;
            MAX{ HEADER{
                    TEMPERATURE{ TABLE{100 110 125}}
                    AREA{ TABLE{0.0169 0.0338}}
            } TABLE {0.271 0.189 0.067
                     0.542 0.378 0.134}
```

```
                    }
                }
            }
        }
}
```

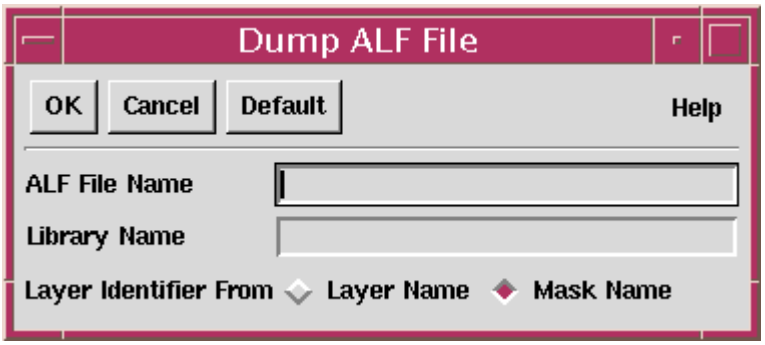
Reporting ALF Information

Write the ALF file out to a text file to verify whether the loading was done correctly.

To report ALF information to an ASCII file,

1. Enter `auDumpALF` in the command window.

   The Dump ALF dialog box appears.



2. Enter the name of the ALF file.
3. Enter the name of the library that contains the design being analyzed.
4. Specify to identify the layer by layer names or mask names (the default).
5. Click OK or Apply.

Deleting ALF Information

If you want to remove the ALF information from the database, use the `auPurgeSIOfAlf` command, like

```
auPurgeSIOfAlf "libName"
```

5

Design Setup and Checking

In addition to library input data, PrimeRail requires the rail setup information that is generated by IC Compiler to be available before performing power and ground net extraction, power analysis and rail analysis. When the rail setup information is loaded, run the checking commands to verify the correctness and consistency of the input data. Then link the power and ground nets before executing any analysis command.

This chapter contains the following sections:

Preparing Rail Setup Data

You can analyze reliability effects of the design in PrimeRail when you have done global routing, detailed routing, or chip finishing in a place and route tool, such as IC Compiler.

Because both PrimeRail and IC Compiler are built on the Milkyway database, PrimeRail is able to read in the necessary information directly from Milkyway where IC Compiler saves the rail setup information.

In IC Compiler, run the `create_rail_setup` command to generate the necessary information for running reliability analysis in PrimeRail. The `create_rail_setup` Tcl command is available in IC Compiler version A-2007.12-SP4 or later. You will also need to run the `set_tlu_plus_files` command to specify the TLUPlus files to be used during extraction. The PrimeRail extraction engine supports the TLUPlus models for resistance effects on power and ground nets and CONN views. If you plan to run PrimeRail static rail analysis within IC Compiler, you do not need to execute this command.

Note:

In version C-2009.06, only PrimeRail static rail analysis is supported within the IC Compiler tool; cell-level dynamic rail analysis is not supported. You need to execute the `create_rail_setup` command for generating necessary information for dynamic analysis in IC Compiler if you are planning to conduct a cell-level dynamic analysis in the standalone version of PrimeRail.

The following information is generated by this command:

- Directory path
- Logical library paths and names
- Operating conditions
- Power supply net names and voltage values
- TLUPlus file path and name
- UPF file

This section includes the following topics:

Creating Rail Setup Data in IC Compiler

When you complete global routing, detailed routing, or chip finishing in IC Compiler, run the `create_rail_setup` command on the design being analyzed to generate the necessary settings for running reliability analysis in PrimeRail.

If you have completed and closed the design, be sure to open the design and source all the variables (such as, .db or TLUPlus files) before running the `create_rail_setup` command.

By default, the command writes data to a directory called synopsys_rail_setup. You can specify a directory for saving output files if you prefer. The output files that will be saved to the directory are

- The SDC file
- The signal parasitic file (in the SPEF format)
- The Verilog netlist
- The synopsys_pr_setup.e file

Then, run the `poLoadRailSetup` command in PrimeRail to load the data saved in the synopsys_pr_setup.e file to the RAIL view (see “Loading Rail Setup Information” on page 5-5).

If you want to view or modify the information saved in the synopsys_pr_setup.e file, run `poDumpRailSetup` to write out the information to an ASCII output file. You can then load this output file back to the tool with the `poLoadRailSetup` command (see “Writing Out Rail Setup Information” on page 5-5).

Note:

The `create_rail_setup` Tcl command is available in IC Compiler version A-2007.12-SP4 or later. Some Milkyway, Astro, Astro-Rail, or PrimeRail Scheme commands also store certain types of information into Milkyway, like `gePrepareLMView`, `cmdReplaceTLUPlus`, and `atTimingSetup`. The data generated by the above Scheme commands will not be taken into consideration by the `poLoadRailSetup` command and other analysis commands in PrimeRail.

To create rail setup data in IC Compiler, type the following in icc_shell:

```
% icc_shell> create_rail_setup
```

The syntax of the command is as follows:

```
create_rail_setup
          [-no_rc_extract] [-hierarchy] [-directory dir_name]
          [-sdc sdc_file] [-spef spef_file]
          [-verilog verilog_file][-parasitic_corner max|min] [-no_save]
```

Place options in parentheses ([ ]). A hyphen (-) precedes option names.

`-no_rc_extract`

Skip the IC Compiler extraction. When specified, the tool will not execute the `extract_rc` command but the `write_parasitics` command to generate the SPEF file based on the data saved in memory. Therefore, make sure the parasitics are in memory before using the `-no_rc_extract` option.

By default, running the `create_rail_setup` command always executes the `extract_rc` command to generate the signal net SPEF file.

`-hierarchy`

Specify this option if you are running a hierarchical or chip-level (ILM-based) flow in IC Compiler. When specified, only Verilog and synopsys_pr_setup.e files will be created in the output directory. If you already have signal parasitics and SDC files and do not want the command to create them (for example, in a full-chip sign-off flow), enable this option.

`-directory dir_name`

Specify the name of the directory where all the output files will be saved. When not specified, the tool saves the output files to the directory, called synopsys_rail_setup.

`-sdc sdc_file`

Specify an existing SDC file to be used. When not specified, the tool will create an SDC file based on the data saved in the database.

`-spef spef_file`

Specify an existing parasitic SPEF file to be used. When not specified, the tool will create an SPEF file based on the data saved in the database.

`-verilog verilog_file`

Specify an existing verilog file. When not specified, the tool will generate a Verilog netlist based on the data saved in the database.

`-parasitic_corner max | min`

Specify the parasitic corner to be either maximum (max) or minimum (min). This option is written into the synopsys_pr_setup.e file for PrimeRail to know which corner to use for power and ground net extraction. By default, the corner is set to maximum (max).

`-no_save`

Specify this option when you do not want to save the cell as *design_name*_pr.

Usage Example

There are different scenarios you may experience while running the `create_rail_setup` command in IC Compiler.

- Run the `create_rail_setup` command without setting any options (the default)

```
% icc_shell> create_rail_setup
```

In this case, the command will write the Verilog netlist and SDC file and run `extract_rc` to write signal parasitics into the SPEF format. All the output files will be named with the name of the current design as a prefix. For example, if the name of the current design is top, the output files will be top.sdc, top.spef, top.verilog and synopsys_pr_setup.e.

- Run the `create_rail_setup` command with user-specified files

```
% icc_shell> create_rail_setup -verilog top.v -sdc top.sdc -spef
```

In this case, the command will use the files specified and create only the synopsys_pr_setup.e file.

- Run the `create_rail_setup` command with the `-hierarchy` option

```
% icc_shell> create_rail_setup -hierarchy
```

The command will create only Verilog netlist and synopsys_pr_setup.e files. In this case, it is recommended that you use the `-sdc` and `-spef` options for specifying the SDC and SPEF files obtained from the sign-off StarRC extraction.

Writing Out Rail Setup Information

Run the `poDumpRailSetup` command to write the data of the synopsys_pr_setup.e file to an ASCII output file that you can load it back to the tool with the `poLoadRailSetup` command.

Here is the syntax of the command:

```
poDumpRailSetup "synopsys_pr_setup.e" "output_file"
```

Loading Rail Setup Information

When you have run the `create_rail_setup` command in IC Compiler and the synopsys_pr_setup.e file is present, run the `poLoadRailSetup` command to read in this setup information and save it to the RAIL view.

To load rail setup information to RAIL view, invoke PrimeRail and open the Milkyway library and cell to be analyzed. It is recommended that you run the `poPurgeRail` command to delete the existing data saved in the RAIL view before loading the data. Doing this ensures a clean RAIL view for analysis.

When the synopsys_pr_setup.e file is available, choose Cell-level Analysis > Design Setup and Checking – Load Rail Setup or enter the following:

```
poLoadRailSetup "synopsys_pr_setup.e"
```

Here, “synopsys_pr_setup.e” is created by running the `create_rail_setup` command in IC Compiler (see “Creating Rail Setup Data in IC Compiler” on page 5-3). The file contains all the necessary information for PrimeRail, such as search_path, link_path operating conditions, supply net voltages, current design information, directory path and so on.

Note:

The `poLoadRailSetup` command will issue an error message if you have run the `poLoadPowerSupply` and `poLinkPGNetlist` commands on the design being analyzed. If this is the case, run the `poPurgeRail` command to delete the existing RAIL view and rerun the `poLoadRailSetup` command.

Reporting Rail Setup Information

When the rail setup file is loaded to Milkyway, run the `poReportRailSetup` command to write what has been saved in the log window, as well as the status of the power and ground netlist. This is to make sure the tool is running the desired flow.

To write the rail setup data to the log window, choose Cell-level Analysis > Design Setup and Checking – Report Rail Setup from the menu or enter `poReportRailSetup 1` in the command window. The tool writes the information of what has been saved to the log window.

Note that by default a full list of the `set_operating_conditions` commands present in the rail setup information from IC Compiler will be written to the PR_LOG_DETAIL/ set_operating_conditions.txt file. If you choose to write the log message to the log window by executing the `poReportRailSetup 1` command, PrimeRail prints all the `set_operating_conditions` commands found to the log window.

Frequently Asked Questions

This section lists the questions that are frequently asked when running the IC Compiler `create_rail_setup` command to generate the necessary rail setup data for PrimeRail.

1. Why the `create_rail_setup` command does not generate the required rail analysis setup data?

   Answer: Make sure the current_design variable is defined and the power and ground nets are of the correct property type. If the tool does not find the current design when executing the command, it reports the following error message:

   ```
   Error: Current design is not set yet
   Please set current design name - the script should be sourced after
   ```

   When you do not define at least one power net and one ground net in the current design, the tool reports the following error message:

   ```
   Power net is not defined
   ```

```
Please define a power net
Ground net is not defined
Please define a ground net
```

Make sure the current design is defined and power and ground nets are properly defined with the correct Milkyway attribute.

2. What should I do if the `create_rail_setup` command reports missing connections, as shown in the following example:

```
Reading reference libraries ...
Info: The design is missing more than 1000 PG connections
Info: Please make sure the PG network is properly connected
Info: Use the command "connect_pg_nets" to make PG connections
Info: The top 1000 missing connections can be found in file
miss_conn.txt
```

Answer: The messages provide the important connectivity information, so you can check the logical connection of the power and ground network in advance. From the above messages, more than 1,000 missing power and ground pin connections are reported. Therefore, you need to run the IC Compiler `connect_pg_nets` command to establish proper connectivity with a power and ground net to correct power and ground ports.

3. How do I check the information saved in the synopsys_pr_setup.e file?

Answer: Use the `poReportRailSetup` command to write out the information to the log window for verifying the information. Alternatively, run the `poDumpRailSetup` command to save the information to an output file for modification and reload.

4. What should I do if my design is analyzed with the hierarchical (ILM-based) flow in IC Compiler?

Answer: Specify the `-hierarchy` option while running the `create_rail_setup` command. In this case, the command creates only the Verilog netlist and the PrimeRail setup file (that is, the synopsys_pr_setup.e file). However, you will need to manually create a PrimeTime PX script for a standalone PrimeTime PX run, instead of using the `poCreatePTPXScript` command to automatically create a script.

It is recommended that you use the signal SPEF file that is generated by StarRC and provide an SDC file for running PrimeTime PX in a hierarchical IC Compiler flow. All other steps remain the same as in the block-level flow.

5. What should I do to run a UPF-based IC Compiler-PrimeRail flow?

Answer: You should run IC Compiler in the UPF mode (icc_shell -upf). Then run the `create_rail_setup` command to write out the UPF file that is to be used by the `poCalculatePower` command.

6. Why the `create_rail_setup` command is saving the design?

Answer: After writing all the necessary information to the synopsys_rail_setup directory, the command by default saves the top cell as *design_name*_pr. This is to record all the design condition to be used in PrimeRail and make sure there is no inconsistency between IC Compiler and PrimeRail database. If it takes too long or you do not want to save it into a different name, set the `-no_save` option with the `create_rail_setup` command.

7. What should I do if the PrimeTime PX script that is generated by the `poCreatePTPXScript` command does not have all the options needed or contains some unsupported commands?

   Answer: Because the script is automatically generated, it may not be feasible for the script to support all the PrimeTime PX options. Manually modify the script if preferred.

8. Why the vector-free PrimeTime PX script with the Switching Activity Format option selected as None has the option "# set power_default_toggle_rate 0.5" commented out?

   Answer: This is to make sure that you are aware of the default switching activity for the vector-free flow is selected as 50% for the PrimeTime PX run. If it is too high, modify it to your desired value.

9. What is the difference between the flows before and after version A-2008.06 of PrimeRail?

   Answer: Beginning with version A-2008.06, PrimeRail changes the steps in data preparation, gate-level power analysis, and power and ground net extraction.

   For more information, see Appendix A, "Backward Compatibility," in this user guide.

10. Do I need to prepare a power and ground specification (pg.spec) file?

    Answer: Beginning with version A-2008.06, the power and ground specification (pg.spec) file is no longer required and the tool automatically extracts the information from FRAM view power and ground pins if the power and ground pins are not found in the libraries.

    However, PrimeRail continues supporting the pg.spec file and honors it with the highest precedence. It is highly recommended that you add the power and ground pin information to non-power-and-ground pin logical libraries, especially for multiple VDD libraries, by using the Library Compiler utility before running PrimeRail. It is also recommended that you run the Library Compiler checker in IC Compiler, rather than in PrimeRail.

11. How the PVT (process, voltage and timing) corner is selected by PrimeRail?

    Answer: PrimeRail analyzes one corner at a time and errors out if the main library voltage supply is different from that defined with `set_operation_conditions` (that is written from IC Compiler), `set_voltage` or the voltage value found in the PrimeTime PX binary.

    The IC Compiler `extract_rc` command, which writes signal parasitics into the SPEF format for the PrimeTime PX run, uses the temperature value defined in `set_operating_conditions`. When not found, the command uses the temperature

value from the .db file. Because this temperature value needs to be used during power and ground net extraction in PrimeRail, the information is passed to PrimeRail through the synopsys_pr_setup.e file.

To make sure the PrimeTime PX generated power report is consistent with the expected behavior, examine the PVT corner information in the log when the run of the `poCalculatePower` command is complete. PrimeRail automatically synchronizes with the signal parasitic corner selected in IC Compiler during power and ground net extraction.

Checking Design Data

The PrimeRail tool runs on the Milkyway design database. If your library data is inconsistent or insufficient for completing a job, PrimeRail fails in executing the analysis command, because PrimeRail cannot acquire the necessary information from the database.

A complete Milkyway database with all the reference libraries linked should be verified to make sure that

- The representations of the data, such as units, in the related libraries are consistent
- The library has the minimum information required for power analysis, power and ground net extraction, and rail analysis

PrimeRail provides the `poCheckDesignDB` command that automatically checks for design database consistency and design data sufficiency and debugs the problems that may occur due to inconsistency, corruption, or lack of data. The `poCheckDesignDB` command works on a complete database, diving into the hierarchy as necessary to ensure the functionality of the rail-analysis-related commands.

Important:
It is recommended that you check on the design data throughout the design verification project to ensure that all the necessary design information for PrimeRail analysis is handed over. Consistency checking is needed to debug any problem that may occur.

The `poCheckDesignDB` command reports an error or a warning, depending on which type of data is incorrect or missing. For example, if data is missing and PrimeRail stops the task, an error is reported. If the tool continues to run but the results might not be as accurate (for example, a CONN view of a hard macro), a warning message is issued.

Currently, the `poCheckDesignDB` command is able to check

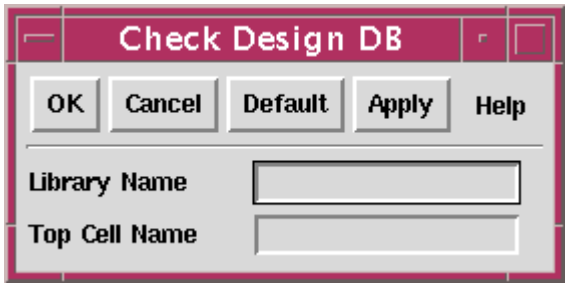
- Whether all the reference libraries are consistent in unit and setting
- Whether the top-level library has the required settings
- Whether soft macros in the design have the following:

  - FRAM view with power and ground pins
  - White box model
- Whether hard macros in the design have the following:
  - CEL view
  - CONN view
  - FRAM view with power and ground pins
- Whether there is a port mismatch between FRAM view and CONN view

To run the `poCheckDesignDB` command,

1. Enter `poCheckDesignDB` or choose Cell-level Analysis > Design Setup and Checking – Check Design DB.

   The Check Design Database dialog box appears.



2. Enter the name of the library and the top cell to be examined.
3. Click OK or Apply.

   PrimeRail writes the checking results to a log file and to the command window.

Log Message

The following is a sample message that the `poCheckDesignDB` command writes to the command window and to the default PrimeRail log file, named PrimeRail.log.DATE.

```
Opening library design
Opening top cell top
Technology Table opened sucessfully
Technology data ....

   ******* TECH DATA LISTING *******
   UNIT and CONSTANT---------------
                          Unit              UserUnit
    distance              micron            1000
```

```
    time                  ns              1000000
    capacitance           pf              100000000
    resistance            kohm            100000000
    inductance            nh              100
    voltage               v               100000000
    current                               0
    power                 mw              1000
    dielectric            constant        3.900000e-05

  LAYER and RULE--------------
   NumberOfWiringLayers: 6
   Layername :  CP (13) (Maskname: poly)
   Capacitance: 0.000000
   Resistance:  0.000000
   Minimum width: 240
   Minimum Spacing: 360
   Max current density: (UNDEFINED)

   Layername :  METAL (16) (Maskname: metal1)
   Capacitance: 2870.000000
   Resistance:  6000.000000
   Minimum width: 320
   Minimum Spacing: 320
   Max current density: 2

   ViaLayername :  VIA (17) (Maskname: via1)
   Capacitance: 0.000000
   Resistance:  0.000000
   Minimum width: 360
   Minimum Spacing: 440

  CONTACT CODE------------------
   Contact Code 1: lower layer 16, via layer 17,
     upper layer 18, area 129600
   Contact Code 2: lower layer 16, via layer 17,
     upper layer 18, area 129600

******* CELL DATA LISTING *******
  TOP CELL: top (library: design)
   CEL  view      : Yes
   PARA view      : Yes
  SOFT MACRO counter_module (library: design)
   CEL  view      : Yes
   FRAM view      : Yes
   PARA view      : Yes
  SOFT MACRO rom_64x64 (library: aspc_mem_lib_fr)
   CEL  view      : Yes
   FRAM view      : Yes
   CONN view      : Yes
   PARA view      : Yes
   White-Box Model: Yes
   WARNING: Hard macro has a CEL view
  SOFT MACRO timer_0 (library: design)
```

```
   CEL  view      : Yes
   FRAM view      : Yes
   PARA view      : Yes

******* DESIGN HIERARCHY LISTING *******
 top (TOP CELL, #_of_stdcells: 1003)
 + core_inst/user_ram_inst/user_ram_0
       (cell: ram_16x2048, #_of_stdcells: 5)

******* PG CONNECTIVITY LISTING *******
 top (TOP CELL) VDD_Net:   GND_Net:
 + core_inst/user_ram_inst/user_ram_0 VDD(IP: VDD, PN: VDD)
     GND(IP: VSS, PN: VSS)
 + core_inst/new_soft_macro VDD(IP: VDD, PN: VDD) GND(IP: VSS, PN: VSS)

******* FRAM vs CONN VIEW CHECKING *******

 library "/remote/snps2/LIBRARIES/SYNOPSYS/PLL_tdd_B", cell
"PLL_tdd_B_TOP": number of ports in FRAM and CONN views do not match (11
vs 4)
 library "/remote/snps2/SYNOPSYS/PLL_tdd_A", cell "PLL_tdd_A_TOP": number
of ports in FRAM and CONN views do not match (13 vs 4)
 library

Checking Design DB  Memory: 1.593e3 MB  CPU: 92 seconds  Elapse: 2.772e3
seconds
```

Checking Logical Connectivity

For PrimeRail to correctly analyze a hierarchical design, different levels of hierarchy must be both physically and logically connected in various power and ground nets.

PrimeRail provides the `poCheckPGNetlist` command to check for logical connectivity between the power and ground nets of each cell instance to its parent. The command reports an error if a cell instance has one or more power or ground nets, but none of them is connected to the power or ground net at the top level. For example, if a cell instance has two nets, vdd1 and vdd2, and the vdd1 net is connected to the VDD net at the top level but the vdd2 net is floating, PrimeRail flags an error on the vdd2 net.

The command checks if

- An instance power pin connected to either signal net or ground net
- An instance ground pin connected to either signal net or power net
- An instance power or ground pin not connected to top-level power or ground net
- A root power or ground port (or net) does not have a power supply value

Note:

The command cannot be run before the `poLoadRailSetup` command is executed. However, you can choose not to run the `poCheckPGNetlist` command if you have full confidence in the data quality; that is, you have performed the checking in the previous run.

Cell masters for which no power and ground ports exist are flagged and the corresponding cell instances are skipped.

To check the power and ground netlist saved in the Milkyway database, choose Cell-level Analysis > Design Setup and Checking – Check PG Netlist or type the following in the command window:

```
poCheckPGNetList
```

The tool then reports the numbers of errors found in the log window:

```
poCheckPGNetlist Summary

Total Number of Errors ..... 0
Total Number of Warnings .... 49
Total Number of Info Msg .... 50
```

The details of the errors are reported in an output log file, named PR_LOG_DETAIL/poCheckPGNetlist_rpt.txt.

Checking Physical Connectivity

You can check a power or ground net for unconnected cell instances, floating geometries (disconnected wires and vias), and possible missing vias by using the `poRailChecking` command. If you are analyzing a hierarchical design, use the command to check for missing vias across two levels of design hierarchy or missing macro pins in the design.

Note:

The command does not check the rail data of white box macro cells.

The `poRailChecking` command allows you to save the results in an error cell that you can examine by choosing Diagnosis > Verify. An error cell is created for each cell master found in the design. Thus, if the design is hierarchical, the `poRailChecking` command creates multiple error cells. This means that you can investigate any errors on a cell-master-by-cell-master basis.

The error information for the top level is saved into the error cell with the name you specify. If no error cell name is given, it is named as follows:

*cellName_netName*_RAIL.err

If the reference library of the cell examined is not writable and no error file is specified with the Flatten Hierarchical Cells option enabled, PrimeRail skips the checking and writes warning messages like the following:

```
Checking net VDD of cell timer_0.CEL ...
Create error cell timer_0_VDD_RAIL.err ...
WARNING : unable to open error cell timer_0_VDD_RAIL.err
WARNING : no error cell/file. Skip checking the cell
```

However, if the reference library of the cell being examined is not writable, but you specify an error file with the Flatten Hierarchical Cells option enabled, PrimeRail finishes checking and writes the results in the error file you specify. The message in the command window is like the following:

```
Checking net VDD of cell timer_0.CEL ...
Create error cell timer_0_VDD_RAIL.err ...
WARNING : unable to open error cell timer_0_VDD_RAIL.err
result will only show in error file: [filename]
```

Checking Discontinuity Between Blocks

Rather than checking possible missing vias at the same hierarchical level, PrimeRail enables you to check overlapping metal rails for missing via connections across two levels of design hierarchy.

Checking Macro Pins

When extracting resistances for power and ground nets in a hierarchical design, PrimeRail assumes that all the connections across the layer are made only through the FRAM and CEL pins. The tool promotes the pins in the lower macros to the upper macro for top-level extraction and creates boundary nodes from the pins for establishing connections across the macros.

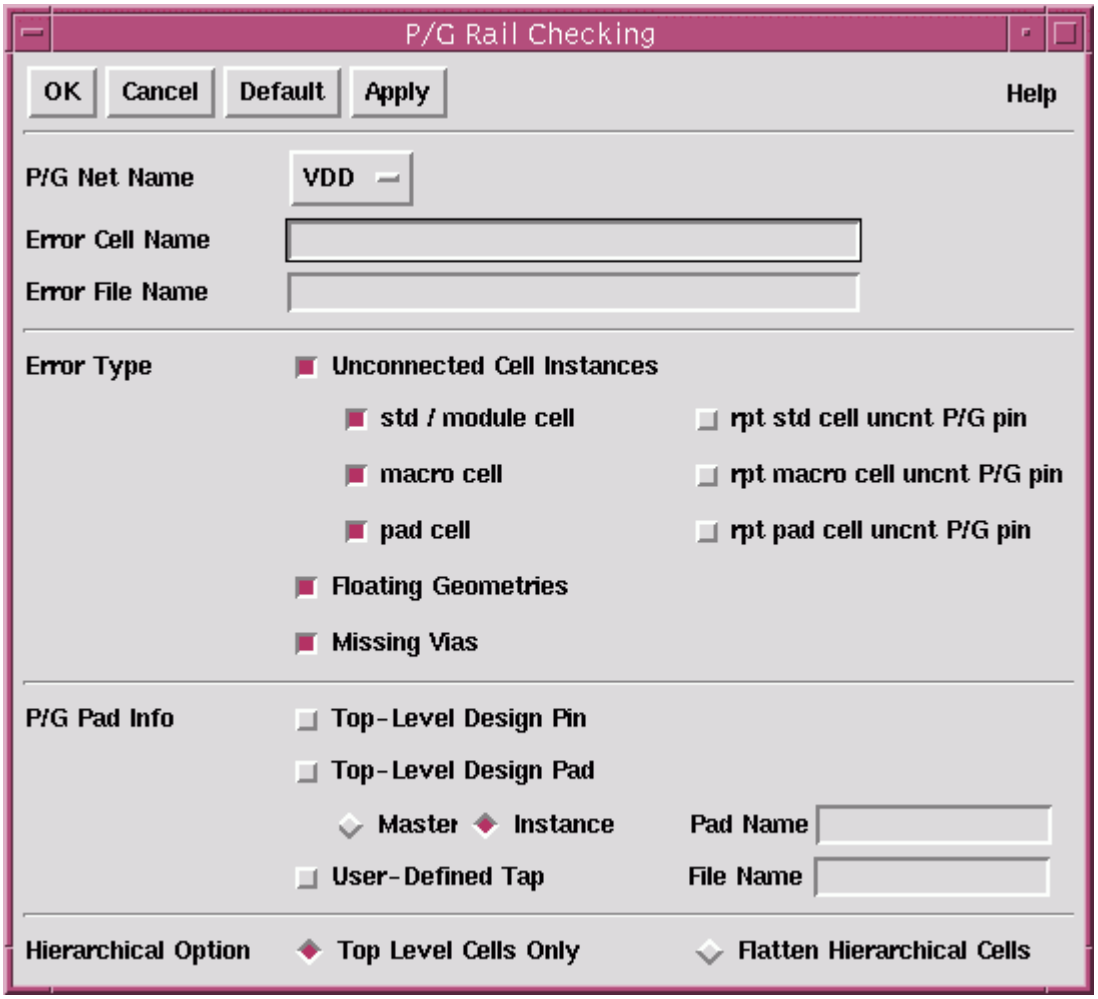
However, the above assumption might not be correct because some designs might have macros connected (or overlapped) beyond the FRAM pins or make the connection from lower level to upper level by overlapping wires, thus resulting in inaccurate resistance extraction.

With the `poRailChecking` command, you can check whether the macros are connected by pins or the overlapping wires are well connected.

To check and save the rail data of a cell to a file,

1. Enter `poRailChecking` or choose Cell-level Analysis > Design Setup and Checking – Check PG Net.

   The P/G Rail Checking dialog box appears.



2. Choose the name of the power and ground net you want to check.
3. Enter the name you want to assign to the error cell.
4. Enter the name of the error file in which you want to write all the error information in the ASCII format.
5. Select the types of errors you want to check:
   - Unconnected Cell Instances – Check cell instances that have zero connected cell instance pins.

Cell type	PrimeRail does this
std / module cell	Reports any unconnected standard cells.

Cell type	PrimeRail does this
macro cell	Reports any unconnected macro cells.
pad cell	Reports any unconnected pad cells.
rpt std cell uncnt P/G pin	Reports standard cells that are not connected to the power and ground pin. By default, this option is off.
rpt macro cell uncnt P/G pin	Reports macro cells that are not connected to the power and ground pin. By default, this option is off.
rpt pad cell uncnt P/G pin	Reports pad cells that are not connected to the power and ground pin. By default, this option is off.

- Floating Geometries – Find wires and vias that are not connected to the power and ground net, including
  - Floating top-level power and ground pins
  - Single floating wire and via
  - A subnet that is not connected to any ideal voltage source
  - A via that is not connected to any wire at one end

  When you select this option, be sure to specify the source of ideal voltage in the P/G Pad Info section at step 6. The `poRailChecking` command needs the information to check which wire or via is not connected to the net.

  Note:
  If you want PrimeRail to ignore any floating vias in the design when enabling the Floating Geometries option, write the following line in the command window before you execute the `poRailChecking` command:

  ```
  define poNotReportFloatVias 1
  ```

- Missing Vias – Find possible missing vias on the power and ground net.
- 2-Level Missing Vias – Find possible missing vias of the overlapping metal rails across two levels of design hierarchy.
- Missing Macro Pins – Find possible missing pins that connect macros to other layers.

6. In the P/G Pad Info section, specify the ideal voltage sources if the Floating Geometries option is selected. This is required only if you want to check for floating geometries of the net in question.
   - Top-Level Design Pin – Find the associated power and ground pins, and treat them as the ideal voltage sources.

- Top-Level Design Pad – Use the existing top-level design pads as the ideal voltage sources by

  - Master

  - Instance (the default)

  - Pad Name – Enter a file name that contains the master or cell instance names of the power and ground pad.

- User-Defined Tap – Use the user-defined pad locations as the ideal voltage sources, and enter the name of the file that contains the power and ground pad location information, in the following format:

  ```
  netName padLayerNumber Xcoordinate Ycoordinate
  ```

7. In the Hierarchical Option section, select Top Level Cells Only if you want to check the top level only. Select Flatten Hierarchical Cells if you want to check all the hierarchical macro cells.
8. Click OK or Apply.

Linking Power and Ground Netlist

When rail setup information has been saved to the RAIL view with the `poLoadRailSetup` command, you must link the power and ground netlist that is saved in RAIL view using the `poLinkPGNetlist` command. Once the link is successfully established, an analysis command, like `poExtractPGParasitics` or `poCalculatePower`, can be run. In other words, you cannot proceed to extraction and transient power analysis before `poLinkPGNetlist` is run. The execution of the `poLinkPGNetlist` command will fail if either the .db file, TLUPlus file, or supply net voltage is missing.

Enter `poLinkPGNetlist` or choose Cell-level Analysis > Design Setup and Checking – Link PG Netlist.

Here is the syntax of the `poLinkPGNetlist` command:

```
poLinkPGNetlist
```

6

Library Characterization for Cell-Level Dynamic Analysis

Running cell-level dynamic analysis in PrimeRail requires the characterization of the gate-level cells in addition to the established library characterization data.

PrimeRail provides an automatic and built-in characterization capability by using the HSPICE technology. Characterization is performed on a master basis and you need to run the characterization only once for each library and each corner. The characterization results are saved in the Milkyway library.

Characterization runtime usually requires only a few hours when performed on a network of distributed CPUs.

Note:

You do not need to run library characterization if your library files are created with a CCS power library.

This chapter describes how to characterize intrinsic parasitics and current waveforms in the following sections:

About Library Characterization

Running a cell-level dynamic analysis flow is like performing a full-chip analysis on a design that is given at the gate-level representation without sacrificing the capacity and runtime benefits this representation provides. At the same time, the dynamic noise waveforms of the power supply network have first-order contributions from the physical effects that are not accessible at the gate-level representation. These effects are

- Intrinsic parasitics of logic gates and decoupling capacitors
- Dynamic current waveforms associated with each switching event

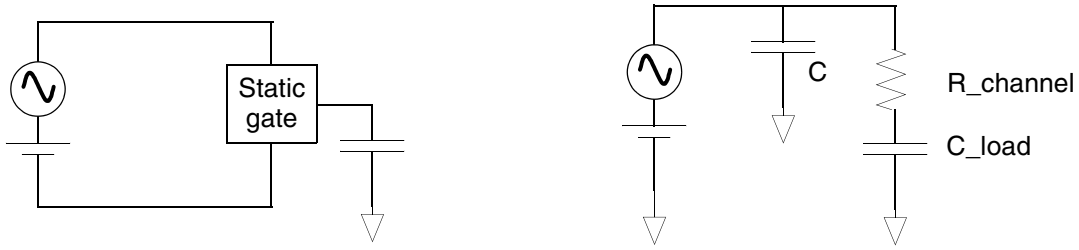
Library characterization in PrimeRail is capable of capturing these effects in a lookup table, so the impact of the effects can be included when dynamic current waveforms are generated from the gate-level design representation. Characterizing parasitics of power management cells and leakage currents of filler cells is also supported.

Intrinsic Parasitics

Intrinsic parasitics provide a significant contribution to the total capacitive loading of the power supply network seen in a CMOS circuit. They denote an additional loading of the power supply network due to the signal net load of logic gates in the circuit. This additional load is relevant because a direct current path exists from the power supply network to this load, depending on the state of the gate. The nominal output load is shielded by the channel resistance of the gate output driver. If the gate is static, the channel resistance is fairly large.

Shown in Figure 6-1, during intrinsic characterization PrimeRail first captures an effective capacitance value by performing an AC analysis, in which the sinusoidal waveform is added to power and ground pins and the capacitance value is calculated based on magnitude and phase of the current response. PrimeRail then performs a DC analysis to capture an effective resistance value which represents the channel resistance. The characterization output of a static gate consists of one resistance and one capacitance. However, the full representation of such gate during rail analysis is a PI model, where the first capacitance and resistance values are from characterization, and the load capacitance is stitched as the second capacitance value. This means that there is no load dependency during library characterization in PrimeRail.

Figure 6-1 Characterizing and Validating Circuit Structure



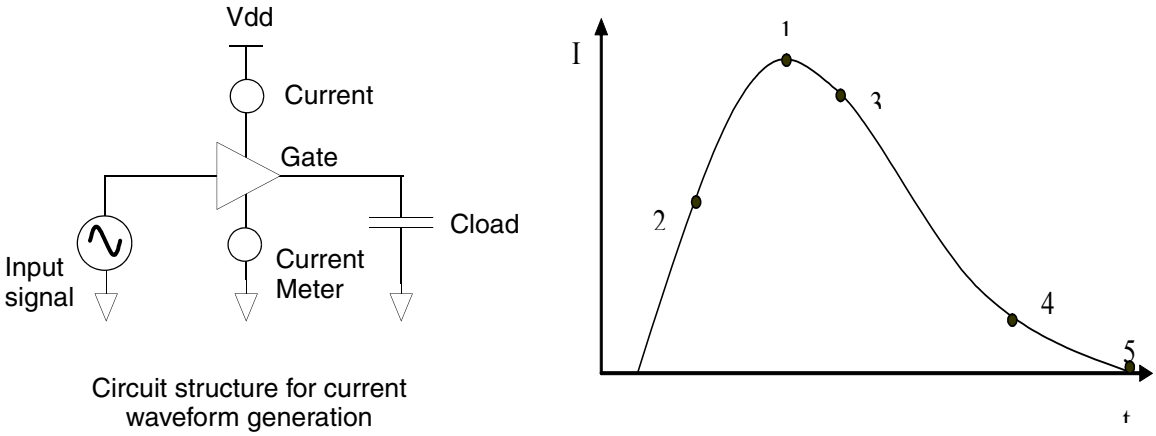
Characterization circuit structure:
Reff/Ceff from VDD

Validation circuit structure:
Reff/Ceff from VDD

Current Waveforms

According to the Liberty standard, the existing gate-level power models capture the total energy dissipated by a cell for a given switching event. This is not sufficient for the dynamic cell-level analysis in PrimeRail. The detailed dynamic current waveform drawn from the power supply network by the cell for a given switching event is needed for generating accurate analysis results, as shown in Figure 6-2.

Figure 6-2 Generating Current Waveforms



Circuit structure for current waveform generation

Points recorded for current waveforms

The PrimeRail current waveform characterization runs HSPICE for a range of input transitions and output loads for each timing arc. This is like the delay characterization, except the current waveforms drawn from the supply network are to be captured. The waveforms are converted to an internal format that is similar to piecewise linear format and are indexed in a lookup table.

If insufficient information is available to run HSPICE, PrimeRail generates current waveforms based on the Liberty timing tables. If timing tables are unavailable, a simple triangular waveform is used. Both of the mechanisms are fallback solutions and accuracy will be degraded. It is strongly advised to provide enough input data to run HSPICE characterization.

Power Management Cell Characterization

The rush current during power up (or down) sequence flow is more like an analog signal, which requires to accurately model the power management (or multithreshold-CMOS) cell. It is found that current-based models are necessary for accurate rush current analysis. During power management cell characterization, the current-based model consists of a current table (a set of IV curves) as a function of control pin voltage level and virtual power or ground pin voltage level. By using those IV curves, the tool is capable of accurately modeling the current drawn from the actual power or ground pin to the virtual one for arbitrary waveforms at the control pin and for various loads at the virtual power or ground pin. During library characterization, the tool captures IV curves by using DC analysis in HSPICE, where the current drawn from the actual power or ground pin to the virtual power or ground pin is measured by varying the voltage level at the control pin and the virtual power or ground pin.

If the power management cell consists of prelogic gates in front of the power management transistor, the timing information—in terms of delays at certain thresholds—will be captured to restore accurate voltage waveforms in the input pin of the transistor.

Filler Cell and Decap Cell Leakage Characterization

The Liberty tool does not analyze leakage currents for filler or decoupling capacitance (decap) cells, except in the CCS power library. However, such information is necessary during rail analysis as well as decap insertion flow in PrimeRail. The circuit structure for leakage current characterization is very similar to that of intrinsic parasitic characterization, except it is the DC current at each power or ground pin that is measured.

Before Characterization

In order to run SPICE for library characterization, you must

- Make sure the Milkyway reference library of the library to be characterized is open.
- Have access to the HSPICE binary and be able to execute it. PrimeRail will invoke the binary for the characterization.
- Have the Liberty models (that is, Synopsys .db files) or the Milkyway LM views available.
- Have a power and ground port specification file (pg.spec) for each reference library (even it is not characterized).

Preparing Power and Ground Port specification File (pg.spec)

The power and ground port specification file lists the power and ground ports used in a given library, the voltage levels for each port, as well as the mapping information for the power and ground ports. The mapping information is needed to express current source and sink.

Specify this power and ground port specification file in the P/G Port Spec File text box of the Library Characterization dialog box (see Figure 6-3 on page 6-7).

The following is an example of the power and ground port specification file:

```
definePowerPort                 "VCC" 1.08
definePowerPort                 "VBB" 1.2
definePowerPort                 "VDD" 1.32
defineGroundPort                "VSS" 0
defineGroundPort                "GND" 0
defineDefaultPowerPort          "VDD"
defineDefaultGroundPort         "VSS"
defineGroundPowerMapping        "GND" {"VCC" VDD"}
defineGroundPowerMapping        "VSS" {"VBB"}

*Specify well capacitance values*
defineUnitWellCap               value
defineWellCap "cellA"           "VDD" value
```

The voltage levels specified in the power and ground specification file have to be consistent with those in the Liberty models (.lib or .db files). Otherwise the results of the subsequent cell-level dynamic current waveform generation will be inaccurate.

Specifying Well Capacitance Values

You can also specify the well capacitance value that is to be added to each cell master port during rail analysis in the power and ground specification file. Use `defineUnitWellCap` to have the tool automatically calculate the well capacitance value for each power and ground port, based on the unit capacitance per unit area. The unit of the well capacitance value is farad/square meter. For example, if you want to specify it as 1ff/um2, write it as 0.001 in the statement.

Or you can use the `defineWellCap` command that simply takes any value you specify in the power and ground specification file. It is recommended that you use either `defineUnitWellCap` or `defineWellCap` to include well capacitance values. When both statements are used, `defineWellCap` always has higher priority. In other words, the capacitance value specified with `defineWellCap` overwrites what is automatically generated by `defineUnitWellCap`. Other ports that are not specified through `defineWellCap` will not be affected.

To check the well capacitance values to be used during rail analysis, type the following in the command window before invoking the `poCalculatePower` command:

```
define pgDumpWellCap 1
```

The tool writes the well capacitance values to an output file, called wellCap.dmp, in the working directory. If the design has multiple reference libraries, all the well capacitance values will be reported. Note that if the well capacitance output file already exists before rail analysis is performed, the tool will append the values to the end of the existing file.

The following is an example of the well capacitance output file:

```
# of entries = 52
bufx5.vdd=> 2.565e-15
bufx5.vss=> 2.565e-15
adder2.vdd=> 3.476e-15
adder2.vss=> 3.476e-15
```

Characterizing Cell Data

The `pgLibCharacterize` command enables you to complete the library characterization process in one graphic user interface. Thus it is unnecessary to run multiple commands—`pgSpiceSetup`, `pgPreCharacterize`, `pgLinkPGSpec`, and `pgLinkCharacterize`—to complete the entire process. By default, the characterization of current waveforms, intrinsic parasitics, power management cells, and filler cell leakage currents is completed in a single run. If anything goes wrong with `pgLibCharacterize`, you can use the original four commands (that is, `pgSpiceSetup`, `pgPreCharacterize`, `pgLinkPGSpec`, and `pgLinkCharacterize`) to rerun the process from the necessary point of the flow.

For a detailed description about running the `pgSpiceSetup`, `pgPreCharacterize`, `pgLinkPGSpec`, and `pgLinkCharacterize` commands, see the online Help for the commands.

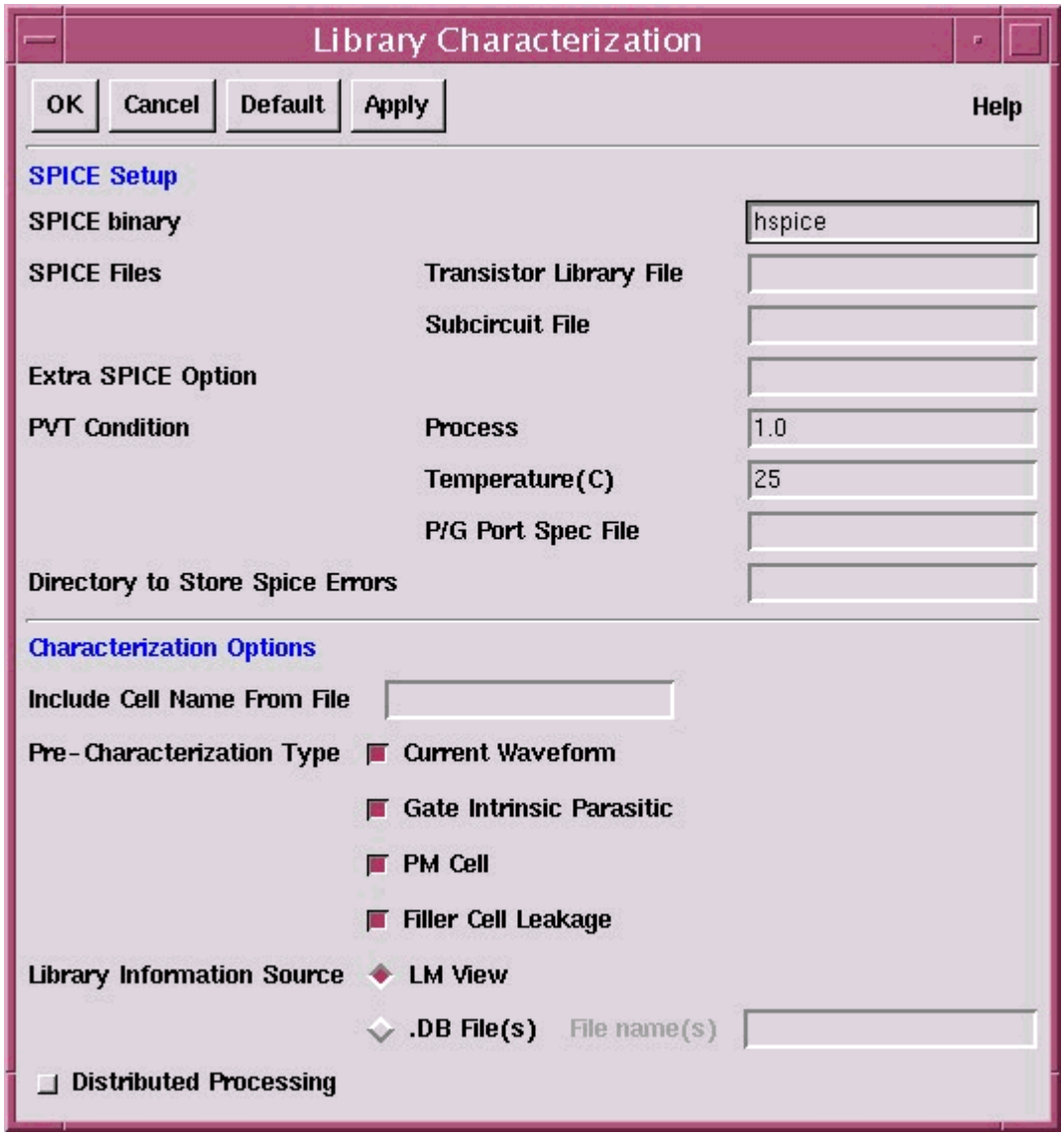
When characterization is done, run `pgListCharResult` with the “detail” option to save the characterized library to an output file for checking if the cells are characterized with correct PVT corners.

To characterize a cell,

1. Open the library to be analyzed. Enter `pgLibCharacterize` or choose Library Characterization > Library Characterization.

   The Library Characterization dialog box opens.

Figure 6-3 LIbrary Characterization Dialog Box



2. In the SPICE setup section, configure the generic HSPICE settings for characterizing cells. This is to configure additional settings for the HSPICE characterization run.

   - Enter the name of the SPICE binary that you want to use. Including the full path is unnecessary because the tool is capable of finding the binary by searching the $PATH environment.
   - In the SPICE Files section, enter the names of the files to be used.

     Transistor Library File – Enter the name of the file that contains device models used in the subcircuit descriptions for standard cells.

Set any configuration settings for the device models that are needed to select a specific process corner in this transistor library file.

Subcircuit File – Enter the name of the SPICE netlist file that contains subcircuit description definitions for all the decoupling and standard cells to be characterized.

- In the Extra SPICE Option text field, enter additional SPICE options you want to run.
- In the PVT Condition section, specify the necessary configuration settings.

  Process – Enter the exact process specified in the .lib file. The process number has to be exactly the same as in the .db file. If the .db file has multiple processes, you can run multiple library characterizations by choosing different processes.

  The default is 1.0.

  Temperature (C) – Enter the temperature to be used in the SPICE simulations (in degree Celsius). This value must be consistent with the Library model (.lib or .db file). The default is 25.

  P/G Port Spec File – Enter the name of the file that defines the power and ground ports in the design to be analyzed.

  See “Preparing Power and Ground Port specification File (pg.spec)” on page 6-5 for details about creating the power and ground port specification file (pg.spec).

- In the “Directory to Store Spice Errors” text box, enter the name of the directory where you want to save the error messages.

3. In the Characterization Option section, specify the options to characterize both the intrinsic parasitics and the current waveform tables for the cells under characterization. PrimeRail requires the intrinsic parasitics and current waveforms of decoupling capacitance cells to be characterized for dynamic analysis.

   - Enter the name of the file that contains the cell names to be characterized.

     By default, PrimeRail automatically scans each cell in the FRAM view and the .db file (or LM view) and runs characterization for the uncharacterized cell, as long as the corresponding SPICE subcircuit exists. By specifying a file with a list of cells, you can request the characterization to be run only on the cells that are listed in the file.

     Specifying the cell name file is optional.

   - Pre-Characterization Type – Select the type of characterization you want to run.

     Current Waveform – Characterizes current waveforms.

     Gate Intrinsic Parasitic – Characterizes intrinsic parasitics.

     PM Cell – Characterizes power management cells when you are running an in-rush analysis. For more information, see “In-Rush Analysis” on page 11-6.

Filler Cell Leakage – Characterizes leakage current information for filler cells before inserting a decoupling capacitor. For more information, see “Decoupling Capacitor Insertion” on page 12-10.

- Library Information Source – Select the source of the library to be used for characterization. You can choose to obtain the library information from the LM view or the existing Synopsys .db files.

  If .db files are to be used, enter the names of the files in the “File name(s)” text box.

- Distributed Processing – Distributes multiple HSPICE jobs to the machines and concatenates the results if you are running a parallel characterization.

  For more information, see “Distributing Multiple Characterization Processes” on page 6-9.

4. Click OK or Apply.

   When the characterization process is done, PrimeRail generates characterization output files. Depending on the type of characterization selected in the Library Characterization dialog box, the output files are named *libChar_reflibName*.cw, *libChar_reflibName*.cin, *libChar_reflibName*.pm, and *libChar_reflibName*.fc.

   When you rerun the characterization, PrimeRail first checks whether any output files exist in the directory. If they are corrupted, PrimeRail preserves as much of the previously calculated information as possible and restarts the run. Otherwise PrimeRail assumes that you want to start a new run (perhaps with different options) and removes the existing files.

Distributing Multiple Characterization Processes

If you want to perform multiple library characterization processes in parallel, run the `pgParallelJob` command before executing the `pgLibCharacterize` command. The `pgParallelJob` command automatically distributes HSPICE runs to multiple machines with or without the use of the load-sharing facility (LSF) farm.

When performing characterization on SPICE simulation, the tool creates a subdirectory in the working directory, called .libChar_reflibName, to store intermediate files. For intrinsic parasitics characterization, these files are named by the cell being characterized. For current waveform characterization, they are named by a string constructed to contain information about the particular arc being characterized. These intermediate files are deleted at the end of a successful run.

If you want to keep these intermediate files without being deleted, type the following in the command window before invoking the `pgParallelJob` command:

```
define rfKeepSpiceFiles 1
```

If the characterization fails, all the files pertaining to that arc or cell will be copied to the directory as specified in the “Directory to Store Spice Errors” text box of the Library Characterization dialog box (see Figure 6-3 on page 6-7).

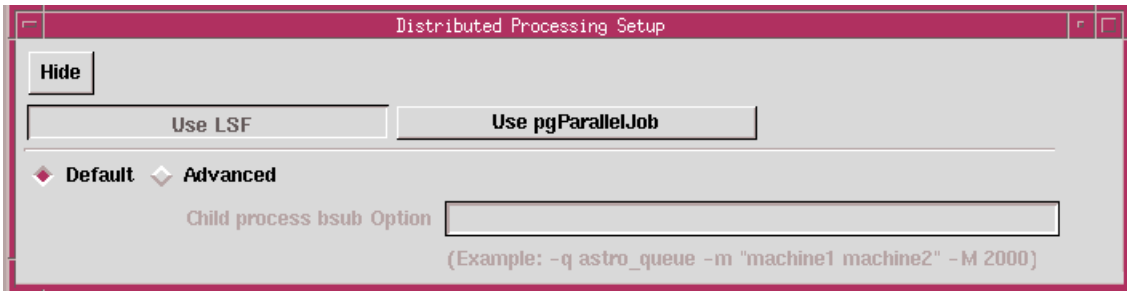
Note:
Each machine to be distributed needs to have an HSPICE license used during library characterization. You need only one PrimeRail license.

To distribute multiple characterization processes:

1. Enter `pgParallelJob` in the command window or choose Library Characterization > Distributed Process Setup.

   The Distributed Processing Setup dialog box appears, as shown in Figure 6-4.

Figure 6-4 The Distributed Processing Setup Dialog Box



2. If an LSF farm is available, click Use LSF and specify the “Child process bsub Option” for distribution.

   If you do not have an LSF farm, click “Use pgParallelJob” and enter the names of the machines to which HSPICE jobs will be distributed and the PrimeRail binary location (see Figure 6-5 on page 6-11).

Figure 6-5 Specifying Settings for Distributing Multiple Characterization Processes

3. Click Connect and then Hide to apply the changes you make and close the dialog box.

   PrimeRail will employ the maximum number of CPUs detected by `pgParallelJob` when distributing HSPICE processes. However, each time a process is distributed, PrimeRail looks for the fastest available machine, rather than always dividing jobs evenly among the CPUs. Therefore, you will achieve the maximum speed benefit if the machines, where the HSPICE run is to be distributed, are similar in speed and load.

4. Next run the `pgLibCharacterize` command to perform the library characterization. In the Library Characterization dialog box (see Figure 6-3 on page 6-7), select the Distributed Processing option.

   Specify other options as necessary in the Library Characterization dialog box.

For more information about running library characterization, see “Characterizing Cell Data” on page 6-6.

Validating Characterization Results

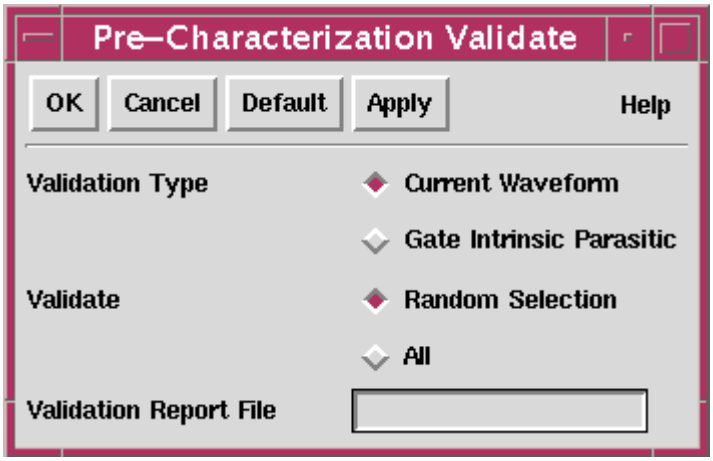
Run the `pgValidateLib` command to validate the tables (either intrinsic capacitance or current waveform) generated by the library characterization against the original SPICE descriptions of the cell.

To validate the characterized data,

1. Enter `pgValidateLib` or choose Library Characterization > Validate Library Characterization.

   The Pre-Characterization Validate dialog box appears, as shown in Figure 6-6.

Figure 6-6 The Pre-Characterization Validate Dialog Box



2. Validation Type – Choose to validate current waveform or intrinsic parasitic tables.
3. Validate – Choose to validate a randomly chosen subset of the total cells or all the cells found in the library.
4. Validation Report File – Enter the name of the file where the validation report is to be saved.
5. Click OK or Apply.

Writing Characterization Results to ASCII Files

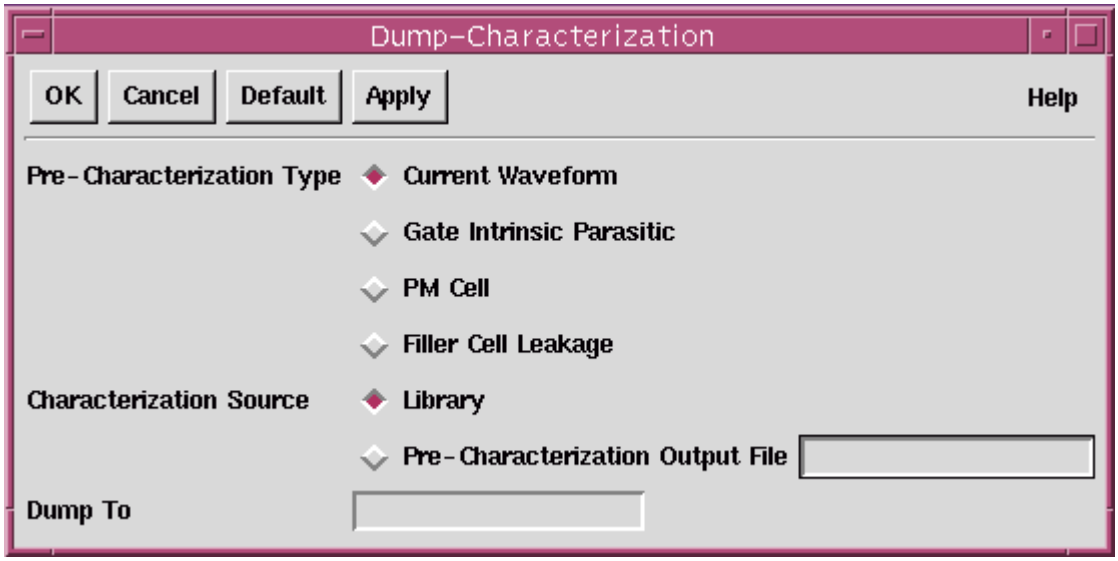
You can choose to write the characterization results from the characterized reference library or from the characterization output file to an ASCII file.

To save the characterized results to an ASCII file,

1. Enter `pgDumpCharacterize` or choose Library Characterization > Dump Library Characterization.

   The Dump-Characterization dialog box appears, as shown in Figure 6-7.

Figure 6-7 The Dump-Characterization Dialog Box



2. Pre-Characterization Type – Select the type of the characterization data.

   You can choose to translate current waveform or intrinsic parasitic tables. Or you can choose to write the power management cell data (in ASCII format) or the filler cell leakage data to a file.

3. Characterization Source – Choose whether to translate the tables attached to the current Milkyway library or from the output characterization file by `pgLibCharacterize`.

4. Dump To – Enter the name of the output ASCII file where the translated characterized data or power management cell data is to be saved.

5. Click OK or Apply.

Listing Characterization Results

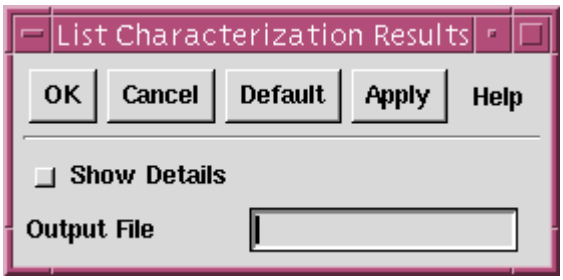
Run the `pgListCharResult` command to save the list of the cells of a reference library to an output file for checking the status of the characterization data. In the output file, each of the cells is listed under current waveform, capacitance intrinsic, and filler cell leakage tables. From the table you can easily tell whether the cell was characterized; what the PVT corner is; and also whether the cell has an LM view or a FRAM view. The PVT corner has an index for the reference file.

To list the characterization results in an output file,

1. Enter `pgListCharResult` or choose Library Characterization > List Library Characterization.

   The List Characterization Results dialog box opens, as shown in Figure 6-8.

Figure 6-8 The List Characterization Results Dialog Box



2. Select Show Details if you want to include information on whether the cells in the reference library are in an LM or a FRAM view and their PVT conditions. When deselected, the tool lists only the information on how many cells are characterized for current waveform, capacitance intrinsic, and filler cell leakage.
3. Enter the name of the file where the characterization results are to be saved.
4. Click OK or Apply.

Example

The following is an example of the listed characterization results, with the Show Details option selected.

```
The Summary of Pre-characterization Results
-------------------------------------------------------------
Process  Voltage   Mapped_PG_Value  Temperature    Index
  1.0      2.50       0.00              25          "0"

-------------------------------------------------------------
List of Pre-Characterization Results for Current Waveform
```

```
Cell_Name             LM_View  FRAM_View   CW_Result  PVT(index)
lanhn1                  *         *
lanhn2                  *         *
nr04d1                  *         *          from SPICE    "0"
nr04d2                  *         *
nr04d4                  *         *
or02d0                  *         *          from SPICE    "0"
or02d1                  *         *
or02d2                  *         *         from SPICE    "0"

-----------------------------------------------------------------
List of Pre-Characterization Results for C-Intrinsic

Cell_Name             LM_View  FRAM_View  Cin_Result  PVT(index)
lanhn1                  *         *
lanhn2                  *         *
nr04d1                  *         *          from SPICE    "0"
nr04d2                  *         *
nr04d4                  *         *
or02                    *         *          from SPICE    "0"

-----------------------------------------------------------------
List of Pre-Characterization Results for Filler Cell Leakage

Cell_Name             LM_View  FRAM_View  FC_Leakage  PVT(index)
lanhn1                  *         *
```

Deleting Characterization Data

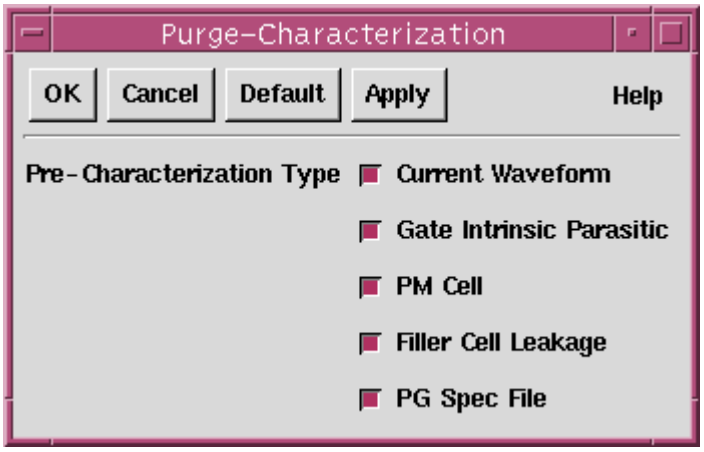
Run `pgPurgeCharacterize` to remove the characterization data from the database.

To remove the characterization data,

1. Enter `pgPurgeCharacterize` or choose Library Characterization > Purge Library Characterization.

   The Purge-Characterization dialog box appears, as shown in Figure 6-9.

Figure 6-9 The Purge-Characterization Dialog Box



2. Select the type of data to be deleted.
   - Current Waveform – Removes the current waveforms.
   - Gate Intrinsic Parasitic – Removes the intrinsic parasitic tables.
   - PM Cell – Removes the characterization data for power management cells.
   - Filler Cell Leakage – Removes the leakage current information of the filler cells.
   - PG Spec File – Removes the data listed in the power and ground specification file, loaded with the `pgLibCharacterize` command.
3. Click OK or Apply.

Multiple PVT Analysis

During the library characterization, PrimeRail is capable of choosing the correct cell information when multiple .db files with different nominal voltages exist in the logic model (LM) view. This allows you to perform power and rail analysis on several different PVT corners for a single design. Later when performing dynamic power analysis, the tool chooses the correct library characterization information based on the operating voltage when multiple PVT sets exist.

For more information about multiple PVT analysis, see “Multiple PVT Analysis” on page 12-29.

7

Performing Cell-Level Analysis

This chapter contains information about how to analyze effects on voltage and current violations of a design at the cell level. The tool supports running both cell-level time-average or dynamic analyses.

This chapter contains the following sections:

Cell-Level Time-Average and Dynamic Analysis Flows

Figure 7-1 and Figure 7-2 illustrate the steps taken in completing time-average and dynamic power and rail analyses at the cell level.

Figure 7-1 PrimeRail Cell-Level Time-Average Analysis Flow

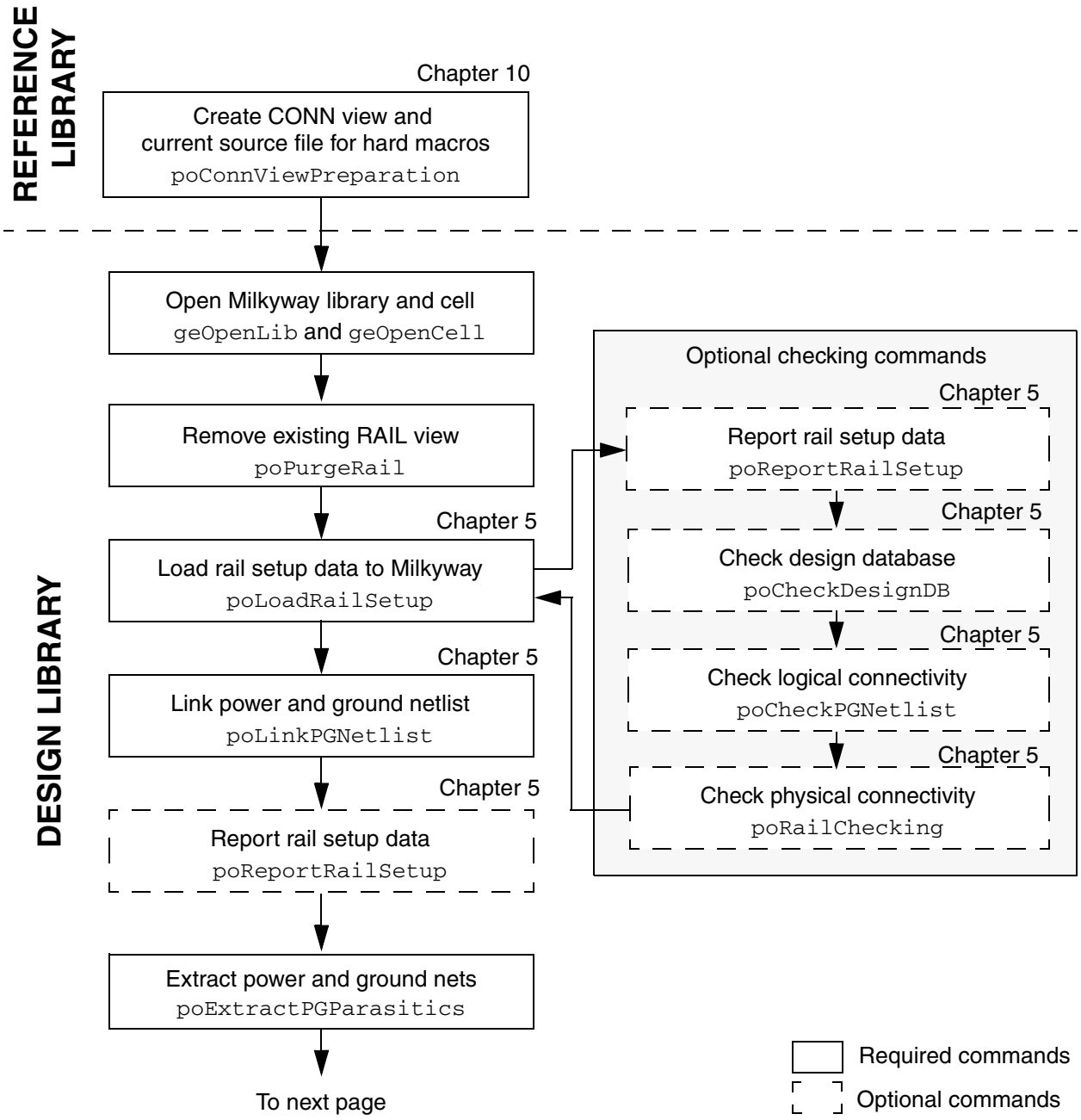


Figure 7-1 PrimeRail Cell-Level Time-Average Analysis Flow (Continued)

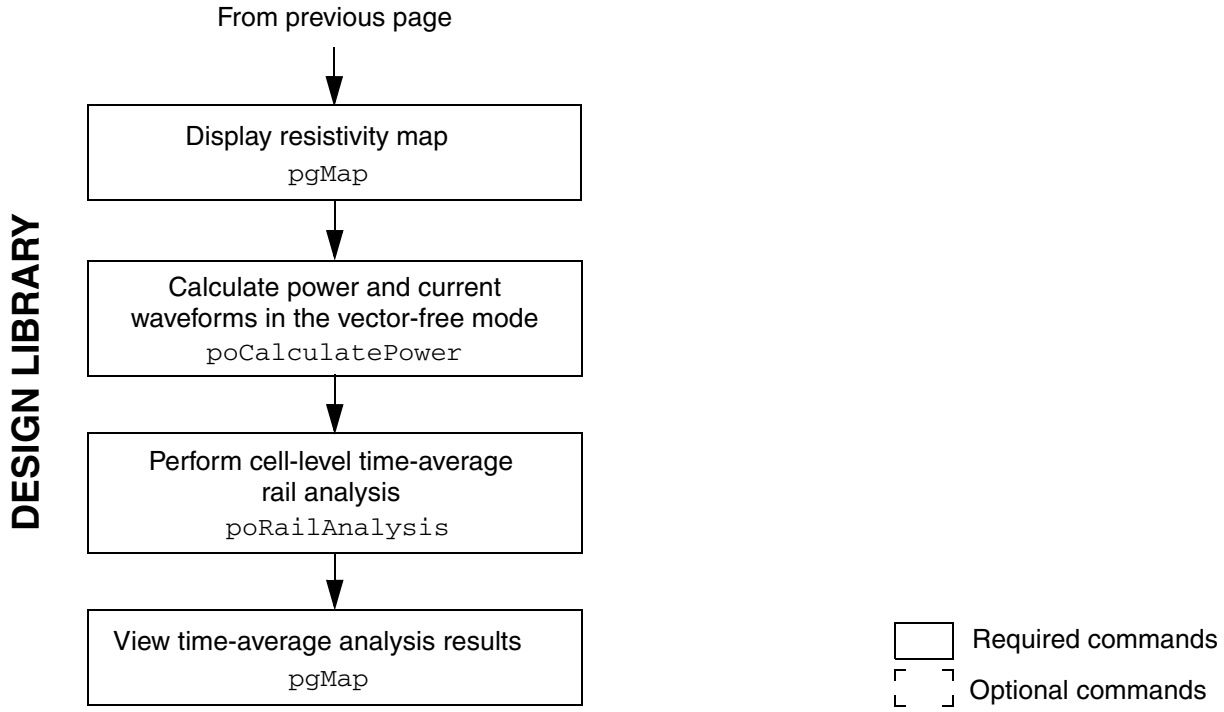


Figure 7-2 PrimeRail Cell-Level Dynamic Analysis Flow

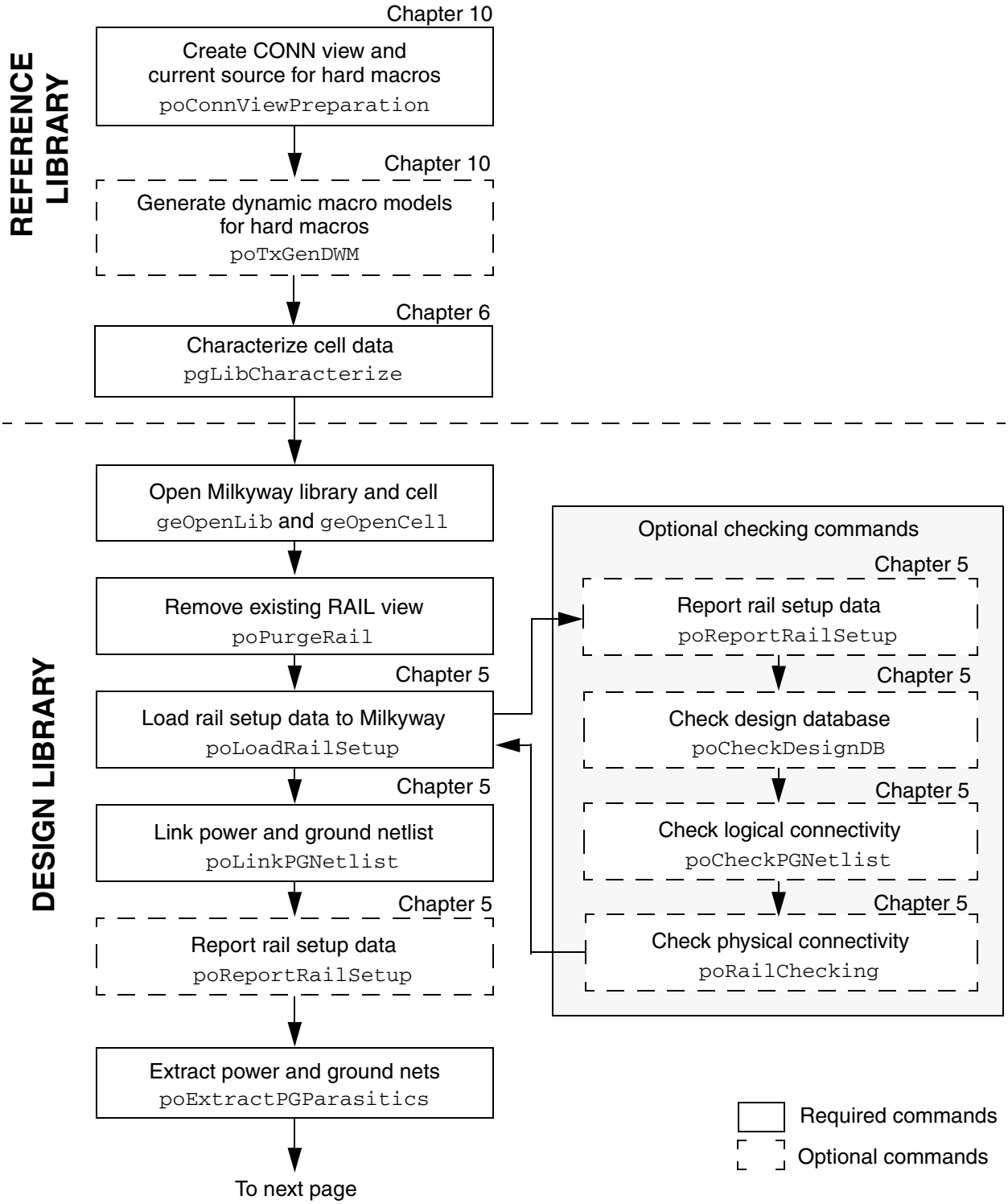
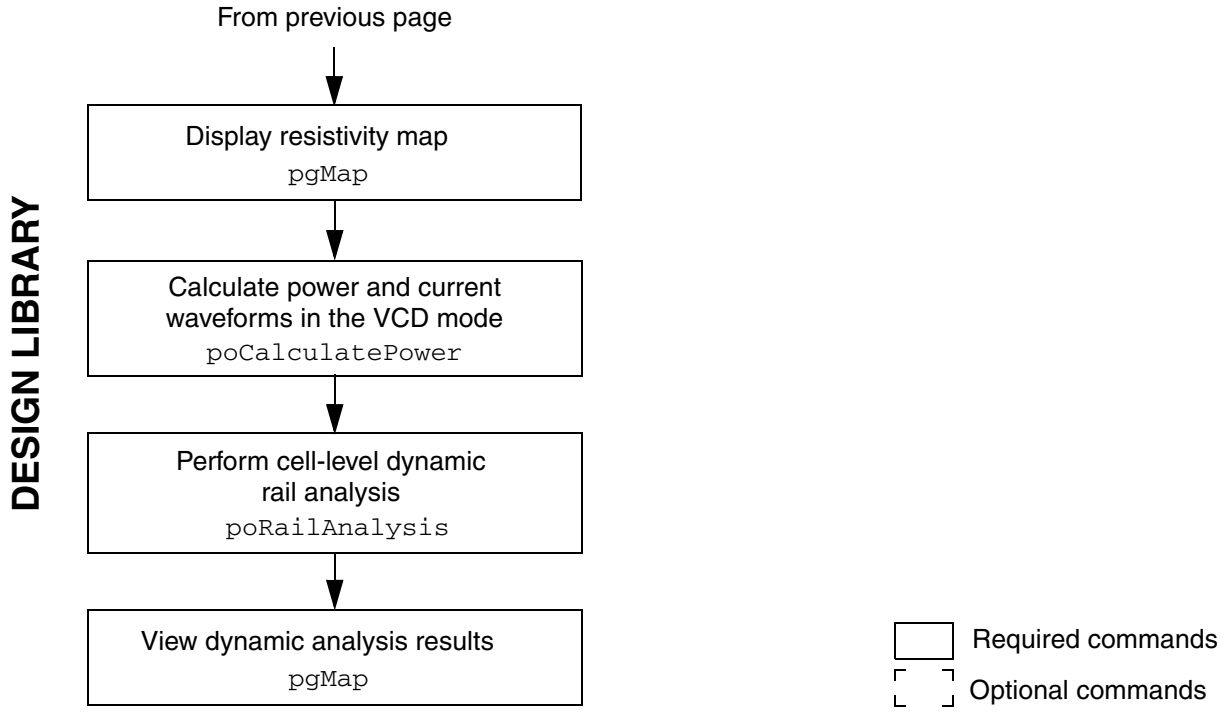


Figure 7-2 PrimeRail Cell-Level Dynamic Analysis Flow (Continued)



Preparing and Checking Input Data

Before analyzing time-average or dynamic reliability effects of the design, you need to load the necessary configuration settings and input files to the RAIL view and check the correctness and consistency of the data.

For a detailed description of preparing and checking input data, see Chapter 4, “Preparing Library and Other Input Data,” and Chapter 5, “Design Setup and Checking,” in this user guide.

Extracting Power and Ground Parasitics

PrimeRail requires the geometries and resistances of the power and ground networks in the database for performing rail analysis. The tool provides the `poExtractPGParasitics` command that extracts the RC network of the power and ground nets, automatically links the power and ground parasitics, and saves the results into the RAIL view. The `poRailAnalysis` command then uses this information saved in the RAIL view in combination with the results from the power analysis stage.

The `poExtractPGParasitics` command is able to extract the parasitics of power and ground nets inside a soft or hard macro (or at any level lower than the top level) from the top-level. This capability is useful when the design has a power management cell within a macro block and has virtual power and ground nets internal to the macro block.

When extraction is done, you can display the parasitic map to view the parasitic data saved in the RAIL view or display the resistivity map using the `pgMap` command. (See "Displaying Parasitic Map" on page 8-7 and "Displaying Resistivity Maps" on page 8-9.) This allows you to see possible locations for ideal voltage sources in your design right after extraction is complete.

This section includes the following topics:

Extracting Power and Ground Nets

When extracting only resistances for power and ground nets, PrimeRail uses the parameters from the technology file for extraction. When extracting both resistances and capacitances, PrimeRail requires the TLUPlus files that are loaded with the `poLoadRailSetup` command (see "Preparing Rail Setup Data" on page 5-2):

The following values defined in the technology file about each metal and via layer's resistivity are used to determine how much resistance and capacitance exists in the geometry of the nets that are stored in the CEL view of the database:

- contactCodeSetting – Required for resistance and capacitance extraction and resistance-only extraction
- unitCapacitanceName – Required for resistance and capacitance TLUPlus extraction
- unitMaxResistance – Required for resistance and capacitance extraction and resistance-only extraction
- unitMinResistance – Required for resistance and capacitance extraction and resistance-only extraction
- unitNomResistance – Required for resistance and capacitance extraction and resistance-only extraction

The decoupling capacitance values are also extracted.

With the information saved in the synopsys_pr_setup.e file, you can choose whether to perform

- R-only extraction, which is suitable for time-average analysis flows and takes less runtime and memory to perform
- RC extraction, which is necessary for dynamic analysis and time-average analysis with TLUPlus models

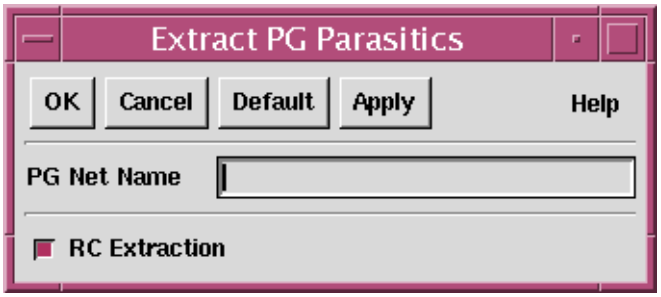
Important:
Before running the `poExtractPGParasitics` command, you must complete the runs of `poLoadRailSetup` and `poLinkPGNetlist`.

To perform power and ground net extraction,

1. Enter `poExtractPGParasitics` or choose Cell-level Analysis > PG Net Extraction – Extract PG Net Parasitics.

   The Extract PG Parasitics dialog box opens, as shown in Figure 7-3.

Figure 7-3 The Extract PG Parasitics Dialog Box



2. Specify the name of the power or ground net whose parasitic data are to be extracted. If you have multiple nets to be extracted, separate them with a white space. For example, VDD VSS.

   If you want to extract an internal power or ground net that resides in a soft macro or at any level lower than the top level, use the following syntax:

   ```
   netName instancePath.netName
   ```

   For example, if a design with a top cell “core” has main power and ground nets, VDD and GND, and an internal power and ground net VDD_int within a soft macro instance soft_instance_1, specify the power and ground net names as follows:

   ```
   VDD GND core/soft_instance1.vdd
   ```

3. Select RC Extraction if you want to extract both resistance and capacitance for the design and use the data saved in the TLUPlus file given in the synopsys_pr_setup.e file. Otherwise, the tool will extract only resistance for the design and use the data from the technology file.
4. Click OK or Apply.

Check the extraction log file and make sure block-level PARA views, that is, the reference libraries, are not used. When extraction is complete, run the `pgMap` command to display parasitic map or resistivity map. Displaying the resistivity map is to find the power and ground grid weakness before running power or rail analysis. Note that you need to first specify boundary conditions by selecting the BC Setup option in the Display Map dialog box (`pgMap`) or the source of ideal voltages if you want to display resistivity map before running power or rail analysis.

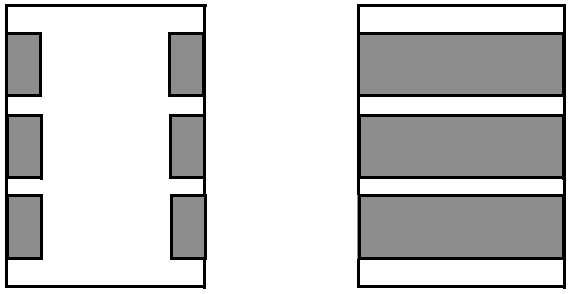
For more information about displaying a resistivity map, see "Displaying Resistivity Maps" on page 8-9.

Handling Long Pins

Using long pins in a large design allows the router to establish connections at any location along the pins. In a "flip chip" design, you can use long pins to bump pad cells while establishing connections. The pad cells can be placed at almost any location on the chip.

In PrimeRail, a pin is defined as a long pin if its length is greater than half of the cell boundary length, as shown in Figure 7-4.

Figure 7-4 Defining Long Pins



PrimeRail provides two methods for handling long pins during parasitic extraction. That which method to use is dependent on the design size and type.

- Default Long Pin Flow
- Running Long Pin Flow with Better Performance

Default Long Pin Flow

By default, PrimeRail uses the power and ground pins of a macro to route the child block with the top level in a hierarchical design. In PrimeRail, macro pins are marked as boundary nodes in the power and ground parasitic data of the child block and as port instances in the top level. PrimeRail creates one boundary node for each well-defined small rectangle pin.

For long pins, PrimeRail creates multiple boundary nodes based on the connection between the lower-level metals and the upper-level routing. The long pins themselves are promoted to the top level.

How PrimeRail Handles Nested Long Pins

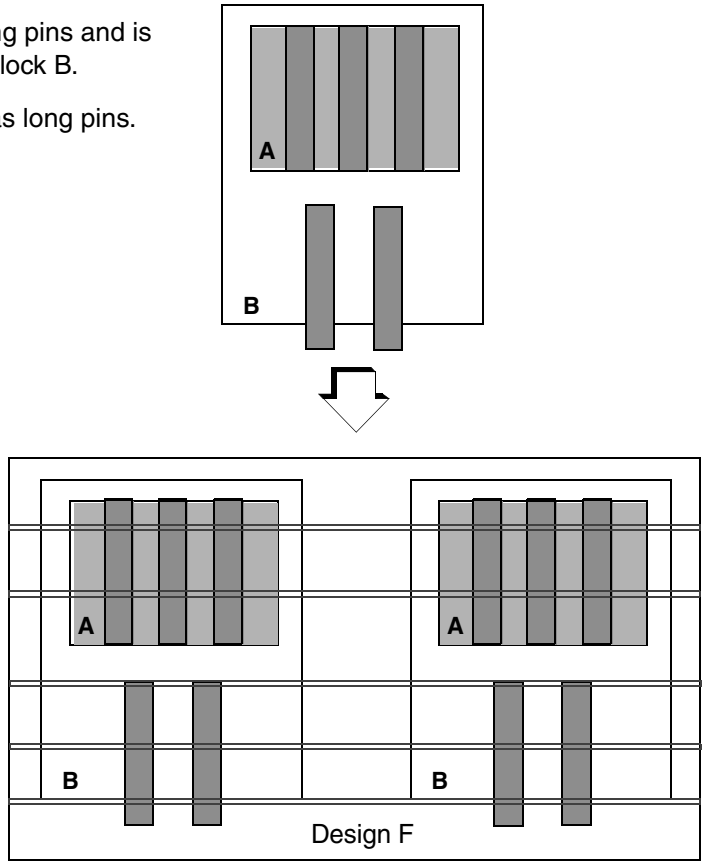
When extracting long pins, PrimeRail promotes the long pins from a subblock to their parent block, not to their grandparent block. Consequently, the connectivity in the design might not be established between the top-level routing and the grandchild block.

In Figure 7-5, Block A is placed inside Block B by instantiation as A FRAM with long pins. Block B is then instantiated at the top level inside Design F. Blocks A and B both have vertical long pins on the metal 6 layer. Design F has metal 7 horizontal routing connecting to bump the pad cells. In this situation, PrimeRail promotes the pins only one level up and thus does not recognize metal 6 long pins of the grandchild Block A connecting to the top-level design F.

Figure 7-5 Handling Nested Long Pins

Block A has long pins and is placed inside Block B.

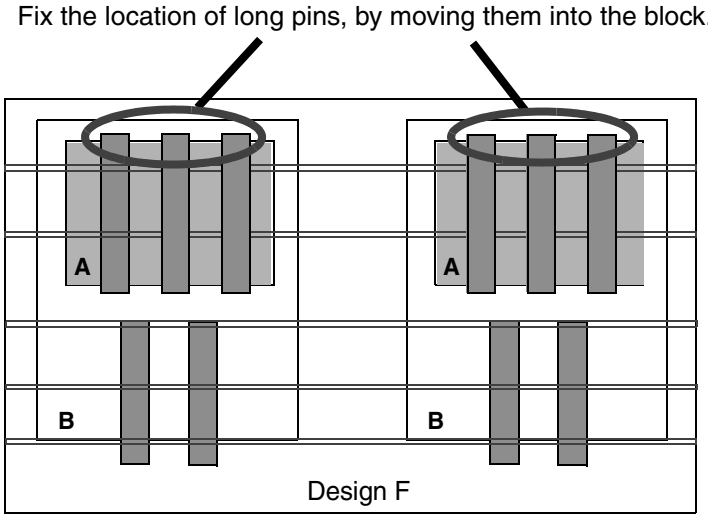
Block B also has long pins.



Showing three levels of data with long pins

The workaround is to promote the metal 6 long pins of the grandchild Block A into the child cell B manually (see Figure 7-6). In this way, the connectivity is established between the top-level metal 7 routes to all of the appropriate metal 6 rails.

Figure 7-6 Promoting Long Pins to Fix Disconnections



Running Long Pin Flow with Better Performance

In addition to the default long pin flow, PrimeRail implements another way of handling long pins during parasitic extraction by enabling the `poNewLongPinFlow` switch. This greatly improves the performance when running the parasitic extraction in terms of result accuracy, memory consumption, and runtime.

To run long pin flow with better performance, type the following in the command window before executing the `poExtractPGParasitics` command:

```
define poNewLongPinFlow 1
```

This long pin flow is off by default.

The tool saves the extraction data in the RAIL view of the Milkyway database.

Running long pin flow with the `poNewLongPinFlow` switch has the following features and capabilities:

- No boundary pins to be created when extracting all the macro cell masters (soft macro and hard macro)

When a block is extracted regardless of the macro type (hard macro or soft macro), no boundary ports will be created during the extraction. When the block is used at the next higher level of hierarchy, the tool will read the extraction data and do an instance-specific modification to include the connection to the top.

This means the tool first runs the master-based extraction on a macro cell and then performs an instance-specific post-process to modify the master-based extraction result to account for the instance-specific connectivity to the top level. PrimeRail reads top-level power and ground geometries and figures out which geometry overlap touches a macro during parasitics extraction.

- No missing geometry during block-level analysis

  When performing the block-level analysis, all the geometries are available in the block-level extraction. This is different from the default long pin flow in which some geometries will move to the higher level and cannot be seen in the block-level analysis.

- Able to apply recursively bottom up for several levels of hierarchy

  When extracting a block that has several levels of hierarchy, the tool requires a strict age relationship among the subblocks when moving from top to bottom. For example, no extraction on the child block can be newer than its parent block. If the child extraction is found to be newer than its parent during extraction, the tool will extract the parent block recursively. This is because the child extraction will be used and changed during parent extraction. If a child extraction is updated, the parent extraction also has to be updated.

- Backward compatibility

  If a block has been extracted with the default long pin flow, it will be neither reextracted nor included in the long pin with the `poNewLongPinFlow` switch overlap processing. The tool will use the RAIL view of that macro according to the default flow scheme.

  If a block has been extracted with the long pin flow with the `poNewLongPinFlow` switch enabled and you try to use its RAIL view in the long pin flow with the switch disabled, the block will be automatically reextracted even if it is a white box.

- Improved accuracy when extracting long pins at the block level

  In the default long pin flow, the tool identifies long pins based on the FRAM view pins and their length. The pins which are labeled as long pins are treated separately for extraction accuracy. The polygons belonging to long pins are propagated to the top level. Hence, these pins are not recognized for the blocks.

  The long pin flow with the `poNewLongPinFlow` switch is able to extract long pins both in the block-level and top-level analysis without losing any geometries.

- Power management cells supported

  The long pin flow supports the usage of power management cells that are imported by the `poPGImportModel` command.

- When running extraction on the macro, PrimeRail has the following features:

  PrimeRail reads the data stored in the RAIL view and reconstructs geometries for layers with top-level overlap. It will analyze the interaction of top-level and macro-level geometries if any of the following cases occurs:

  - The via connection explodes the block-level wire to the top.
  - The partial overlap (t-junction or jog) explodes the block-level wire to the top.
  - The straight overlap and top wire contained in the block wire remove the top segment of wire.

  Additionally, according to exploded wires PrimeRail creates new boundary pins in the RAIL view according to the removed segments and creates instance-specific RAIL views for macros by removing parasitics. The tool saves data to the new resistance-inductance-capacitance (RLC) net inside the RAIL view of the parent master cell.

- When running the top-level extraction, PrimeRail does the following:

  - Removes top-level geometries marked as overlapped with block-level geometries
  - Adds geometries exploded from the block level
  - Creates current sources based on the boundary ports created in the instance-specific RAIL views

Note:

If you enable the `poNewLongPinFlow` switch to extract FPGA designs or other design styles using a structure with abutting tiles and no top-level routing, the extraction result is not expected to be good.

Using gray boxes in the long pin flow with the `poNewLongPinFlow` switch is not supported.

Handling a Long Via Array

Use the `poViaCutRowColumn` switch to break a large square or a rectangular via array into smaller square via arrays, based on the value you specify.

To use the `poViaCutRowColumn` switch, enter the following syntax in the command window:

```
define poViaCutRowColumn value
```

where *value* can be any integer.

For example, if you define the size as 5 and the design has one 1 x 10 via array, one 10 x 1 via array, and one 9 x 9 via array, the cutting results are as follows:

Original via layer	After cutting
1 x 10 via array	Two 1 x 5 via arrays
10 x 1 via array	Two 5 x 1 via arrays
9 x 9 via array	One 5 x 5 via array, one 4 x 5 array, one 5 x 4 via array, and one 4 x 4 array

Characterizing Cell Data

In a cell-level dynamic analysis flow, PrimeRail requires the characterization of the gate-level cells in addition to the established library characterization data. The run of library characterization is not required in the time-average flow.

You must complete library characterization before proceeding to power or rail analysis in a dynamic analysis flow.

For more information, see Chapter 6, “Library Characterization for Cell-Level Dynamic Analysis,” in this user guide.

Calculating Power and Current Waveforms

When the rail setup information from IC Compiler is saved to the database, and extraction data (and characterization results if in a dynamic flow) are available, run the `poCalculatePower` command to calculate power and current waveforms of the design which is being analyzed. (See “Preparing Rail Setup Data” on page 5-2.)

PrimeRail adopts the PrimeTime PX technology to generate the necessary power and timing information for calculating current waveforms of the design. When executing the `poCalculatePower` command, a limited version of PrimeTime PX is invoked within PrimeRail to generate necessary power and timing information without requiring the PrimeTime license key. When PrimeTime PX run is complete, the command uses the output PrimeTime timing and power reports, either in the vector-free or vector-based mode, to calculate current waveforms of the design.

Note:

The `poCalculatePower` command enables you to run PrimeTime PX gate-level power analysis and to generate current waveforms of the design in one graphic user interface. Thus, it is not necessary to run multiple commands, such as `poCreatePTPXScript`, `poCallPTPX`, and `poTransientPowerAnalysis`, to complete the process.

For information on running each individual command to complete the process, see the online Help for details about running the commands.

PrimeTime PX Gate-Level Power Analysis

Using the rail setup information from IC Compiler, the `poCalculatePower` command first utilizes the PrimeTime PX timing sign-off technology to analyze timing and power values of the design. PrimeTime PX is the power analysis extension to PrimeTime which enables you to perform both timing and power analysis within a single, easy-to-use environment.

The PrimeRail `poCalculatePower` command automatically searches the PATH environment variable for the pt_shell program. If the binary does not exist in your PATH variable, you must provide the complete path to pt_shell.

This section describes only the basics of running PrimeTime PX for gate-level power analysis. For more in-depth information about PrimeTime PX, see the PrimeTime PX documentation.

Note:

You must have PrimeTime PX version B-2008.06-SP3 or later.

If your rail setup data is not from IC Compiler, follow the steps listed in the “Differences When Using IC Compiler Input Data” on page A-2 to analyze power and timing effects of the design.

If a PrimeTime PX script already exists, PrimeRail removes unsupported PrimeTime PX commands from the script file and renames the script file with an affix, *_pr*. If any of the required files, such as .db files, Verilog files, SDC files, parasitics, and SAIF files, are missing, PrimeRail stops executing the command.

The following commands are supported when you invoke PrimeTime PX within the PrimeRail environment:

- File-reading commands: `read_verilog`, `read_sdc`, `read_parasitics`, `read_vcd`, `read_saif`, and `read_sdf`
- Design or environment-setting commands: all SDC commands, all collection commands, and all setting commands (`set_switching_activities`, `set_current_rail`, and so on)
- Power report command: `report_power_rail_mapping`

Using CCS Power Data

Both PrimeTime PX and PrimeRail support CCS power data. In PrimeTime PX, you can use the `power_model_preference` switch to instruct PrimeTime PX which power data is preferred over the other. Then if both types of data are available in a library, PrimeTime PX will use the one you prefer. If only one type of data is available, PrimeTime PX uses that data type.

For more information about using CCS power modeling technology in PrimeTime PX, see the PrimeTime PX documentation.

When a CCS power .db file is present, PrimeRail automatically reads the data from the CCS power .db file. (See "CCS Power Model Support" on page 4-7.) Note that when CCS power data is used, no library characterization is required for performing gate-level power analysis with PrimeTime PX. However, if your library is in NLPM format, you need to characterize the library data before performing gate-level power analysis.

Calculating Current Waveforms

When the PrimeTime PX run is complete, the `poCalculatePower` command uses the PrimeTime PX timing and power reports to generate current waveforms of the design and loads additional parasitics of the power supply network for all the power and ground ports of each cell in the design. The generated waveforms, parasitics, and timing-related data are saved in the Milkyway RAIL view, from which they can be referenced during rail analysis.

PrimeRail enables you to generate current waveforms in any of the following analysis modes:

- Vector-Free Mode

  No vector file is required for running the time-average analysis. In the vector-free mode, the current waveforms determined by PrimeRail are representative of the circuit, such that its time-average properties are preserved. Currently, running the time-average analysis cannot predict the absolute worst-case behavior of the design.

  You can also specify what information to use for current calculation. The information can be from the current waveform, the cell instance power, or the intrinsic parasitics of a cell. If this is the case, the information computed by the `poCalculatePower` command will be overridden by the information you specify.

- Vector-Based Mode

  Running a vector-based analysis requires a VCD file that contains detailed switching event information. Therefore, running a vector-based analysis generates the most accurate dynamic current waveforms for the design. The vector-based analysis is also called event-based analysis.

Like in the vector-free mode, you can specify what information to use during current calculation. The information can be from the current waveform, the cell instance power, or the intrinsic parasitics of a cell. If this is the case, the information computed by the `poCalculatePower` command is overridden by the information you specify. Use this capability when the design contains cells for which library characterization cannot be done, for example, where no accurate SPICE description is available.

Vector-Free Mode

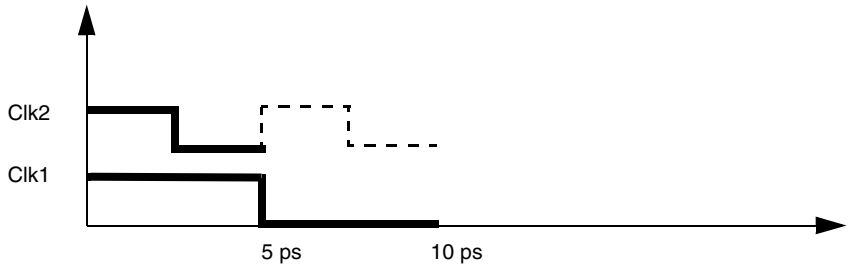
In the vector-free mode, PrimeRail generates the current waveforms using the signal probabilities and timing windows that are determined by PrimeTime PX. The PrimeTime PX power analysis provides timing windows for each switching event of all the cells for a full clock period. Rising and falling switching events are considered separately.

The current waveform generated for a switching event also has a dependency on the state of the inputs of the cell switching. In the time-average analysis, PrimeRail accounts for this dependency by generating a composite current waveform. This current waveform is a superposition of current waveforms for all input states of the cell that are scaled by the probability of a state according to the signal probabilities of the inputs of the cell.

Handling Multiple Clocks

If a design has multiple clocks switching at different frequencies, PrimeRail repeats the smaller clock period up to the maximum clock period during time-average analysis. This ensures that clocks with smaller periods will be simulated up to the whole simulation time.

In the following design, for example, clock clk1 has a larger clock period, whereas clock clk2 has a smaller clock period. In this case, PrimeRail repeats the clock period of the clock clk2 until it reaches the period of clock clk1. (See the dotted lines in the design.)

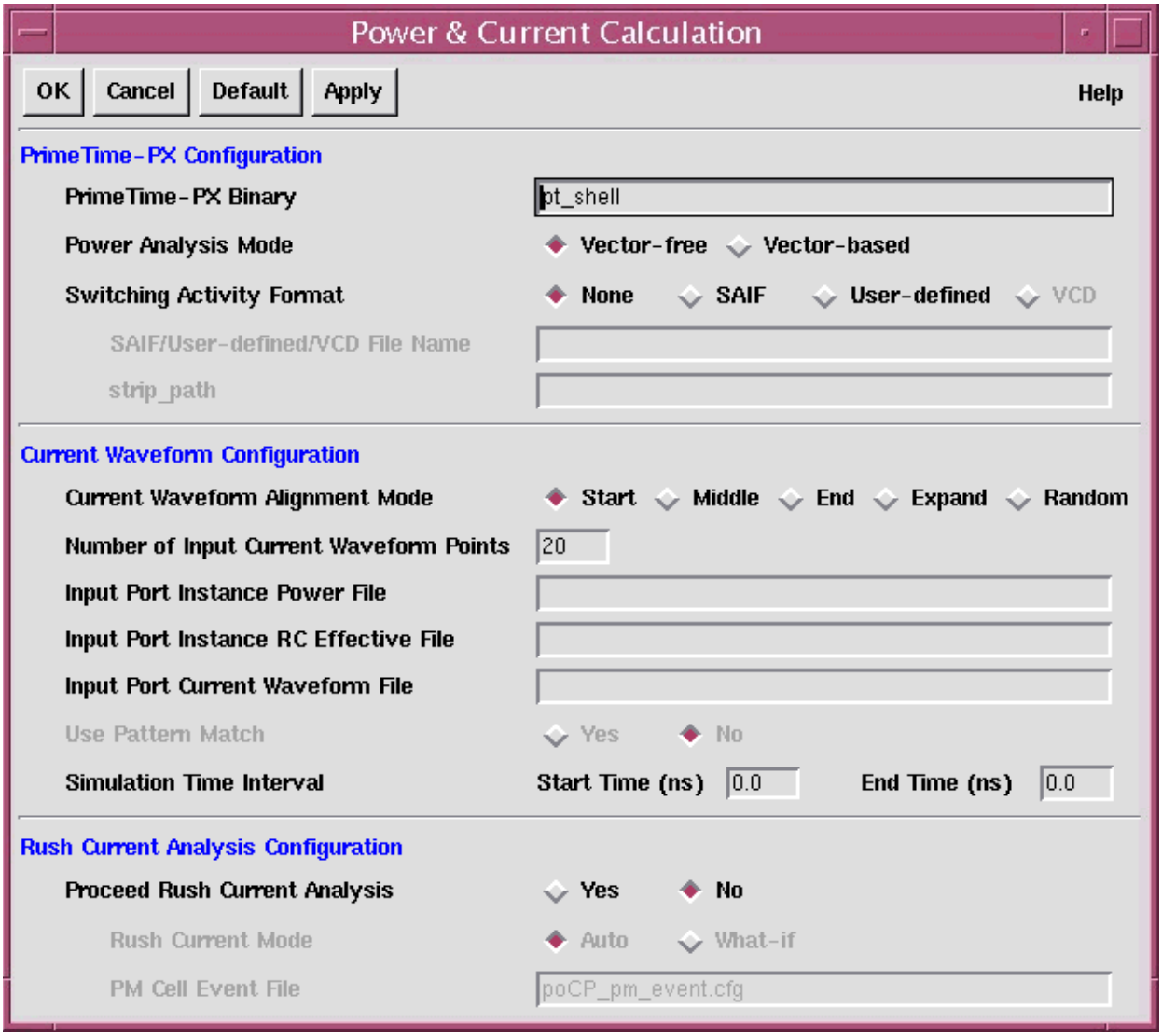


To calculate power and current waveforms without vector files,

1. Enter `poCalculatePower` or choose Cell-level Analysis > Power Analysis – Calculate Power.

   The Power & Current Calculation dialog box opens.

Figure 7-7 Power & Current Calculation Dialog Box



2. In the PrimeTime-PX Configuration section, specify options for the PrimeTime PX run.
   - PrimeTime-PX Binary – Enter the path to the PrimeTime PX binary. The default is pt_shell.
   - Power Analysis Mode – Choose Vector-free as the power analysis mode to analyze current waveforms of the design.
   - Switching Activity Format – Select None because you do not need switching activity information to run a time-average analysis.

When no switching activity information is available, PrimeTime PX uses the default switching activity for its primary inputs and black box outputs. Starting from these known points, PrimeTime PX runs switching activity propagation by running zero-delay simulation to find out the rest of the unknown switching activities. If only partial switching activity information is given, the zero-delay simulation is used to derive the switching activity for the nets without given switching activity.

3. In the Current Waveform Configuration section, specify options for generating current waveforms.

   - Current Waveform Alignment Mode – Determine the alignment of the computed current waveform relative to the timing window of the event. Select this option only when you are performing a time-average analysis.

     Start (the default) – Align the current waveforms with the starting point of the time window.

     Middle – Align the current waveforms with the middle point of the time window.

     End – Align the current waveforms with the ending point of the time window.

     Expand – Scale the current waveform such that the total energy of the event is conserved but the current waveform completely covers the time window.

     Random – Pick a random starting point for the current waveform from the time interval given by the time window.

   - Number of Input Current Waveform Points – Determine the number of points used in a piecewise linear representation to describe the current waveform. Each point is represented by (t, i).

   - Simulation Time Interval – Enter the start and end time for the simulation.

4. Enter the name of the input port instance power file that defines the power values of cell instances in the design. Use the port instance power file for the hard macro without library power models where the accurate power value is unavailable. This setting is optional.

   The syntax of the file is

   ```
   defineCellInstancePower cellId cellInstName value
   defineCellInstancePowerByPort cellId cellInstName portName value
   ```

   The value of the variable is in the unit defined in the header section of the technology file. The power values defined in the port instance power file overwrite those described in the PrimeTime PX report file. The use of wildcard or pattern matching is supported.

   For a detailed description about using these Scheme commands, see the online Help for the commands or Appendix C, “Commands for Specifying Hard Macro Power,” in this user guide.

5. Enter the name of the input port instance effective file if you want to set user-specified effective resistance and capacitance values, by using the following two Scheme commands:

```
definePortInstanceRCEffective cellId cellInstName portName \
        R_value C_value
definePortInstanceRCEffectiveByMaster cellId cellInstMasterName \
        portName R_value C_value
```

   The value variable is in the unit defined in the header section of the technology file.

   For a detailed description about using these Scheme commands, see the online Help for the commands.

6. Enter the name of input current waveform file if you want to assign user-specified current waveforms and to describe the current waveforms in the piecewise linear format, by using the following Scheme command:

```
definePortInstanceCurrentWaveform cellId cellInstName portName \
       numPoints'(t1 i1 t2 i2...)
```

7. In the Use Pattern Match text box, select Yes if any of the above files involve pattern matching when applying the user-defined settings. However, because using names involving pattern matching can degrade runtime, you have to explicitly enable pattern matching if it is to be used by the configuration file syntax.

   Do not use Pattern Match if no pattern is used in cell instance and port instance names. Otherwise it will increase runtime and memory usage.

8. Click OK or Apply.

Vector-Based Mode

With the detailed information about switching events in the VCD file, the timing analysis results generated by the PrimeTime engine and the Liberty power models, PrimeTime PX can derive the detailed switching, short-circuit, and leakage power consumption values for every cell instance and for each event in the given simulation vector.

When the library characterization is complete and the PrimeTime PX reports are available, do the following steps:

1. Enter `poCalculatePower` or choose Cell-level Analysis > Power Analysis – Calculate Power to open the Power & Current Calculation dialog box (see Figure 7-7 on page 7-17).
2. In the PrimeTime-PX Configuration section,
   - PrimeTime PX Binary – Enter the path to access the PrimeTime PX binary.

- Power Analysis Mode – Select Vector-based as the power analysis mode model to analyze dynamic current waveforms of the design.
- Switching Activity Format – Select SAIF or VCD, based on the format of the input switching activity information. If you have a user-defined switching power file and want to use it for current waveform generation, select User-defined.

  SAIF/User-defined/VCD File Name – Enter the name of the input file that contains the switching activity information.

  The following variables and commands are supported by the user-defined switching information file:

  Variables:

  ```
  power_default_toggl_rate (the default is 0.1)
  power_default_static_probability (the default is 0.5)
  power_default_toggle_rate_reference_clock [fastest|related]

          (the default is related)
  ```

  Commands:

  ```
  set_case_analysis 0 reset
  set_switching_activity
  set_annotated_power
  ```

  For more information about these variables and commands, see the PrimeTime PX documentation.

  strip_path – Enter the path of the instance to be removed.

3. Determine the number of points used in a piecewise linear representation to represent a single switching event. Each point is represented by (t, i). The number can be any integer from 3 to 6.

   Note that because the current waveform lookup tables created during library characterization represent a current waveform by 6 points (5 points plus the peak value), it does not make sense to set a value higher than 6 in the vector-based analysis.

4. Specify the start and end time for the simulation.

5. Enter the name of the input port instance power file that defines the power values of cell instances in the design. Use the port instance power file for the hard macro without library power models where the accurate power value is unavailable. This setting is optional.

   The syntax of the file is

   ```
   defineCellInstancePower cellId cellInstName value
   defineCellInstancePowerByPort cellId cellInstName portName value
   ```

The value of the variable is in the unit defined in the header section of the technology file. The power values defined in the port instance power file overwrite those described in the PrimeTime PX report file. The use of wildcard or pattern matching is supported.

For a detailed description about using these Scheme commands, see the online Help for the commands or Appendix C, "Commands for Specifying Hard Macro Power," in this user guide.

6. Enter the name of the input port instance effective file if you want to set user-specified effective resistance and capacitance values, by using the following two Scheme commands:

```
definePortInstanceRCEffective cellId cellInstName portName \
        R_value C_value
definePortInstanceRCEffectiveByMaster cellId cellInstMasterName \
        portName R_value C_value
```

The value variable is in the unit defined in the header section of the technology file.

For a detailed description about using these Scheme commands, see the online Help for the commands.

7. Enter the name of input current waveform file if you want to assign user-specified current waveforms and to describe the current waveforms in the piecewise linear format, by using the following Scheme command:

```
definePortInstanceCurrentWaveform cellId cellInstName portName \
       numPoints'(t1 i1 t2 i2...)
```

8. In the Use Pattern Match text box, select Yes if any of the above files involve pattern matching when applying the user-defined settings. However, because using names involving pattern matching can degrade runtime, you have to explicitly enable pattern matching if it is to be used by the configuration file syntax.

   Do not use Pattern Match if no pattern is used in cell instance and port instance names. Otherwise it will increase runtime and memory usage.

9. Click OK or Apply.

Saving Current Waveforms to FSDB File

Use the `poDumpPowerDBToFsdb` command to write the current waveforms generated by `poCalculatePower` to an output Fast Signal Database (FSDB) file. This FSDB current waveform file contains the information for each power and ground port instance. You can open the FSDB file and view the current waveforms in a standard waveform viewer, like nWave.

To save the calculated current waveforms to the FSDB file, you first need to prepare a configuration file that contains the necessary settings by using the following syntax:

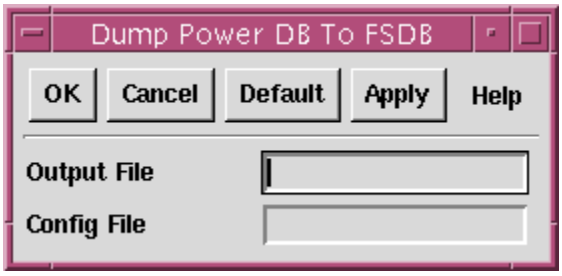
```
Start_time ;(in seconds if not specified)
End_time ;(in seconds if not specified)
Instance_name.PGportname ;(All are written to the output file if not
  specified. Pattern-matching is supported.)
```

To write current waveforms to an output FSDB file,

1. Enter `poDumpPowerDBToFsdb` or choose Diagnosis > Report – Dump Current Waveforms.

   The Dump Power DB To FSDB dialog box appears, as shown in Figure 7-8.

Figure 7-8 The Dump Power DB To FSDB Dialog Box



2. Enter the name of the output file to be generated.
3. Enter the name of the configuration file that contains the necessary settings.
4. Click OK or Apply.

Viewing Power Analysis Results

When power analysis is complete, you can check for power distributions and current waveforms in your design. You can generate a report and display the power map through the `pgMap` command.

For more information, see Chapter 8, “Viewing Cell-Level Analysis Results,” in this user guide.

Cell-Level Rail Analysis

After current waveform generation has been completed for all cells in the design, use the `poRailAnalysis` command to perform time-average or dynamic rail analysis at the cell level.

The following sections show you how to use the `poRailAnalysis` command, including specifying the source of ideal voltage values, describing the choices possible in running rail analysis, and explaining what sort of results are provided by this command.

Ideal Voltage Source Locations for Rail Analysis

During rail analysis, PrimeRail combines the power consumption results and the resistive network results to solve for the voltage drop values at each node in the resistive network. To acquire more accurate results, PrimeRail needs the location of the ideal voltage source and the ideal power supply in the design.

The choices available for specifying ideal voltage sources in the P/G Rail Analysis dialog box (see Figure 7-10 on page 7-28) are as follows:

- Top-Level Pad Cell Master Names
- Top-Level Pad Cell Instance Names
- Top-Level Design Pin Objects
- User-Defined Taps

Top-Level Pad Cell Master Names

When the design that is open to be analyzed is the top level of a chip and the pad cells are placed and connected to the power and ground network, you can specify the name of the pad cell master.

To do this, simply place the name of the pad cell master in an ASCII file. You can place multiple cell master names in the file, if necessary. For example, if the VDD net in a top-level chip design is powered by the pad cells VDD0 and VDD1 (perhaps with different geometries or capabilities), you can create a file that contains the following lines:

```
VDD0
VDD1
```

Next, run the `poRailAnalysis` command, by using the pad master file. In the P/G Pad Info box of the P/G Rail Analysis dialog box (see Figure 7-10 on page 7-28), select the Top-Level Design Pad option and then Master. Enter the name of the pad master file in the Pad Name File text field.

Top-Level Pad Cell Instance Names

When your design has pad cells hooked up to the network of choice but you want to set only some of them to be the source of ideal voltage, specify the names of the instances in an ASCII file, in the following syntax:

```
<cellName> <netName> [<resValue>] [<indValue>] [<capValue>] [<delta-V>]
```

where each variable requires a specific format, as described below:

```
<cellName> := padMasterName | padInstanceName
    <padInstanceName> := instanceName | "hierarchicalInstanceName"
<netName>  := net name
<resValue> := <valueSet> | R(<valueSet>)
<valueSet> := nomValue | nomValue minValue maxValue
<delta-V>  := V(<valueSet>)
```

The `<instanceName>` variable represents the simple name for the instance. The `<hierarchicalInstanceName>` variable describes the hierarchy information (separated with a space and grouped in double quotes).

If the specified net name does not match with the one specified with `poRailAnalysis`, the tool will not use the pad specified.

PrimeRail also supports the IC Compiler power network analysis (PNA) syntax. However, the PNA syntax where the net name follows the cell name is mutually exclusive with the PrimeRail native syntax where the pad can be followed by an optional package parasitic specification. If a file contains mixed PNA and PrimeRail formats, the tool will issue an error message RAIL-403.

Example:

```
"TOP_CELL MACRO_CELL1 PAD_BUMP_01"
```

In this example, TOP_CELL is the name of the top-level design where MACRO_CELL1 represents a soft macro. PAD_BUMP_01 is the pad cell instance, which is placed in the lower-level block MACRO_CELL1.

```
pad_ring_inst/VDD0_inst1
pad_ring_inst/VDD0_inst2
pad_ring_inst/VDD1_inst1
```

Next, run the `poRailAnalysis` command, by using the pad instance file. In the P/G Pad Info box of the P/G Rail Analysis dialog box (see Figure 7-10 on page 7-28), select Top-Level Design Pad and then Instance. Enter the name of the pad instance file in the Pad Name File box.

Top-Level Design Pin Objects

If you are putting a soft macro through the rail analysis flow, there are probably no pad cells in place. However, the FRAM view of this soft macro needs to have its power and ground pins defined, so it can be placed in its parent cell. In this case, you can take these pin objects from the FRAM view as the source of ideal voltage.

When the top-level design pins are connected to voltage sources through external resistances, you must specify a pin resistor file to be used. This field is optional. The format of the file is as follows:

```
pinName nomResValue [minResValue maxResValue]
```

When you enter only one resistance value for a given pin, this resistance is always used. When you enter three values (nom, min, max), the tool picks one for rail analysis, based on the corners of the parasitic.

Next, run the `poRailAnalysis` command, by using the pin resistor file. In the P/G Pad Info box of the P/G Rail Analysis dialog box (see Figure 7-10 on page 7-28), select Top-Level Design Pin. Enter the name of the pin resistor file in the Pin Resistor File box.

Note:

When the long pin flow is applied during parasitics extraction (see “Running Long Pin Flow with Better Performance” on page 7-10), PrimeRail will issue an error message if the length of the pins are greater than 20 microns. To avoid this error from happening, you can create a tap file by using the `poGenUserDefinedTapByPin` command. You may refer to the RAIL-400 error message for further details.

User-Defined Taps

If you are working with a block or a design that does not yet have pins defined, or perhaps the location or design of the pad cells is not finalized, you can create an ASCII file that specifies where the ideal voltage is applied by coordinates and layer numbers with the following format:

```
netName  layerNumber  xCoord  yCoord
```

The coordinates are in microns. For instance, if you want to apply the ideal voltage of the VDD net on layer 3 at three locations, create the file with the following lines:

```
VDD 3 784.000  1826.000
VDD 3 1184.000 1826.000
VDD 3 1584.000 1826.000
```

PrimeRail supports the IC Compiler power network analysis (PNA) syntax. The tool accepts taps in the PNA format that have a defined layer and a resistor on the top of the power and ground geometry. If your user-defined tap file contains information in both PNA and PrimeRail formats, the tool will issue an error message RAIL-403.

The PrimeRail `poGenUserDefineTap` command enables you to insert taps to a power or ground net by directly selecting a point in the design layout. The tool automatically writes the location of the taps you insert to a tap file.

When the generation is complete, run the `poRailAnalysis` command by using the tap file. In the P/G Pad Info box of the P/G Rail Analysis dialog box (see Figure 7-10 on page 7-28), select User-Defined Taps. Enter the name of the tap file in the Tap File box.

The following section describes the procedure for generating the tap file automatically.

Generating User-Defined Tap File

Use the `poGenUserDefineTap` command to automatically generate a user-defined tap file that is used to provide ideal voltage sources during rail analysis.

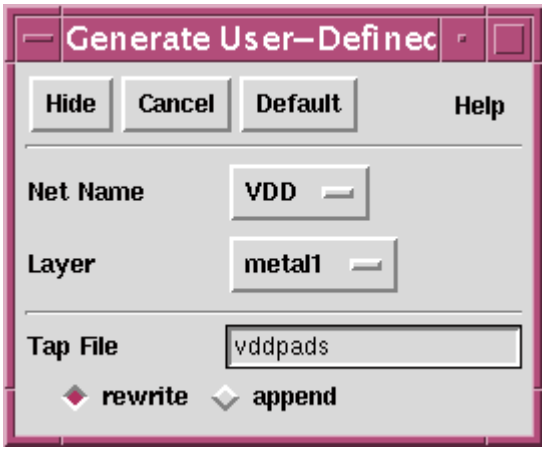
Note:
Be sure to open the design library and the cell to be analyzed before running the `poGenUserDefineTap` command.

To generate a user-defined tap file,

1. In the design layout window, zoom in to the location and click the point where you want to add a tap.
2. Enter `poGenUserDefineTap` in the command window.

   The Generate User-Defined Taps dialog box appears, as shown in Figure 7-9.

Figure 7-9 The Generate User-Defined Taps Dialog Box



3. Choose the name of the power and ground net for which the tap is to be inserted. The default is VDD.
4. Choose the name of the layer for which the tap is to be inserted. The default is metal1.

5. Enter the name of the tap file to be generated or updated. The default tap file name is vddpads.
6. If you have a tap file and you want to update it by inserting additional taps, select “rewrite” (the default) if you want to write over what is written in the tap file. Select “append” if you want to update the new tap location without overwriting what is written in the tap file.
7. Move back to the cell editing window, and click a point where the tap is to be inserted. The tap location is then written to the command window and the tap file is specified in the Generate User-Defined Tap dialog box, in the following syntax:

```
Writing tap (netName  layer_number  point_x  point_y)
into file "fileName".
```

8. Keep clicking the points where you want to insert taps. When finished, click Hide to close the Generate User-Defined Tap dialog box. The generated tap file can be found in the directory in which PrimeRail is installed.

Analyzing IR Drop and Current Density Violations

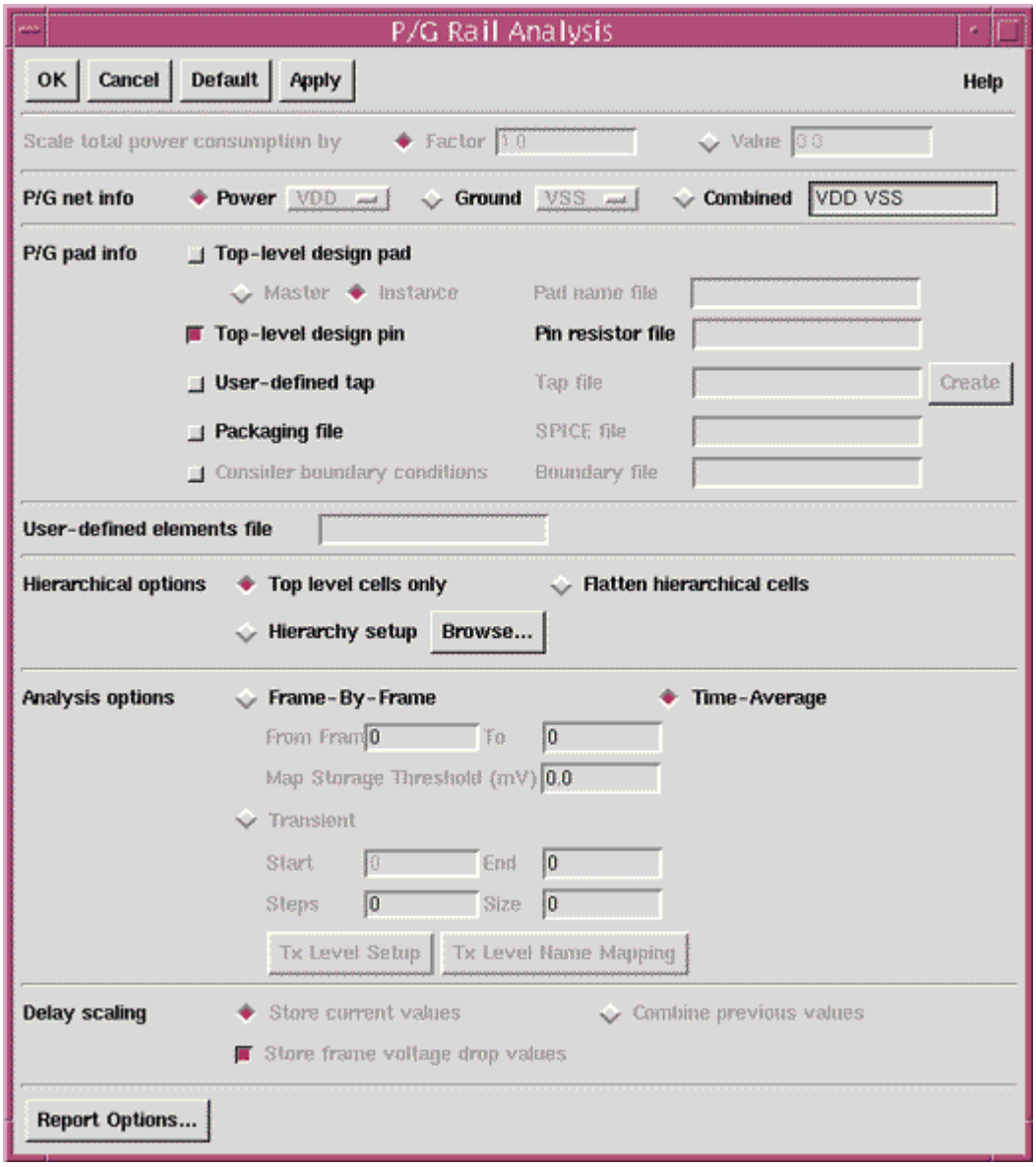
During rail analysis, PrimeRail analyzes the voltage drop and electromigration violations of the specified power and ground nets by using hierarchical techniques. The processes of power analysis and power and ground net extraction must be completed before you can perform the rail analysis. Otherwise PrimeRail stops executing the command and alerts you with a message in the command window.

To perform voltage drop and electromigration analysis,

1. Enter `poRailAnalysis` or choose Cell-level Analysis > Rail Analysis – Rail Analysis.

   The P/G Rail Analysis dialog box appears.

Figure 7-10 The P/G Rail Analysis Dialog Box



2. In the “P/G net info” section, select the names of the power and ground net you want to analyze. The default net names are VDD and VSS.
3. In the “P/G pad info” section, specify the locations of the ideal voltage sources by selecting one or more of the following options:
   - Top-level design pad cell master names
   - Top-level design pad cell instance names

- Top-level design pin objects (the default)
- User-defined taps
- Packaging file

  This option is available only for the dynamic analysis flow.

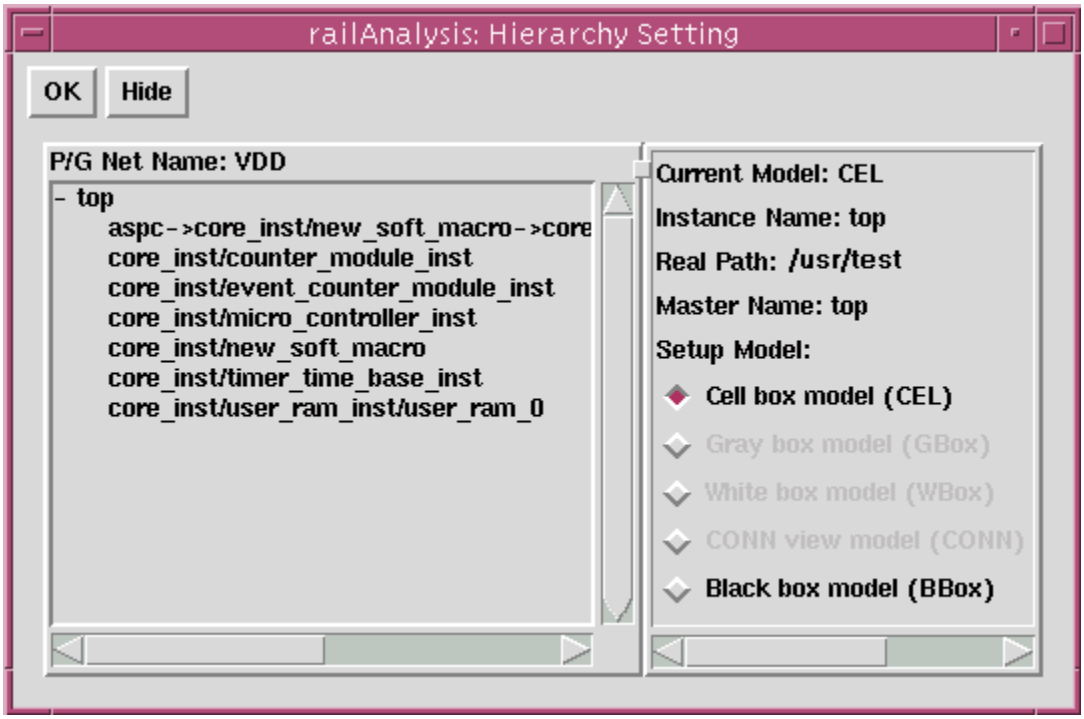
  The package file includes general linear packaging models during full-chip rail analysis. The tool supports package circuits with mutual inductors (K), dependent sources, like voltage-controlled voltage sources (E), current-controlled current sources (F), voltage-controlled current sources (G), current-controlled voltage sources (H) and sub-circuit instances (X) for packaging in the rail analysis. For more information, see “General Linear Packaging Models” on page 12-7 in this user guide.

  For a detailed description about specifying ideal voltage sources, see “Ideal Voltage Source Locations for Rail Analysis” on page 7-23.

4. In the “User-defined elements file” section, enter the name of the elements file that contains the information to be considered during the dynamic rail analysis. This option is available only in the dynamic analysis flow.
5. In the “Hierarchical options” section, specify whether to analyze only the top-level routing or to flatten hierarchical cells and descend into each soft macro in the design.

   In addition, you can specify the type of macro model for a block in the design, by traversing through different levels of hierarchy in the design. To do this, select “Hierarchy setup.” If you want to change the default setting, click Browse to open the railAnalysis: Hierarchy Setting dialog box, as shown in Figure 7-11 on page 7-30.

Figure 7-11 The railAnalysis: Hierarchy Setting Dialog Box



In the railAnalysis: Hierarchy Setting dialog box, click the name of the top cell in the list to display the details of the cell, including its current model setting, instance name, location, master name, and setup model.

Double-click the name of the top cell to show its cell instances in different levels of hierarchy. Click to select an instance for viewing and specifying the type of macro model to be used.

Click OK to apply the changes you've made.

6. In the "Analysis options" section, select one of the following options to perform rail analysis:

   - Time-Average – Select this option to perform a time-average (static) rail analysis at the cell level. Use this option only when you are running the time-average analysis flow.
   - Transient – Select this option to perform a dynamic rail analysis at the cell or transistor level. Use this option only when you are running the dynamic analysis flow.

     Start / End – The start and end time that defines the time segment. This time segment should be a subset of the current waveform generation time window. Specifying start and end time allows you to run a simulation on a specific subwindow of peak activity, instead of on the entire simulation time.

     Steps – The number of time steps the simulation is run.

Size – The size of the time step used during simulation. The default is one-third of the average transition time in the circuit.

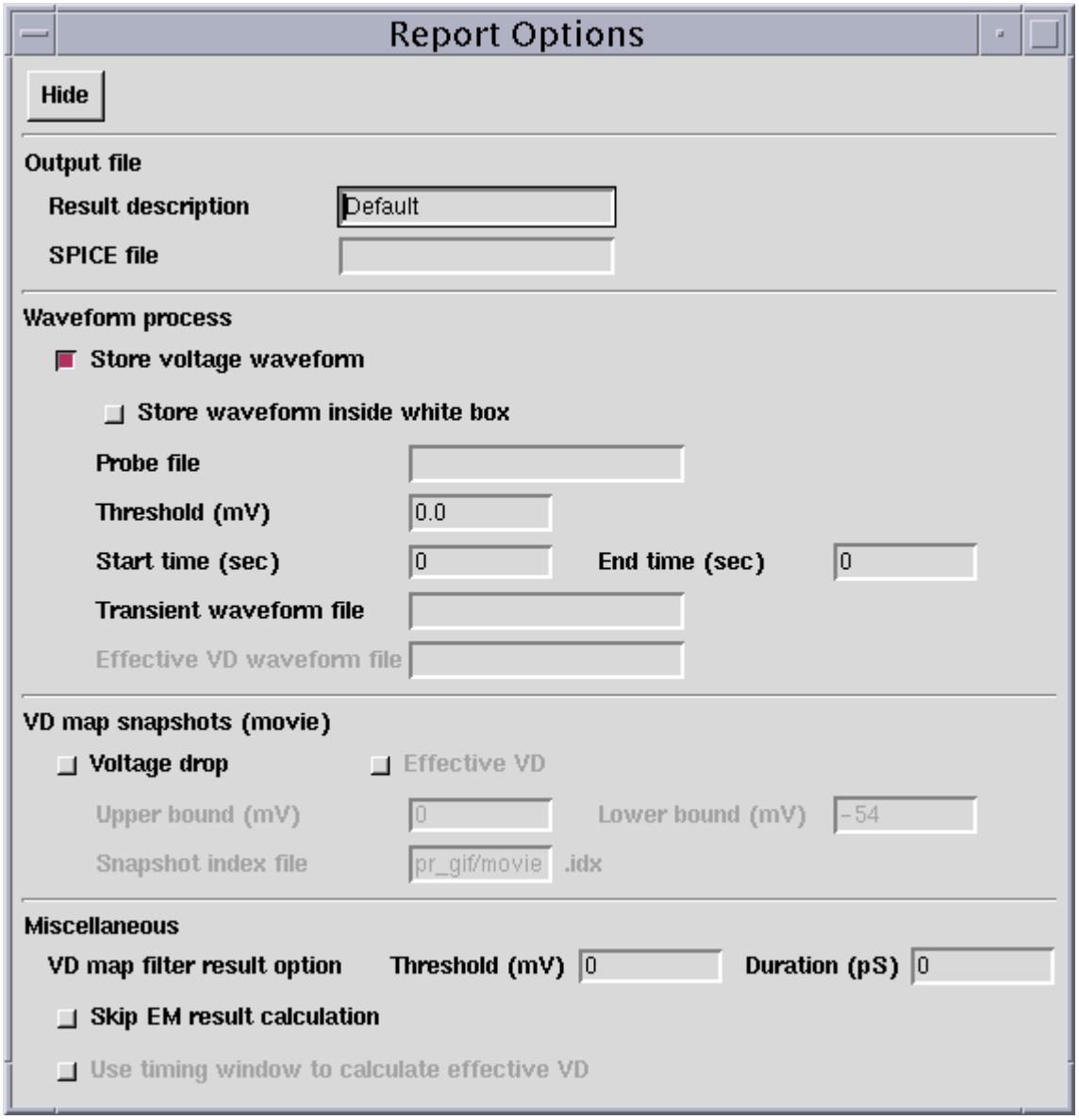
All the settings are optional. If a time step and the number of steps are specified, they take priority over the length of the time window given.

For a detailed description about performing transistor-level dynamic rail analyses, see Chapter 9, “Performing Transistor-Level Dynamic Analysis,” in this user guide.

7. Click “Report Options” to specify which output file you want to generate during rail analysis.

   The Report Options dialog box opens.

Figure 7-12 Specifying Report Options for Rail Analysis



In the “Output file” section,

- Result description – Specify the name to be assigned to the output analysis result file. This result description file is used when you want to load a map through the `pgMap` command.
- SPICE file – Specify the name of the SPICE file if you want PrimeRail to output the information in a SPICE format.

In the “Waveform process” section, specify the waveform options when performing dynamic rail analysis.

- Store voltage waveform – Save the voltage waveforms captured during dynamic rail analysis.
- Store waveform inside white box – Save the voltage waveforms that are inside the white box model during dynamic rail analysis.
- Probe file – Enter the name of the probe file which contains a list of pin names.

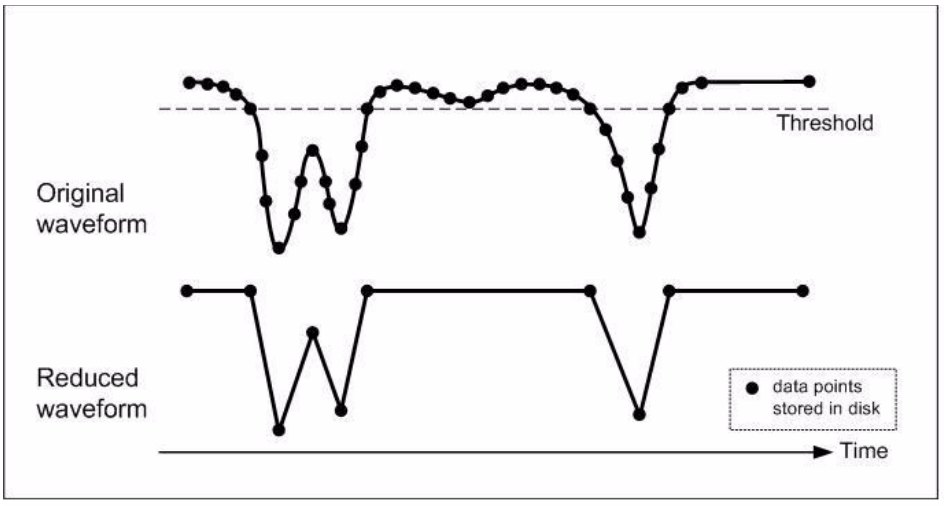
  By default, PrimeRail stores waveforms for all power and ground pins in a voltage waveform file. The size of the voltage waveform file can be huge, especially for the dynamic analysis flow. To reduce the size of the waveform file, you can limit the tool to store voltage waveforms only for the pins specified in the probe file.

  The following is a sample probe file:

```
TOP.MACROA.U212.VDD
TOP.MACROA.U212.VSS
TOP.MACROB.U211.VDD
TOP.MACROB.U211.VSS
```

- Threshold (mv) – To reduce the size of the output file, set the voltage threshold in millivolts. Specifying a threshold allows the tool to save only the turning points—including start time, end time and peak time—of the waveform. The default value is 0.0. While the threshold is larger than zero, PrimeRail stores only the turning points of the waveforms that are greater than the threshold value. This option can significantly reduce the waveform file size.

  The following figure compares the waveforms before and after a threshold is set.



The output transient waveform file is saved in the working directory.

- Start time (sec) – Enter the start time to capture the voltage waveforms. By default, the value is automatically synchronized with the simulation start time specified in the Power & Current Calculation dialog box (see Figure 7-7 on page 7-17).
- End time (sec) – Enter the end time for capturing the voltage waveforms. By default, the value is automatically synchronized with the simulation end time specified in the Power & Current Calculation dialog box (see Figure 7-7 on page 7-17).
- Transient waveform file – Enter the name of the transient waveform file where the voltage waveforms at each power and ground port of cell instances are to be saved.

  By default, the tool saves the transient waveform information to the file in the FSDB format.

  Because the output file is usually large, PrimeRail does not make a separate copy of the result. The transient waveform file is simply a link to the PrimeRail internal waveform storage used during map display and result reporting. If you decide to rerun the `poRailAnalysis` command, you must make a copy of the transient waveform file. Otherwise the internal waveform storage will be overwritten by the next run.

In the “VD map snapshot” section, specify the options for capturing GIF files for displaying an animated voltage drop map when performing dynamic rail analysis.

- Capture transient VD map – Generates multiple GIF files during dynamic rail analysis. These GIF files are used to display the animated voltage drop map. For a description of how to display an animated map, see “Viewing Animated Voltage Drop Maps” on page 8-26.

  By default, this option is off.

  Upper bound (mV) / Lower bound (mV) – Specify the range of voltages to be drawn with different colors. For power nets, the default upper bound is 0 and lower bound is -100. For ground nets, the default upper bound is 100 and lower bound is 0. Violations are colored in red, as set in the `pgMap` command.

  Snapshot index file – Enter the name of the snapshot index file to be generated. This file contains information of the directory where the snapshot index file is located and the prefix of the output GIF files. The output GIF files will be placed under the same directory as the snapshot index file.

  By default, the snapshot index file is named pr_gif/movie.idx, where pr_gif is the name of the directory.

  The output voltage drop map has the same zoom with the current editing view, such as the CEL view for cell level. The size and resolution of the GIF file are automatically determined by the current editing view.

In the Miscellaneous section,

- VD map filter result option – You can set a duration time to be used with a threshold for storing only the voltage drop spikes that are larger than the specified threshold and lasting over the duration time. This is helpful when you want to isolate the voltage drop violations; you can easily pinpoint the problematic area. This duration filter functionality is available only for the WDB flow.

  Specify values for duration time and threshold if you want to isolate voltage drop violations when displaying the voltage drop map.

8. Click OK or Apply.

At the end of the `poRailAnalysis` execution process, PrimeRail reports the minimum and maximum values of voltage drop (for a power net) or voltage rise (for a ground net) to the command window and to the log file. By default, the tool also reports the top five instances of peak voltage drop and time when they occur. Use the `pgStatListSize` switch if you want to change the default setting, by typing the following:

```
define pgStatListSize <value>
```

For example, if you specify the value as 1-100, the tool will report up to 100 instances in the output log file.

You can save the results of voltage drop and electromigration current density values to the database, by saving the CEL view that has just been analyzed. Doing this allows you to close the database and retrieve these results later.

Analyzing Designs Using the LEF/DEF Flow

When you are using the LEF/DEF flow, you need to set the `dbCreateCellProperty` switch if the design being analyzed contains I/O pad cells placed outside of the cell boundary. Otherwise, the tool reports the error that the design is not yet done placement.

To set the `dbCreateCellProperty` switch, enter the following in the command window before running the `poRailAnalysis` command:

```
dbCreateCellProperty (geGetEditCell) "gaTopCellStatus" 8
```

Checking for Voltage Drop and Current Density Violations

When rail analysis is complete, you can view problematic areas for voltage drop and electromigration graphically, by using a “map.” The problematic areas are highlighted in different colors. You can also choose to generate error cells and error reports for locations where limits are violated. You can choose to view the rail analysis results of your design in ASCII format for whatever scripting setup might serve the most useful purpose.

See Chapter 8, “Viewing Cell-Level Analysis Results,” for more information about viewing analysis results.

8

Viewing Cell-Level Analysis Results

PrimeRail enables you to view analysis results graphically in various types of maps, such as voltage drop map or electromigration map. Different colors can be assigned to indicate various step levels of voltage rise and drop or current density limits.

The following sections describe how to view analysis results in PrimeRail:

Displaying Power Maps

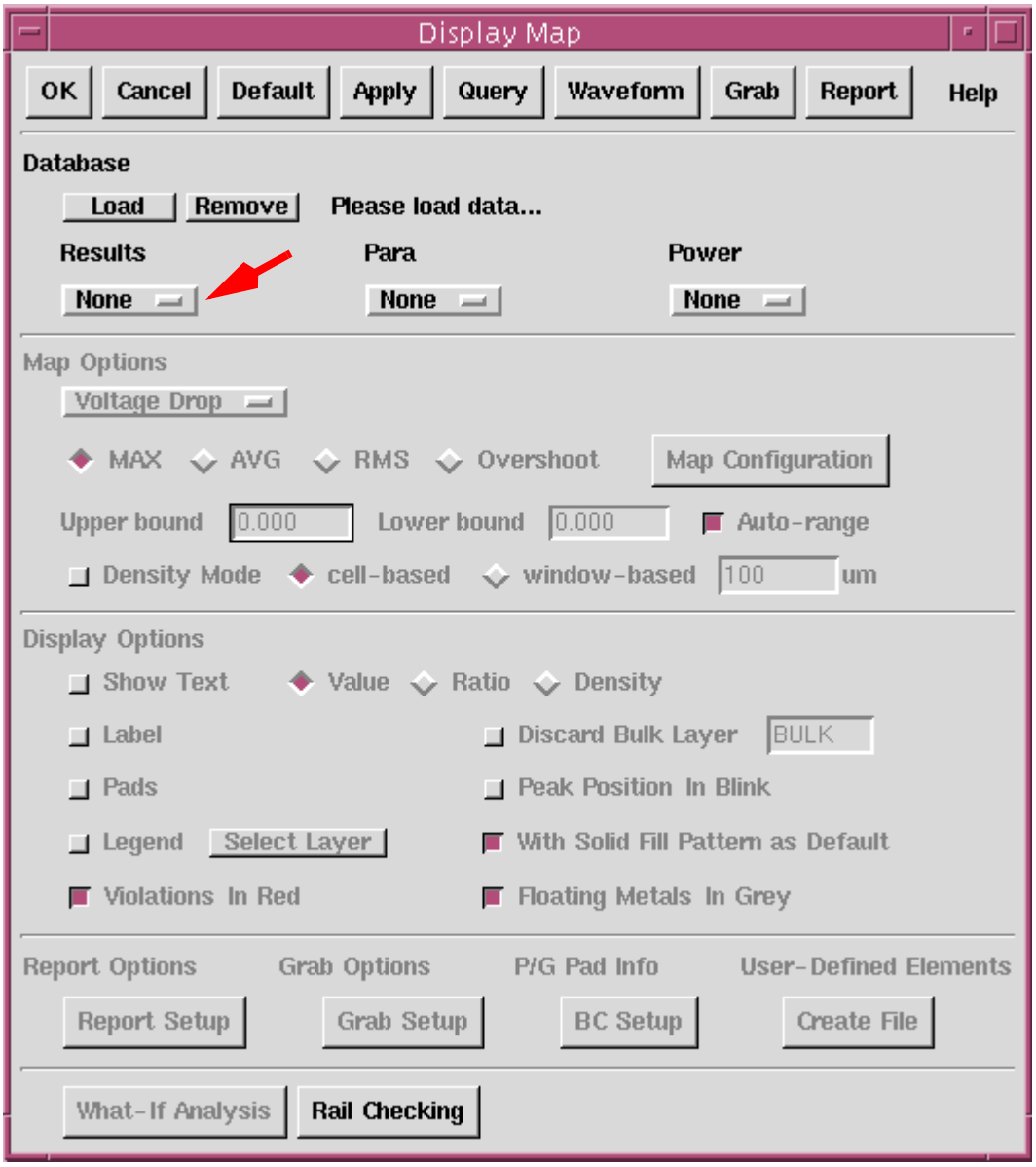
Use the power map to view and investigate the power distribution for a design. Displaying the power map before you perform power and ground net extraction and rail analysis will help you better understand where potential problem areas might occur.

To display the power map,

1. Enter `pgMap` in the command window or choose Cell-level Analysis > Display – Display Cell-level Maps.

   The Display Map dialog box opens.

Figure 8-1 Display Map Dialog Box



2. Choose which analysis results you want to display from the Results list. The Power list is automatically matched to the type of results you choose to display.

   When done, click the Load button to load the data to the database. You must load the data before displaying a map. Click Remove if you want to remove the data.

3. Map Options – From the menu, choose Power to display a power map, and specify the following options:

- Map Configuration – Specify additional options for map display in the Map Configuration dialog box that appears (see Figure 8-7 on page 8-17).

  In the Map Configuration dialog box, choose the layer to be displayed or modify the default layer colors to be used in the map.

  Color Configuration – Define the color of each step to be shown on the map in the Display Color Setup dialog box that appears (see Figure 8-7 on page 8-17).

- Density Mode – Display the power density map. You can choose to display the map on a cell-by-cell basis, or focus on a specific area of the layout in the size of your choice, which allows you to find hot spots more efficiently (see "Viewing Power Distribution by Power Density" on page 8-4).

  To set a view using multiple windows, select "window-based" and enter the dimension of the window.

4. Display Options – Select the options related to the power map.

   - Show Text – Display power distributions on the map by selecting one of the following options:

     Value – Display power values on the map.

     Density – Display power density values on the map.

   - Pads – Highlight the pads used as ideal voltage sources in rail analysis.
   - Legend – Display the upper and lower bound values of the metal and via layers on the map.
   - Violations In Red – Indicate violations in red on the map.
   - Peak Position In Blink – Show a blinking rectangle where the maximum value is located.

5. Click OK or Apply in the Display Map dialog box. The tool displays the results in the cell editing window.

Viewing Power Distribution by Power Density

The power density drawing capability of the `pgMap` command enables you to check whether the adjacent cell instances consume power in a highly fluctuating manner. The power density value of a cell instance is calculated by the following formula:

$$\text{power density} = \frac{\text{power consumption value}}{\text{area}}$$

If the design has a CONN view, PrimeRail calculates its power density values by using the current source data. The CONN view is first split into several subareas where each is sized up with a standard cell row. The number of current sources inside the subarea is then used to determine the power density value.

The method used for calculating the power density values of a white box model is similar to that of a CONN view. The white box model is first divided into several subareas. Each subarea then has a power density value equal to the ratio of the full white box model power to the subarea's dimension times the ratio of the subarea's total currents to the white box model total currents. The following is the formula:

$$\text{power density of subarea} = \frac{\text{total power}}{\text{subarea}} \times \frac{\text{total currents of subarea}}{\text{total currents of white box}}$$

Power Map Versus Power Density Map

Assume that the design has two adjacent cell instances, A and B. The size of instance A is 100 square millimeters and the power consumption value is 100 milliwatts. The size of instance B is 200 square millimeters and the power consumption value is 400 milliwatts. When a power map is drawn, the power consumption range of these two instances is 300. When a power density map is drawn, the power density values of instances A and B are 1 and 2, respectively.

Important:

Be sure that you complete power analysis on the soft macros before you display the power map or the power density map of the soft macros.

Drawing power density maps for CONN views of hard macros requires that the current source data be present in the database.

To display a power density map of a design, run the `pgMap` command and select Power and then "density mode" in the Display Map dialog box (see Figure 8-1 on page 8-3). Select other options if necessary. Click OK or Apply. The power density map of the design is drawn in the cell editing window.

Querying Power Values

When the power map is displayed (see "Displaying Power Maps" on page 8-2), you can check for the power consumption values of a hot spot. To do this, click Query at the top of the Display Map dialog box and then click a spot in the cell editing window. PrimeRail writes the values to the log and the command window.

The following is an example of querying power consumption values:

```
Map = Power
--------------------------------
clk: bBox = (121.200 251.600) (129.200 259.600)
   power = 0.332 mW, power density = 0.00518 mW/um^2

[ Power density map - cell-based ]

   The same as [ Power map ]

[ Power density map - window-based ]  - area query

Map = Power
--------------------------------
clk!6: bBox = (121.200 251.600) (129.200 259.600)
  power = 0.332 mW, power density = 0.00518 mW/um^2
clk!12: bBox = (107.600 251.600) (115.600 259.600)
  power = 0.278 mW, power density = 0.00434 mW/um^2
\m40_reg[10]: bBox = (113.200 242.800) (130.000 250.800)
  power = 0.0344 mW, power density = 0.000256 mW/um^2
------------------------------------------------
Total power under selected area = 0.644 mW
```

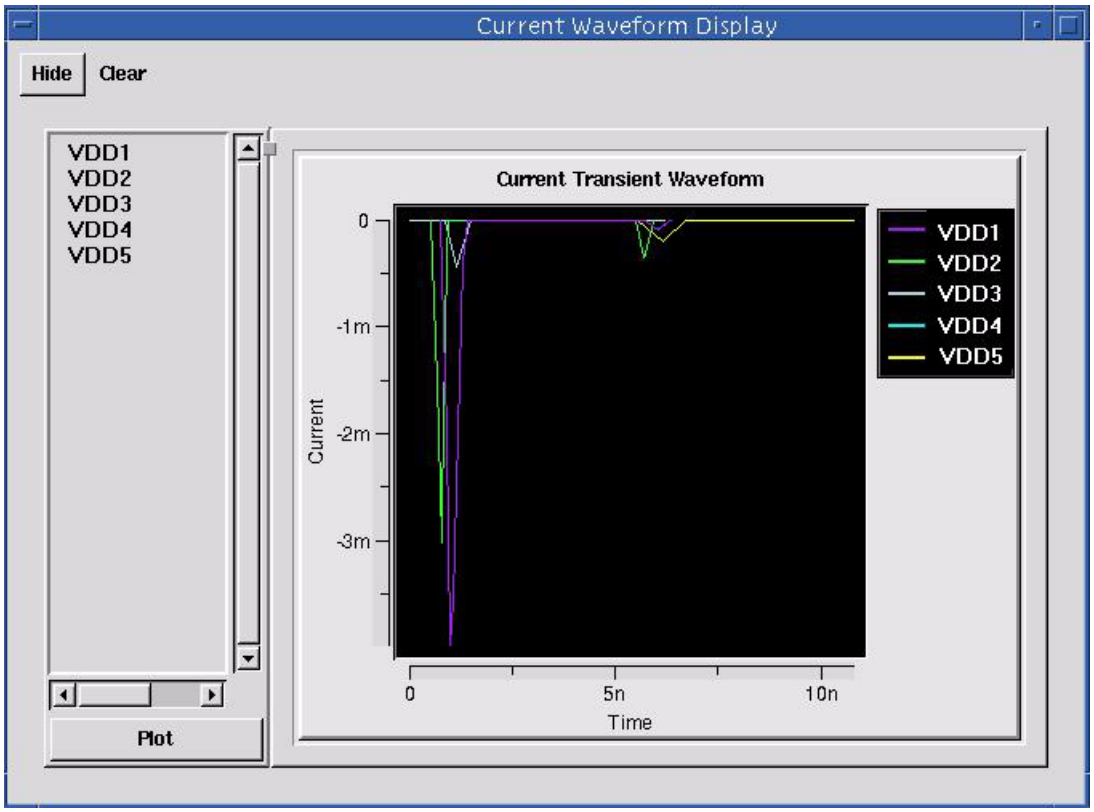
Viewing Current Waveforms

When power map is displayed (see “Displaying Power Maps” on page 8-2), you can display the current waveforms in the Current Waveform Display dialog box, shown in Figure 8-2 on page 8-7.

First you need to load the analysis results of a net and display the power map in the cell editing window, using the `pgMap` command. Then click the Waveform button at the top of the Display Map dialog box and click a current source point in the power map. The current source points are presented as white cross boxes; you should see the white cross boxes shown on the map as soon as you click the Waveform button.

The tool then draws the waveforms of the power or ground pin in the Current Waveform Display dialog box. Click another cross box if you want to check the waveforms of another net. To escape the query mode, click the right mouse button or press the Esc key.

Figure 8-2 Current Waveform Display Dialog Box



To clear the waveforms of a net, select the net in the list and click Clear. Click Hide to close the dialog box.

Before you view the current waveform of another time window, click Clear at the top of the Display Map dialog box to remove the results of the current window.

To reset the cell editing window to its normal status, click Clear at the top of the Display Map dialog box to remove the analysis results.

Displaying Parasitic Map

Use the `pgMap` command to display the parasitic resistors of a given power and ground net in the design through the parasitic map and to query the resistance value of a net.

In the parasitic map, the boundary node that can be an ideal voltage source location or a boundary hook-up point to the upper level is marked with a cross sign in yellow, and the current source location in blue.

To display the parasitics map,

1. Enter `pgMap` in the command window or choose Cell-level Analysis > Display – Display Cell-level Maps.
2. In the Display Map dialog box that appears (see Figure 8-1 on page 8-3), choose which analysis results you want to display from the Results list. The Para (parasitic) list is automatically matched to the type of results you choose to display.

   When done, click the Load button to load the data to memory. You must load the data before displaying a map. Click Remove to remove the data.
3. Map Options – From the menu, choose Parasitics to display a parasitic resistance map.
   - Map Configuration – Specify additional options for map display in the Map Configuration dialog box that appears (see Figure 8-7 on page 8-17).

     In the Map Configuration dialog box, choose the layer to be displayed or modify the default layer colors to be used in the map.

     Color Configuration – Define the color of each step to be shown on the map in the Display Color Setup dialog box that appears (see Figure 8-8 on page 8-17).
4. Display Options – Select the options related to the parasitics map.
   - Show Text – Display the value of the resistor on the map.
   - Pads – Highlight the pads used as ideal voltage sources in rail analysis.
   - Peak Position In Blink – Show a blinking rectangle where the maximum value of resistance is located.
   - With Solid Fill Pattern as Default – Draw the map with the shapes filled with a solid color.
5. Click OK or Apply in the Display Map dialog box. The tool displays the results in the cell editing window.

Querying Parasitic Values

When the parasitics map is displayed in the cell editing window, you can query the parasitic values of one spot in the design. To do this, click Query at the top of the Display Map dialog box and then click a spot in the cell editing window. PrimeRail writes the values to the log and the command window, as well as the information on parasitic data and current direction.

The following is an example of querying parasitic data:

```
Map = Parasitics
--------------------------------
Node 3202: layer = metal1, bBox = (161.200 267.340) (161.560
```

```
268.400)
  Connected edge 7276: res = 0.0577
  Connected edge 696: res = 0.211
```

Displaying Resistivity Maps

Running rail analyses in PrimeRail allows you to find power and ground grid weaknesses due to missing vias, missing stripes, unconnected hard IPs, and so on. Sometimes, however, it may be difficult to detect the design defects at the chip level by simply viewing voltage drop map or electromigration map, such as finding one missing via over millions vias in the design.

You can use the `pgMap` command to display a resistivity map where each node and edge on the power and ground network has a color corresponding to its resistivity value. If the design contains multithreshold-CMOS cells and the library has the multithreshold-CMOS characterization data, the tool supports displaying the resistivity map of the leaf power and ground nets from the top-level power resources via switching cells.

The resistivity value of a node is the resistance value on a signal path that is the smallest resistive path from the node to one of the design's boundary nodes (that is, ideal voltage sources, which can be power and ground pins, user-defined taps, or packagings). Because this value represents the resistivity only on a signal path, it is not exactly the same as the effective resistance value from the node to the global ground. It represents an upper bound of the effective resistance of a node to the ground.

You can draw the resistivity map right after power and ground extraction is done. Or you can display the map when rail analysis is complete. The resistivity map can detect the power and ground weaknesses that a voltage drop map cannot. For example, the resistivity map can show the weaknesses on the unpowered rows, which are not triggered by the input VCD file.

The major difference between drawing based on the rail analysis result and the extraction result is the boundary condition setup. If you have run rail analysis on the design being analyzed, the tool stores the boundary condition of this run in the result database and uses this information when drawing the resistivity map. Note that if this is the case, the settings defined in the P/G Pad Info dialog box through the `pgMap` command will be ignored.

Parasitic Map Versus Resistivity Map

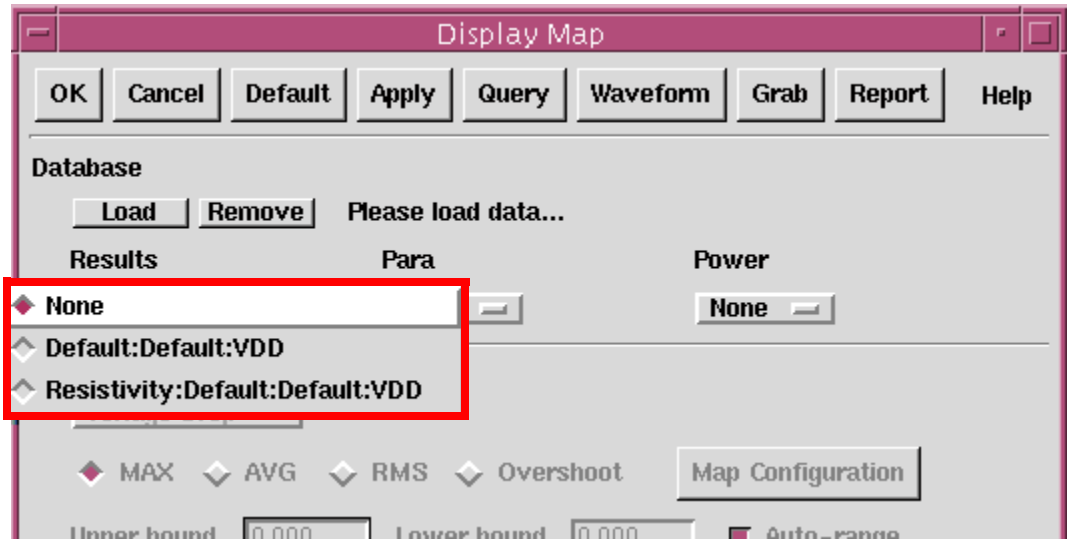
The parasitic map shows the resistance for any given polygon of a net. When you query on a net segment over the parasitic map, the value returned is the resistance of that segment. The resistivity map shows the "minimum resistance" from ideal voltage sources to any given point in the network. Thus when you query on a net segment over the resistivity map, the value returned is the "minimum resistance" from an ideal voltage source to this segment.

It is independent of the distribution of switching activity (that is, the VCD file) or current waveforms necessary for the voltage drop map to show power and ground grid weaknesses.

To draw the resistivity map after power and ground net extraction is done,

1. Run `pgMap` to open the Display Map dialog box (see Figure 8-1 on page 8-3).
2. In the Database section, select the result name from the Results pull-down menu. Note that if you are displaying the result calculated by the `poCalculateResistivity` command, choose the result name with *Resistivity* as a prefix, as shown in Figure 8-3.

Figure 8-3 Choosing Resistivity Information to be displayed

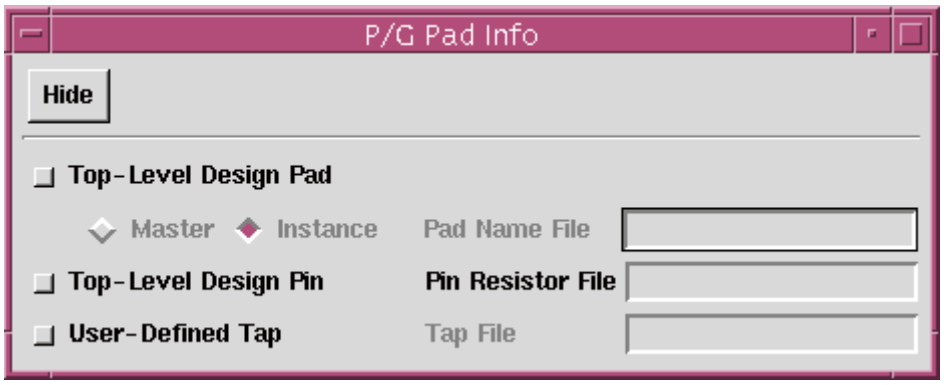


Select the net name from the Para pull-down menu. Then click Load at the top of the dialog box to load the parasitic data to the memory.

3. Choose Resistivity from the Map Options menu.
4. If you want to display the resistivity map by taking into consideration the voltage sources, click BC Setup at the bottom of the dialog box to specify the ideal voltage sources in the design. This is exactly the same as the P/G Pad Info section in the Rail Analysis dialog box.

   The P/G Pad Info dialog box opens, as shown in Figure 8-4.

Figure 8-4 P/G Pad Info Dialog Box



- Top-Level Design Pad – Select when the pad cells are placed and connected to the power and ground network, select Master and enter the name of the pad name file where the pad cell master is specified to be the ideal voltage source.

  When your design has pad cells hooked up to the network of choice but you want to set only some of them to be the source of ideal voltage, select Instance and enter the name of the pad name file where the pad cell instances are specified to be the ideal voltage sources.

- Top-Level Design Pin – Select when your design has no pad cells in place and you want to take the pin objects from the FRAM view as the source of ideal voltage.

  When the top-level design pins are connected to voltage sources through external resistances, you must specify a pin resistor file to be used.

- User-Defined Tap – Select when you are working with a block or a design that does not yet have pins defined, or perhaps the location or design of the pad cells are not finalized. Enter the name of the tap file that specifies where the ideal voltage is applied by coordinates and layer numbers.

You can use some of the options, such as the Top-Level Design Pin and User-Defined Tap options, in combination, to investigate whether the rail analysis results can be improved on the existing design.

For a detailed description of setting ideal voltage sources, see “Ideal Voltage Source Locations for Rail Analysis” on page 7-23.

5. Click Apply or OK.

Querying on Resistivity Maps

When the resistivity map is drawn, you can query the resistance information of an instance. To do this, first zoom to power and ground node location in the resistivity map and then click the Query button at the top of the Display Map dialog box. Now, click the power and ground

net node location. The tool will list the shortest resistance path from the selected instance to the closest pad in the command window, as well as the information of all the resistors on this path. The path will also be highlighted in the map.

Running What-If Resistance Analysis

When the resistivity map is drawn, executing the what-if resistance analysis will recalculate the resistivity values for the design. The what-if feature allows you to make a particular change to the power and ground network design and see the rail analysis result without going through the rail analysis setup steps.

Note that performing what-if analysis over voltage drop map reruns the rail analysis and updates the analysis result. However, when the map being displayed is the resistivity map, executing what-if analysis recalculates only the resistivity values.

For a detailed description about performing what-if analysis, see “Running What-If Resistance Analysis” on page 8-12 and “What-If Analysis” on page 12-21.

Checking Rail Data Through Resistivity Map

When the resistivity map is displayed, you can check the weakness of the power and ground network based on the resistivity map. The weaknesses to be checked include missing vias, discontinuous connections, and insufficient vias. If any error is found, you can then perform what-if resistance analysis immediately, without leaving the resistivity map.

Note:
Geometries inside hard macros are not examined.

PrimeRail performs rail checking on a net-by-net basis. As a result, via stacks might be falsely identified as missing because there are other power, ground, or signal nets at a certain location. The tool assigns geometries to this location in the output Scheme script file; these geometries, however, do not have any DRC or LVS checking. It is recommended that you always double-check the output script file before using it.

During rail checking, PrimeRail reads the following essential data from the Milkyway technology file:

Layers section:

- unitNomResistance

ContactCode section:

- cutWidth
- cutHeight

- lowerLayerEncWidth
- lowerLayerEncHeight
- minCutSpacing
- unitNomResistance
- upperLayerEncWidth
- upperLayerEncHeight

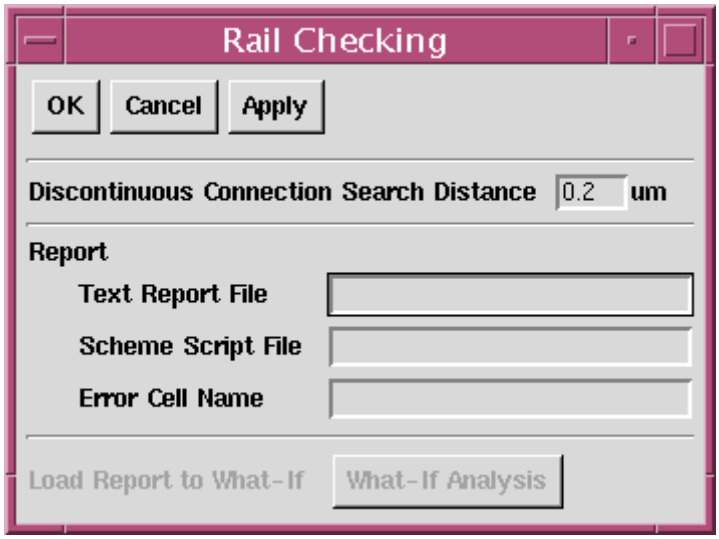
Before running rail checking based on the resistivity map, be sure you have loaded the parasitic data and the analysis results through the `pgMap` command. If you choose not to load the analysis results, you have to set up the boundary conditions by clicking the BC Setup button (see Figure 8-4 on page 8-11).

To check rail data based on the resistivity map,

1. Run `pgMap` and display the resistivity map. For information about how to display the resistivity map, see "Displaying Resistivity Maps" on page 8-9.
2. When the resistivity map is displayed, click the Rail Checking button at the bottom of the Display Map dialog box.

   The Rail Checking dialog box appears, as shown in Figure 8-5.

Figure 8-5 Rail checking over resistivity map



3. Enter the distance threshold used to check discontinuous connections. For geometries, the tool searches for possible discontinuous connections in the given distance threshold. If the distance between two rectangles is farther than the given threshold, the tool does not check whether the connection is discontinuous or not.

By default, the tool uses the minimum spacing on the metal1 layer as the distance threshold.

4. In the Report section, specify the report information. You can write the rail checking results to an ASCII file, a Scheme file, or an error cell.
   - Text Report File – Enter the name of the report file to be generated, in ASCII format.
   - Scheme Script File – Enter the name of the report file to be generated, in Scheme script format.

     Note that only top-level cell violations will be reported. Any violations for leaf cells will not be written to the output Scheme script file.
   - Error Cell Name – Enter the name of the error cell to be generated. It is recommended that you use error cells to debug the violations.
5. Click OK or Apply.

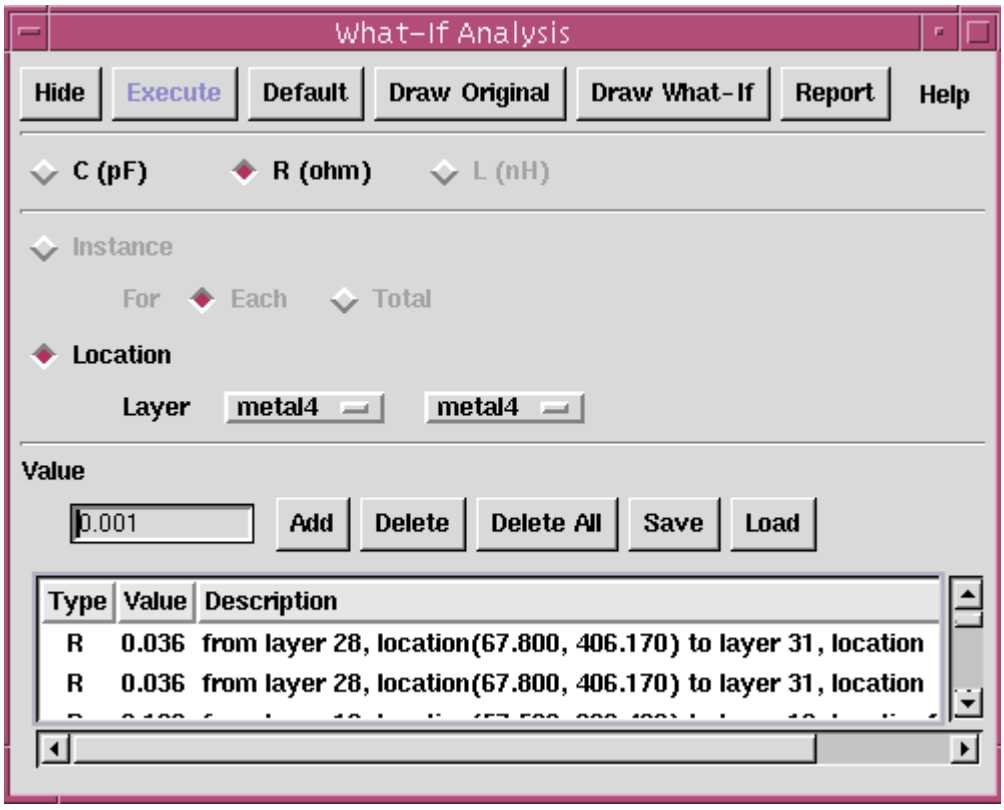
Inserting What-If Resistors After Rail Checking

You can perform what-if analysis based on the rail checking results.

When rail checking is complete, the What-If Analysis button at the bottom of the Rail Checking dialog box (see Figure 8-5 on page 8-13) will be enabled. Click the What-If Analysis button to open the What-If Analysis dialog box, as shown in Figure 8-6 on page 8-15.

The What-If Analysis button is dimmed if no rail checking data is available.

Figure 8-6 What-If Analysis Dialog Box



The tool automatically inserts virtual resistors in the design based on the rail checking results. The inserted virtual resistors are listed in the list box at the bottom of the What-If Analysis dialog box.

Examine the elements listed in the list box. If you want to delete a resistor element, select the resistor and click Delete. If you want to add a resistor, specify the layer information and value for the resistor being added. Then click Add.

When done, click the Draw What-If button at the top of the What-If Analysis dialog box to redraw the resistivity map with the inserted virtual resistors. Click Draw Original if you want to display the resistivity map without these virtual resistors.

For more information about performing what-if analysis, see “What-If Resistance Analysis” on page 12-27.

Displaying Voltage Drop and Electromigration Maps

To display the voltage drop and electromigration maps,

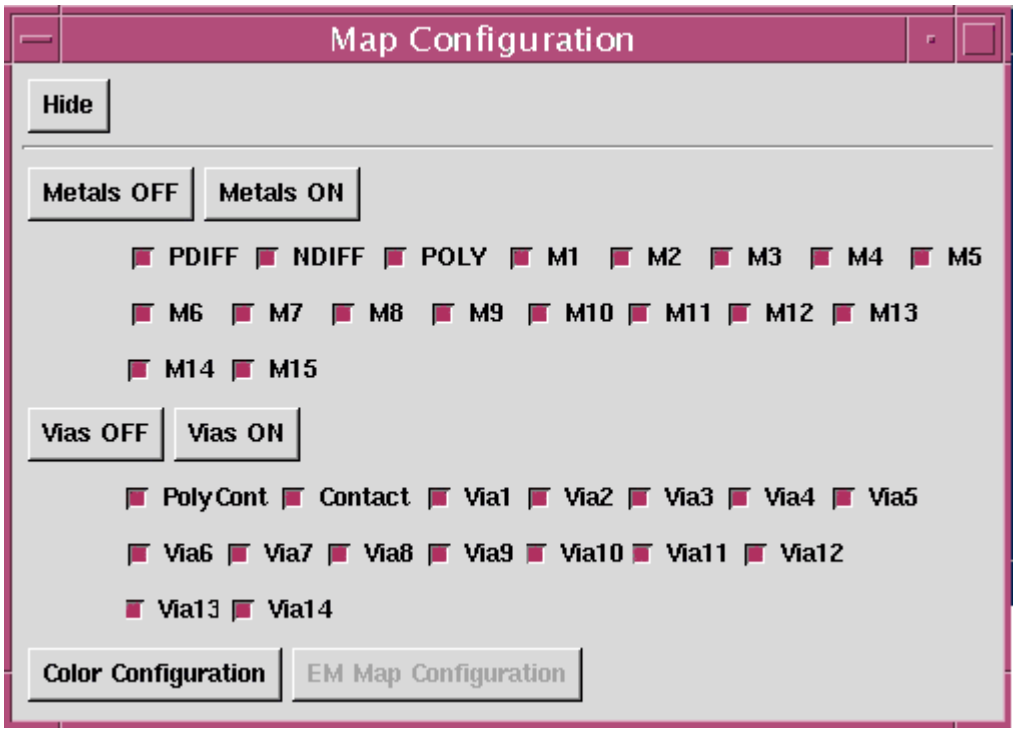
1. Enter `pgMap` in the command window or choose Cell-level Analysis > Display – Display Cell-level Maps. The Display Map dialog box appears (see Figure 8-1 on page 8-3).
2. Database – Choose which analysis results you want to display from the Results list.

   When done, click the Load button to load the data chosen to the memory. You must load the data before displaying a map. Click Remove if you want to remove the data.
3. Map Options – From the menu, choose Voltage Drop to display a voltage drop map, or EM to display an electromigration map.
   - MAX/AVG/RMS – Display the maximum, average, or RMS value of voltage or current.

     Overshoot – Display voltage overshoot and ground undershoot. Overshoots usually occur due to the inductance of a circuit. This option is applicable only to the voltage drop map.

     For example, when you want to display voltage waveforms for a net, choose MAX to display the peak voltage drop and ground bounce of the waveform. Choose Overshoot to display the peak voltage overshoot and ground undershoot of the waveform.
   - Map Configuration – Specify additional options for map display in the Map Configuration dialog box that appears, as shown in Figure 8-7.

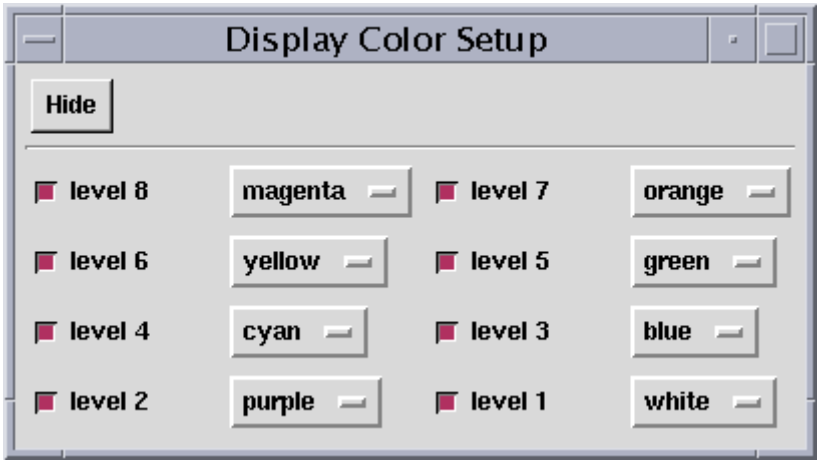
Figure 8-7 Map Configuration Dialog Box



In the Map Configuration dialog box, choose the metal or via layer to be displayed. To modify the default layer colors to be used in the map, see the following Color Configuration section.

Color Configuration – Click to define the color of each step to be shown on the map in the Display Color Setup dialog box, as shown in Figure 8-8.

Figure 8-8 Display Color Setup Dialog Box



Click Hide to apply the changes you make and return to the Map Configuration dialog box.

EM Rules Configuration – If you are displaying an electromigration map, click to define the lower and upper current density bounds in the EM Map Configuration dialog box, as shown in Figure 8-9.

Figure 8-9 EM Map Configuration Dialog Box

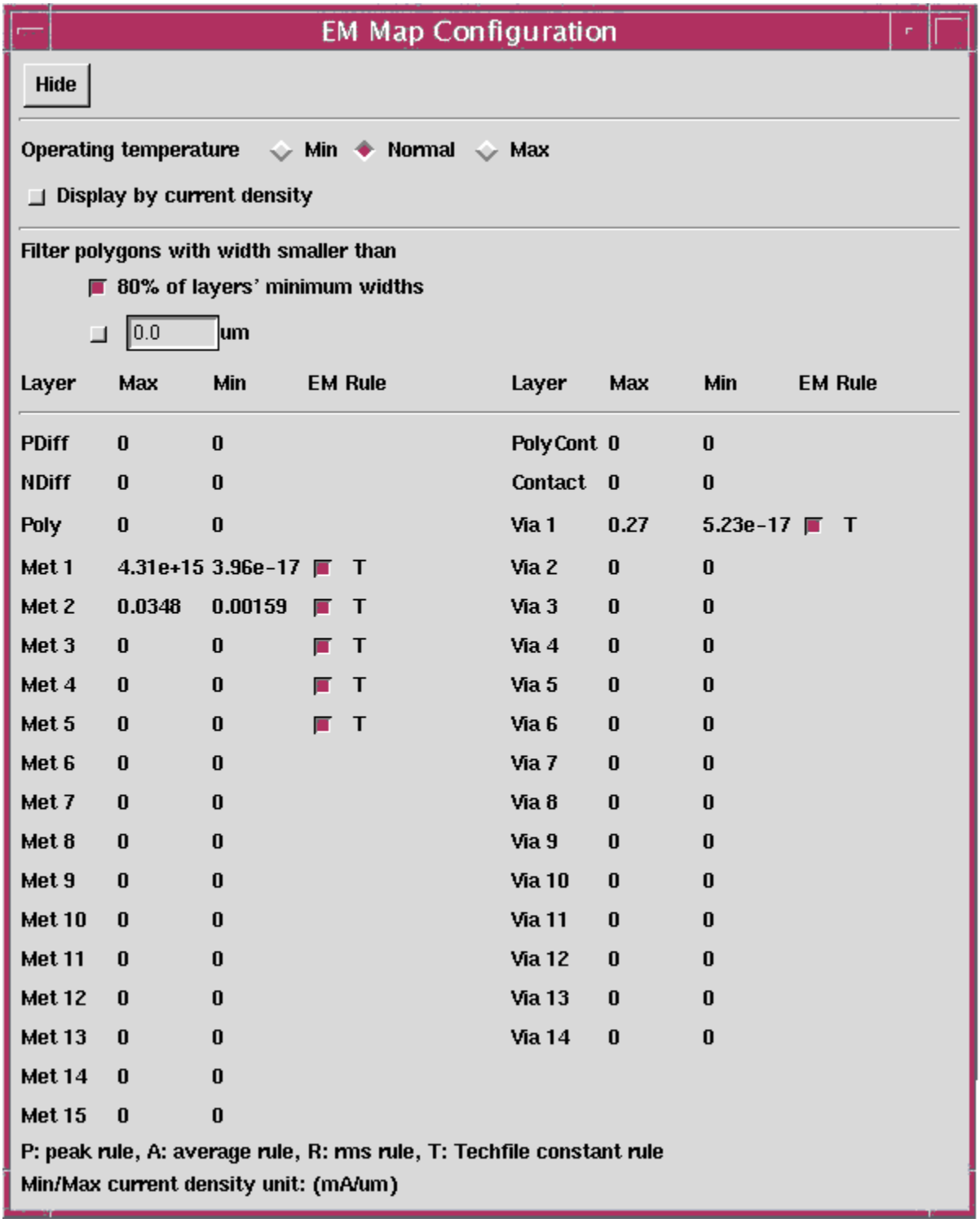


Table 8-1 Option Descriptions EM Map Configuration Dialog Box

Option	Description
Operating temperature	Choose the temperature to be used with the time-average and RMS electromigration rule file that is width or temperature dependent.
Display by current density	Select to display the electromigration map based on the current density values of the selected metals or vias. Note that when the “Violation in Red” option is enabled in the Display Map dialog box (see Figure 8-1 on page 8-3), the tool draws the nodes with violations in color red, and the rest by the variation of current density values in colors as defined in the Display Color Setup dialog box (see Figure 8-8 on page 8-17). In cases when vias are extracted as nodes, not edges, a node hash is built for accumulating the current that flows through the vias.
Filter polygons with width smaller than	Select if you do not want to display the polygons that have widths smaller than 80 percent of the layer’s minimum width as defined in the technology file, or smaller than the value you specify.

If you loaded electromigration rules through ALF files, the EM Rule block of a layer will be highlighted to indicate the presence and type of the electromigration rule—for example, P for peak (maximum), A for average, R for RMS, and T for the constant current density threshold in the technology file.

Click Hide to apply the changes you make and return to the Map Configuration dialog box.

When you complete configuring settings in the Map Configuration dialog box, click Hide to apply the changes you make and return to the Display Map dialog box.

- Upper bound / Lower bound – Enter the values for the upper and lower voltage drop limits. The tool draws the voltages higher than the upper bound in red and disregards those lower than the lower bound.

  This option is applicable only to the voltage drop map.

  Auto-range – Use the maximum voltage as upper bound and the minimum voltage as lower bound.

4. Display Options – Select the options related to the map you choose to display in step 3.

- Show Text – Display voltage drop or current density violations on the map by selecting one of the following options:

  Value – Show the voltage drop or current density value as an absolute value. This option is on by default.

  Ratio – Show the voltage drop or current density value as a ratio to a specified threshold.

- Label – Display the design name, analysis type, net name, and key analysis characteristics in the map.
- Pads – Highlight the pads used as ideal voltage sources in rail analysis.
- Legend – Display the upper and lower bound values of the metal and via layers on the map. Click Select Layer to choose the layer whose upper and lower bound values are to be shown in the map.
- Violations In Red – Indicate violations in red on the map.

  When selected, the tool draws the violations in color red and draws other nodes by the following color ratio:

  ```
  R (violation) = current/current_threshold
  ```

  where R < 1. (R>=1 means a violation.)

  Otherwise, the tool colors the nodes without violations as the lowest level. By default, the color for the lowest level is white, but you can modify the color setting through the Display Color Setup dialog box (see Figure 8-8 on page 8-17).

  Violations are rendered by the following ratio:

  ```
  R (violation) = current/current_threshold
  ```

  where R >=1.

- Discard Bulk Layer – Ignore the bulk layer specified in the text box when displaying the voltage drop map. Note that this option is not applicable in displaying the electromigration map.
- Peak Position In Blink – Show a blinking rectangle where the maximum value of voltage drop or current density is located, based on the type of map to be displayed.
- With Solid Fill Pattern as Default – Draw the map with the shapes filled with a solid color.
- Floating Metals In Grey – Show floating metals on the map.

5. Now click OK or Apply in the Display Map dialog box. The tool displays the results in the cell editing window.

Querying Voltage Drop and Current Density Values

When the voltage drop or electromigration map is displayed, you can check for voltage drop or current density values of a net through the map.

To query values of the net, click Query at the top of the Display Map dialog box and then click the net being studied in the cell editing window. The tool writes the values to the log and the command window. If the maximum value is available, the tool also reports the time when peak voltage or current occurs in nanoseconds.

```
Map = Voltage Drop
Current density ranges:
Poly (mA/um): min = 2.0393^-10, max = 0.0122515
Metal1 (mA/um): min = 3.17317^-10, max = 0.0441959
Metal2 (mA/um): min = 0.0040853, max = 0.0122559
Metal3 (mA/um): min = 4.72318^-10, max = 4.72318^-10
Creating query database...
--------------------------------
Edge 83: layer = metal1, bBox = (60.000 133.290) (80.000 133.650)
         voltage drop = -7.94, res = 6.39
Node 52681: layer = metal1, bBox = (407.100 460.350) (407.220 460.709)
         voltage drop = -8.451, time = 3270.180

Map = Electromigration
--------------------------------
Node 3202: layer = metal1, bBox = (161.200 267.340) (161.560 268.400)
  current density = 0.028
  Connected edge 7276: res = 0.0577
  Connected edge 696: res = 0.211

Map = Electromigration
--------------------------------
Edge 592: layer = metal1, bBox = (141.800 249.740) (144.760 250.800)
  current density = 0.0667, res = 0.179
```

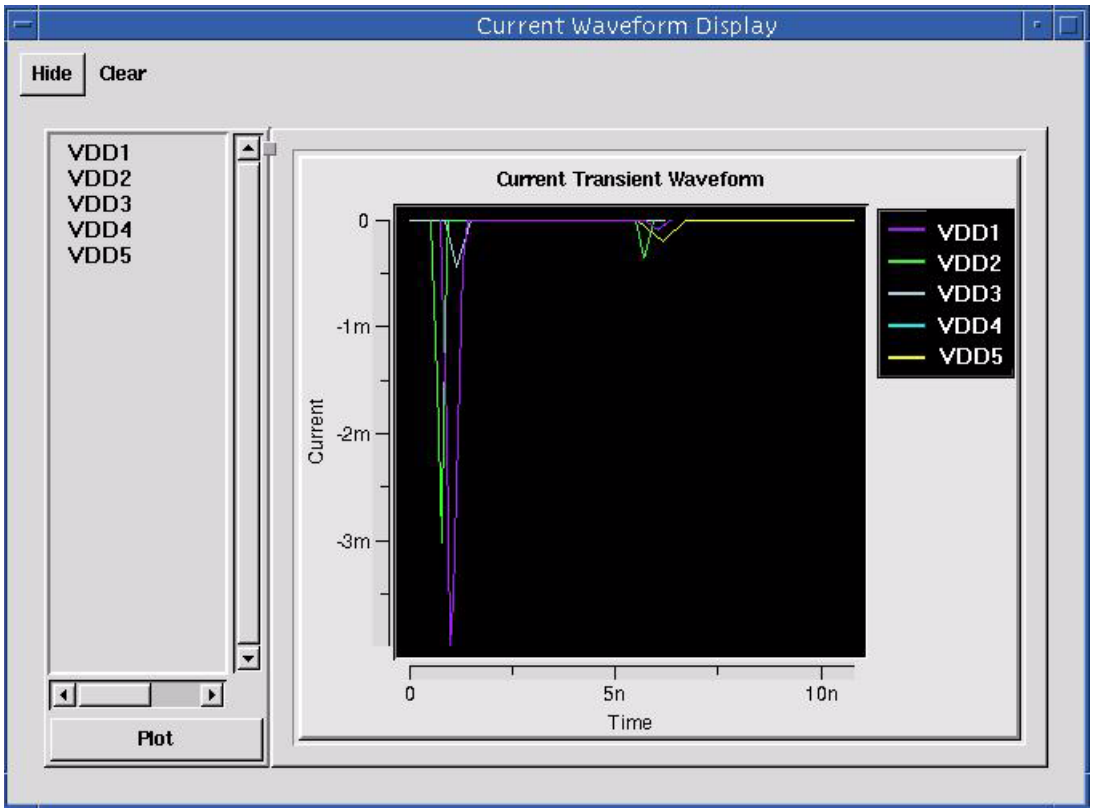
Viewing Voltage Drop Waveforms

Through the Display Map dialog box (`pgMap`), you can open the waveform viewer and examine the voltage drop waveforms of a net in the design.

You first need to load the analysis results of a net and display the voltage drop map in the cell editing window using the `pgMap` command. Then click the Waveform button at the top of the Display Map dialog box. Next, click a white cross box in the voltage drop map to examine the waveforms of that spot. The tool then draws the waveforms of the power or ground pin in the Voltage Waveform Display dialog box that appears, as shown in Figure 8-10.

Click on another cross box if you want to check the waveforms of another spot. To stop querying the waveform, click the right mouse button or the Esc key.

Figure 8-10 Voltage Waveform Display Dialog Box



To clear the waveforms of a net, select the net in the list and click Clear. Click Hide to close the dialog box.

Before you view the voltage waveform of another time window, click Clear at the top of the Display Map dialog box to remove the results of the current window.

To reset the cell editing window to its normal status, click Clear at the top of the Display Map dialog box to remove the analysis results.

Reporting Voltage Drop and Current Density Violations

If you want PrimeRail to report an error cell or file on voltage drop or current density violations, click Report Setup at the bottom of the Display Map dialog box (see Figure 8-1 on page 8-3). The VD Report Setup or EM Report Setup dialog box opens, depending on whether the voltage drop or electromigration map is displayed, as shown in Figure 8-11 and Figure 8-12.

Figure 8-11 Generating Voltage Drop Violation Reports

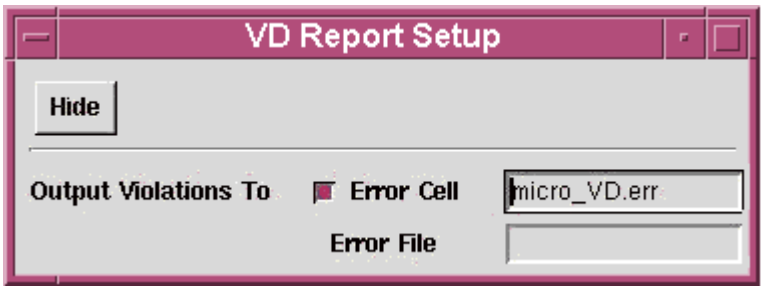
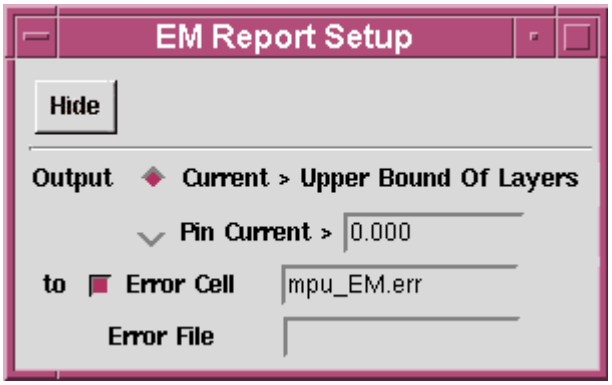


Figure 8-12 Generating Current Violation Reports



Enter the name of the error cell to be created, such as top_VD.err, in the Error Cell text box. If you want to generate an ASCII error file, enter the name of the file you want to create in the Error File text box. Click Hide to return to the Display Map dialog box.

Next, click Report at the top of the Display Map dialog box. The error cell or error file created will reside in the working directory. Be sure you have displayed either the voltage drop or electromigration map before generating an error cell or file.

Here is the format of the output electromigration report:

```
* Title           : P/G Net <MODE> (peak|avg|rms) Electromigration
                    Violation Report
* Date            : <DATE>
*
* Vendor          : Synopsys
* Program         : PrimeRail
* Version         :  <version>
* Design          : <design name>
* Hier Separator  : /
* Temperature     : 25.0 C
*
* Units:
*   Length Unit         : um
*   Current Unit        : mA
```

```
*   Conduct Layer Density Unit : mA/um
*   Contact Layer Density Unit  : mA/VIAs (mA/um^2 if VIA  count is
unavailable
)
*  Conduct Layer Density Threshold Unit: mA/um
*  Contact Layer Density Threshold Unit : mA/VIAs
      (mA/um^2 if VIA count is unavailable )
*
* Total EM violations    : <violation count>
*****************************************************
* EM Rule Table:
<Shows ALF rules>


* Scaling Factor       :  <scaling Factor>
*

* # Violation of each layers
* <layer> violation:  <violation count by layer>
  ...
*
* Constant EM Threshold (Technology File Max Current Density):
*
*
*      Layer   | Threshold |                  | Layer      | Threshold |
*     ---------+-------------+-------------+-----------+--
*      <layer>| <value>       | <has rule> | <layer> |<value> |
       ...
```

The constant EM threshold section shows the maximum density rule in the technology file. The EM Rule Table section shows if the layer has a rule (either in a technology file or an ALF file).

The following is an example of an electromigration violation report:

```
CELL:micro INST: NET:VSS
NODE     LAYER  No MODE CURRENT (>CURRENT   WIDTH  NEW_WIDTH  (LEFT,
BOTTOM)
(RIGHT, TOP) RES NAME  DENSITY  DENSITY    AREA      NEW_AREA  NAME
THRESHOLD)
VIACNT NEW_VIACNT
=========================================================
R10036   metal1 16 avg     -0.31(>0.2) 1.06   1.64 (102.000, 251.600)
(107.599, 252.659)
```

The violation report is sorted by layer name and categorized by cell (or block).

Viewing Animated Voltage Drop Maps

The animated voltage drop map helps in debugging possible dynamic reliability effects found during dynamic rail analysis. When you run the dynamic rail analysis for multiple clock cycles, the animated voltage drop map allows you to easily pinpoint the critical cycles and perform a more detailed analysis to solve the problems. In the animated voltage drop map you can view the voltage drop distribution for each time point, find out which cycles cause IR drop problems, and then run the detailed analysis only on those problematic cycles.

To view an animated voltage drop map, you first need to configure the settings in the “VD map snapshot” section of the Report Options dialog box (see Figure 7-12 on page 7-32) during dynamic rail analysis with the `poRailAnalysis` command. You must do this because in PrimeRail the animated voltage drop map is composed of snapshots by time; each snapshot is saved as a GIF file. When dynamic rail analysis is done, the tool saves the multiple GIF files in the directory. You then run the `pgMoviePlayer` command to invoke a movie player where the output GIF files are played as the animated voltage drop map.

Figure 8-13 shows the graphical user interface of the movie player.

Figure 8-13 Animated Voltage Drop Map Displayed in the Movie Player

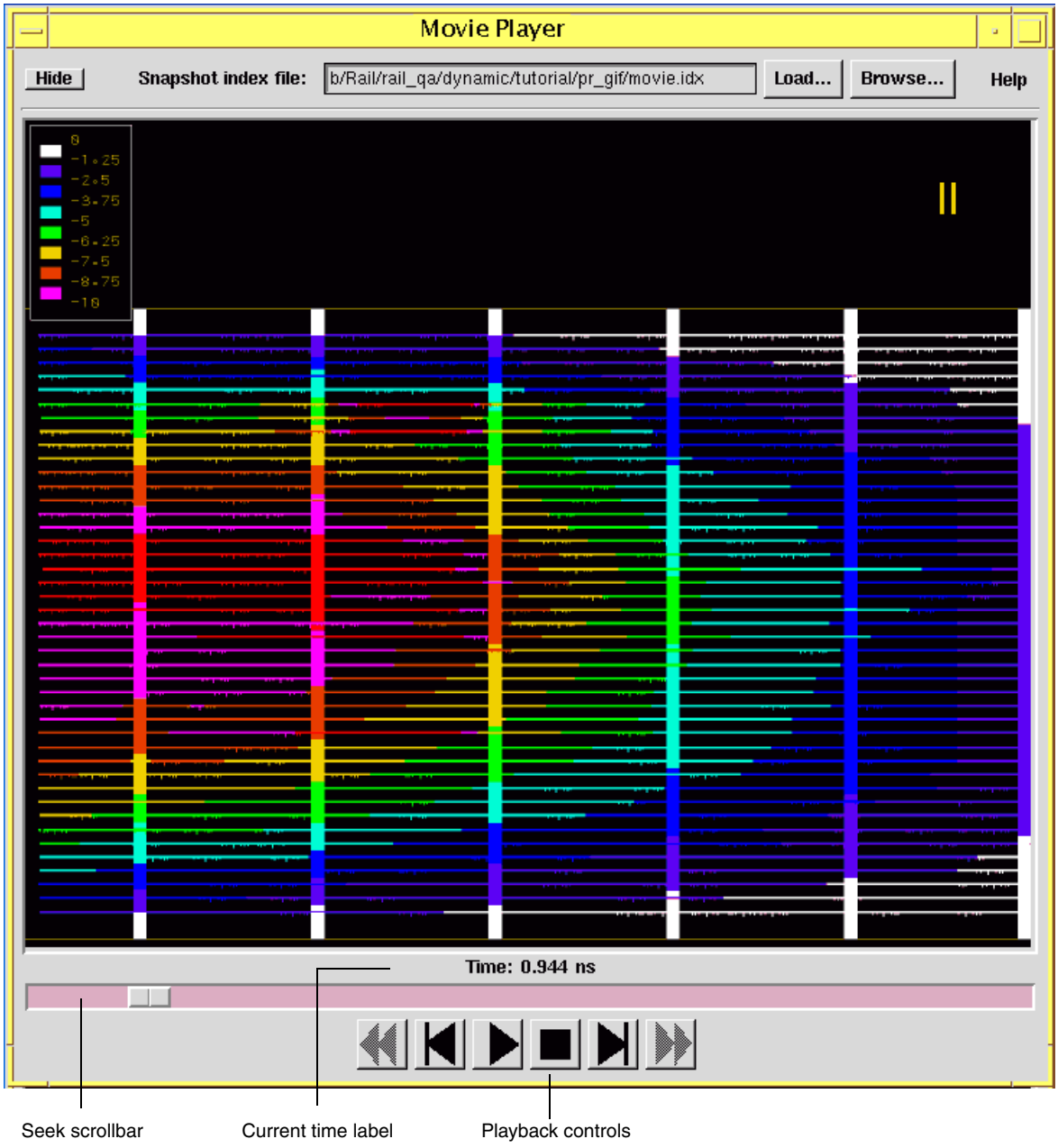


Table 8-2 Options on the Movie Player Dialog Box

Name	Description
Hide Button	Close the movie player.

Table 8-2 Options on the Movie Player Dialog Box (Continued)

Name	Description
Snapshot Index File	The name of the snapshot index file generated by `poRailAnalysis`. A snapshot index file contains the information about the snapshot frames, including the location, the file name, and the associated time information.
Load ... Button	Load the snapshot index file assigned in the “Snapshot index file” text box.
Browse ... Button	Select a snapshot index file through a file browser. When a file is selected, the tool automatically updates the “Snapshot index file” text box and loads the selected snapshot index file.
Movie Panel	The center panel where snapshot frames are displayed.
Current Time Label	The time associated with the currently displayed snapshot frame.
Seek Scrollbar	Move the scrollbar to the place in a selected time frame that you want to play.
Slower Button	Slow down the playing speed. The default speed is 1 frame per second. When the button is clicked, the speed will slow down 2 times; that is, 1 frame every 2 seconds. The tool will print the current speed on the right top corner on the movie panel, such as <<2X. The play speeding can slow down up to 8 times.
Previous Button	Play the previous snapshot frame.
Play Button	Start playing the movie. When the movie is played, the Play button will become the Pause button.
Pause Button	Pause the currently playing movie.
Stop Button	Stop playing the movie.
Next Button	Play the next snapshot frame.
Faster Button	Accelerate the playing speed. The default speed is 1 frame per second. When the button is clicked, the speed will speed up 2 times; that is, 2 frames per second. The tool will print the current speed on the right top corner on Movie Panel, such as 2X>>. The playing speed can be faster up to 8 times.

Usage Example

The animated voltage drop map can be useful in debugging. For example, when you have run dynamic power analysis with a full VCD file, you first perform rail analysis by using a larger time step size without generating the waveform outputs and calculating electromigration results to optimize the speed. By default, the step size for rail analysis is set to one-third of the average transition time. For instance, you can use the average transition time as the initial step size. The voltage drop snapshot is considered a time-specific voltage drop map, which is different from the maximum (or average) voltage drop map as displayed by the `pgMap` command. The latter represents worse-case events over the whole simulation time frame.

When voltage drop snapshots are generated by `poRailAnalysis`, run `pgMoviePlayer` to open the movie player and assess the critical cycle that causes the IR drop problems. Given the simulation time period and waveform probe file, you can limit the waveform data size to be written. This solves the performance and capacity issues while reading huge FSDB files. Then rerun rail analysis with a smaller step size, such as, the default step size, in the given timing period. Doing so gives you a detailed view of the voltage waveform of the signals of interest within the specified time frame.

Viewing Effective Voltage Drop Effects

When cell-level dynamic rail analysis or combined rail analysis is complete, PrimeRail allows you to investigate effective voltage drops on a cell instance. The instance-based effective voltage drop map shows the total reduction of cell bias caused by the voltage drop in VDD and the ground bounce in VSS. You can then use the output effective voltage drop report in PrimeTime SI for timing analysis.

The steps for generating effective voltage drop waveform files for viewing, using the `pgMap` command, are different, depending on whether you were performing a cell-level dynamic rail analysis or a combined rail analysis.

When a cell-level dynamic rail analysis has been performed

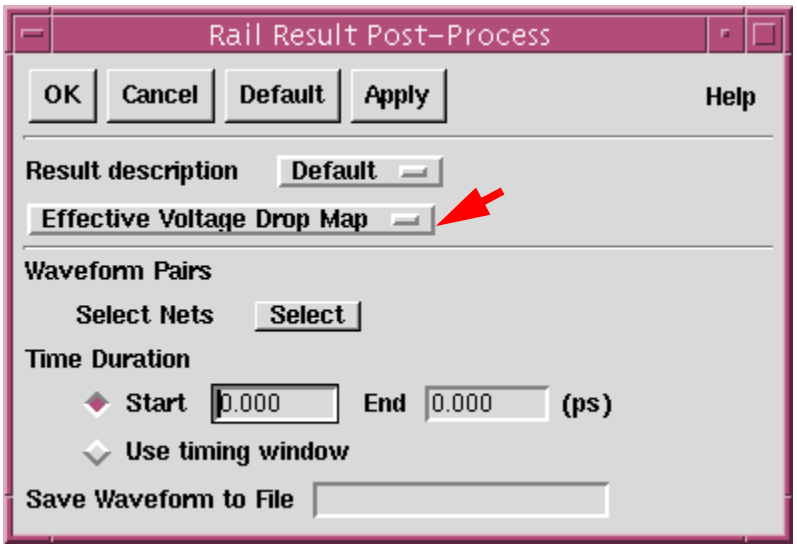
To avoid memory overloading, use the `pgResultPostProcess` command to process the FSDB file and to select the desired timing window for effective voltage drop calculation. Then run the `pgMap` command to display the effective voltage drop map.

To view the effective voltage drop effects for a design,

1. When the dynamic current waveforms are generated with the `poRailAnalysis` command, choose Cell-level Analysis > Display – Postprocess Rail Results or enter `pgResultPostProcess` in the command window. The Rail Result Post Process dialog box then opens.

2. Select the result description and the type of post-process as Effective Voltage Drop Map. The tool displays the related options in the lower section of the dialog box, as shown in Figure 8-14.

Figure 8-14 Rail Result Post Process Dialog Box—Displaying Effective Voltage Drop Map



3. In the Waveform Pairs section, click Select to choose the nets whose dynamic current waveforms are to be processed in the Select Power Nets dialog box that appears.

   Click to select the nets whose analysis results are to be loaded. Click Select All to select all the nets in the list. Click Deselect All to uncheck all the nets in the list.

   Click Hide to return to the Rail Result Post Process dialog box.

4. In the Time Duration section, enter the start and end time for capturing the currents. Select “Use timing window” if a timing window is to be applied.

5. Enter the name of the file that you want to use to save the waveform information.

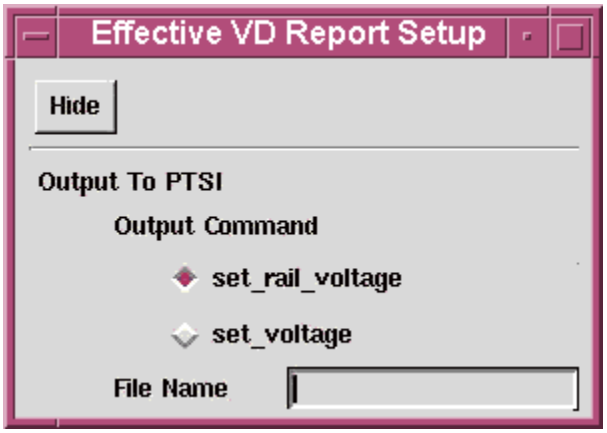
6. Click OK or Apply.

7. Then run the `pgMap` command to display the effective voltage drop map. In the Display Map dialog box that opens, select the Effective VD map option from the Map Options section and click Apply (see Figure 8-1 on page 8-3). The tool displays the effective voltage drop map in the cell editing window.

   Note that you must load the database before you select the effective voltage drop map option. The database is dependent on your database selection during Step 3. For example, if you select both VDD and VSS nets whose dynamic current waveforms are to be processed for effective voltage drop computation, but select only the VDD net during map loading, your effective voltage drop map or report is still based on both VDD and VSS nets.

8. To have the tool report the effective voltage drops, click Report Setup at the bottom of the Display Map dialog box to open the Effective VD Report Setup dialog box. See Figure 8-15.

Figure 8-15 Effective VD Report Setup Dialog Box



9. Choose to save the `set_rail_voltage` or `set_voltage` commands in the output file.
10. Enter the name of the output file to be generated. Click Hide to close this dialog box and return to the Display Map dialog box.
11. In the Display Map dialog box, click Report at the top of the dialog box.

    The output effective voltage drop report will be saved to the working directory.

When a combined rail analysis has been performed

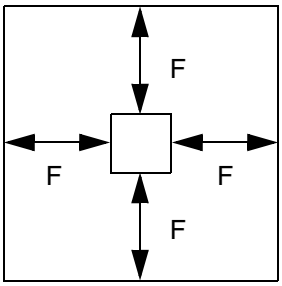
When performing a combined rail analysis, PrimeRail is able to generate the effective voltage drop waveform file during rail analysis. To do this, specify the name of the effective voltage drop waveform file to be generated in the Report Options dialog box through the `poRailAnalysis` command. (See Figure 7-12 on page 7-32.) Then in the Miscellaneous section, choose to use the timing window to calculate effective voltage drops. When rail analysis is complete, run the `pgMap` command to display the effective drop waveforms (see Step 7 in the previous section).

Reporting Via Coverage

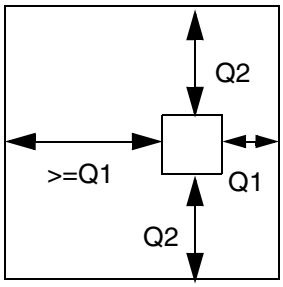
When cell-level rail analysis is complete, run the `pgResultPostProcess` command to write the information of via current and direction to an output file.

Doing via coverage is to classify how different vias should be handled in reliability verification. Coverage is for the top layer and landing is for the bottom layer of a via cut.

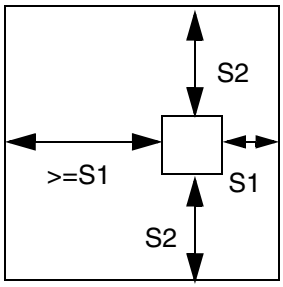
Four rules are available—Full, Quarter, Semi and Partial.



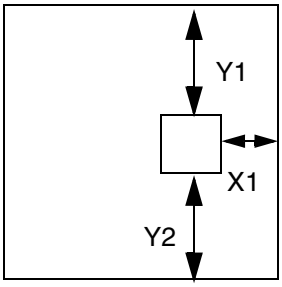
Full: All sides of the via are expanded with a parameter (F).



Quarter: One side of the via is expanded with a parameter (>= Q1). Both adjacent sides are expanded with a parameter (Q2).
Note: Q1 < Q2 < F



Semi: One side of the via is expanded with a parameter (>= S1). Both adjacent sides are expanded with a parameter (S2).
Note: S1 < S2 < Q2



Partial: The coverage or landing does not meet the conditions of Full, Quarter, or Semi.

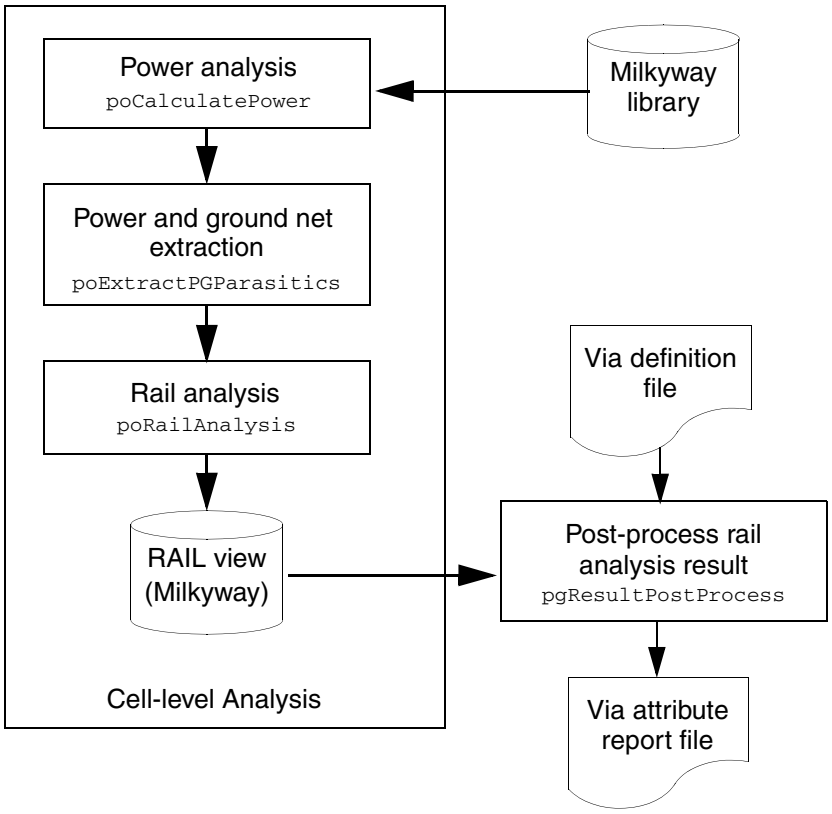
Note:

If the vias are identified as nodes, not resistors, the tool does not report such vias in the output file.

Via Coverage Analysis Flow

Figure 8-16 illustrates the steps of applying via coverage rules to a design.

Figure 8-16 Via Coverage Analysis Flow



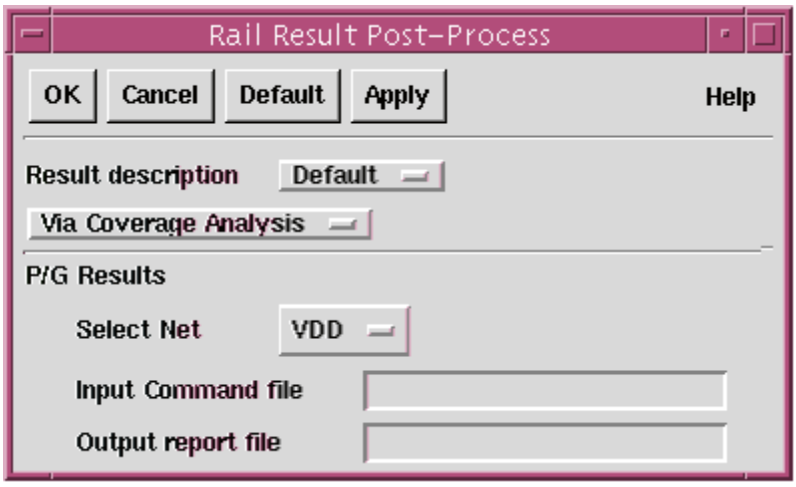
Generating Via Attribute Reports

When rail analysis is complete, run the `pgResultPostProcess` command to generate via attribute reports for a design.

To generate via attribute reports,

1. Enter `pgResultPostProcess` in the command window.
2. In the Post Result Process dialog box that opens, select Via Coverage Analysis from the pull-down list. The tool displays the related options in the lower section of the dialog box.

Figure 8-17 Postprocessing Rail Results for Via Coverage Analysis



3. Choose the name of the net whose analysis results are to be processed.
4. Enter the name of the input command file that contains the via coverage definition.

   The following is the syntax of the via coverage definition:

```
<VIA LAYER> {Xrange = x1,x2; Yrange = y1,y2 ;
               Landing = <FL> <QL2> <QL1> <SL2> <SL1>;
               Coverage = <FC> <QC2> <QC1> <SC2> <SC1>
            }
```

Table 8-3 Arguments for the Via Coverage Definition in Input Command File

Argument	Description
Xrange and YRange	The length and width range of the via
L / C	Landing / Coverage
<FL>/<FC>	Full landing / Full coverage
<QL2>/<QC2>	Quarter landing / Quarter coverage
<SL2>/<SC2>, <SL1>/<SC1>	Semi landing / Semi coverage

   Note that all coverage and landing parameters are positive integers in nanometers.
5. Enter the name of the via attribute file to be generated.
6. Click OK.

   The via attribute file is saved to the working directory when the result process is done.

Via Attribute File Syntax

Here is the syntax of the output via attribute file.

```
<via res ID> <layer name> <via count> <via area> <left> <bottom> <right>
<top>  <Iavg> <Iavg_dir>  <via code> <adjacent metal layer resistors>
```

Via Attribute File Example

```
*************************************************************************
*
* Title : PG Net Via Attributes Report
* Date  : Mon Oct  8 02:11:19 2007

*
* Vendor : Synopsys
* Program : PrimeRail
* Version : A-2007.12
*
* Design : circuit1
* Temperature : 125.0 C
*
* Units:
* Coordinate Unit : um
* Current Unit : A
* Area Unit : um^2
* Total number of vias : 922
* Scaling Factor : 1.000
*
*************************************************************************
*************************************************************************
** *VIA_COVERAGE_CODES

** *index  FF FQ FS FP QF QQ QS QP SF SQ SS SP PF PQ PS PP   X_by_Y
** * 0  0   0  0  0  0  0  0  0  0  0  0  0  0  0  0
** * 1  0   6  0  0  0  0  0  0  0  0  0  0  0  0  0 6x1
** * 2  6   0  0  0  0  0  0  0  0  0  0  0  0  0  0 6x1



CELL: micro2 INST:  NET: VDD
NODE LAYER VIA VIA   LEFT    BOTTOM  RIGHT   TOP     CURRENT  DIR VIA
(METAL RES)
RES  NAME  CNT AREA                                              CODE
NAME
R54  VIA1   8  1.037 669.020 392.490 674.980 392.850 -2.12e-05 D   14
(R55 R59)
R59  VIA2   7  0.907 669.420 392.490 674.580 392.850  2.24e-05 U   13
(R57 R54)
```

9

Performing Transistor-Level Dynamic Analysis

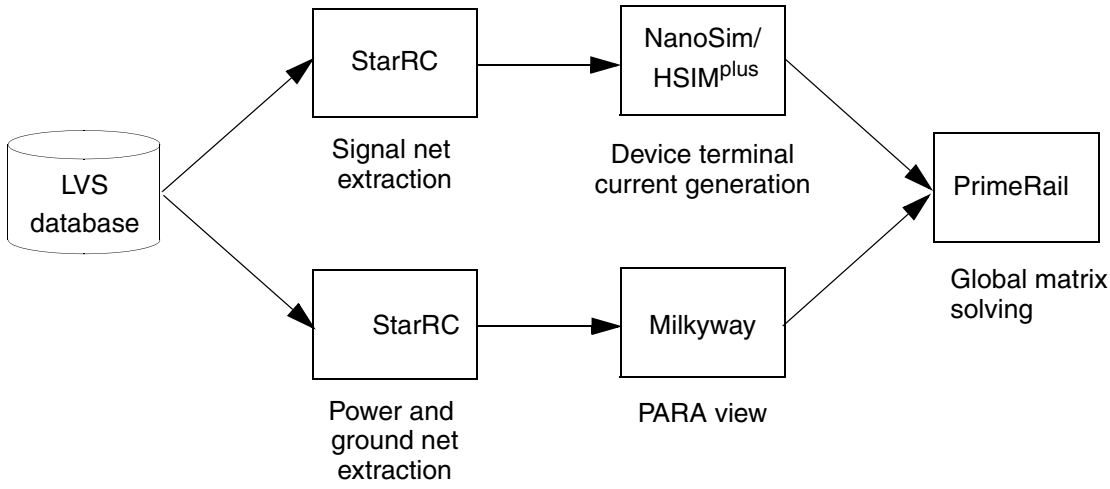
This chapter discusses how to use the PrimeRail transistor-level analysis capabilities to perform dynamic power and rail analysis at the transistor level.

This chapter contains the following sections:

Transistor-Level Analysis Flow

The PrimeRail transistor-level analysis capabilities heavily leverage the established Synopsys dynamic verification products—NanoSim and HSIM^{plus} for SPICE simulation, and StarRC for transistor-level extraction. When analyzing a block at the transistor level, PrimeRail drives the NanoSim (or HSIM^{plus}) and StarRC tools to execute the sequence of steps, as shown in Figure 9-1.

Figure 9-1 PrimeRail Transistor-Level Dynamic Flow



in Figure 9-1, PrimeRail drives StarRC to perform two parasitic extractions starting from the results of a clean layout-versus-schematic (LVS) run that is performed on the transistor-level block. PrimeRail supports LVS data from the Synopsys Hercules tool and the Calibre tool from Mentor Graphics Corporation.

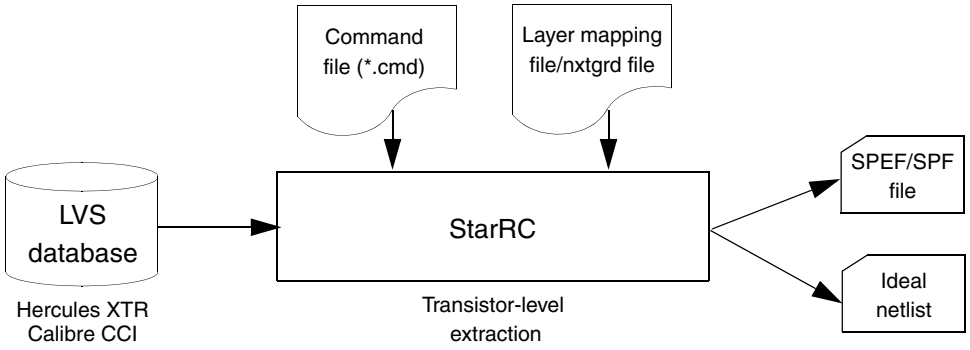
StarRC Transistor-Level Extraction Flow

Figure 9-2 shows the StarRC transistor-level extraction flow. Before extraction begins, you need to provide StarRC process characterization data in the form of a transistor-level extraction-compliant NXTGRD file and a layer mapping file. A skeleton StarRC command file is also required for configuring the type and disk location of the LVS database to be used and for defining the location of the NXTGRD and layer mapping files. PrimeRail augments the StarRC skeleton command file internally by adding command options that are specific to signal net and power net extraction. See the “Calibre Connectivity Interface” and the “Hercules Database Extraction Flow” sections in the *StarRC User Guide* for details on running transistor-level extraction.

PrimeRail imports the Standard Parasitic Exchange Format (SPEF) file that is generated from StarRC power and ground network extraction to a Milkyway PARA view. The PARA view will be used later in the voltage drop waveform simulation with the `poRailAnalysis` command.

The ideal SPICE netlist and DSPF files that are generated from the signal and power and ground net extraction serve as inputs to the next simulation stage for generating device terminal currents. The NanoSim and HSIM^{plus} simulation processes are depicted in Figure 9-3 and Figure 9-4.

Figure 9-2 StarRC Transistor-Level Extraction Flow

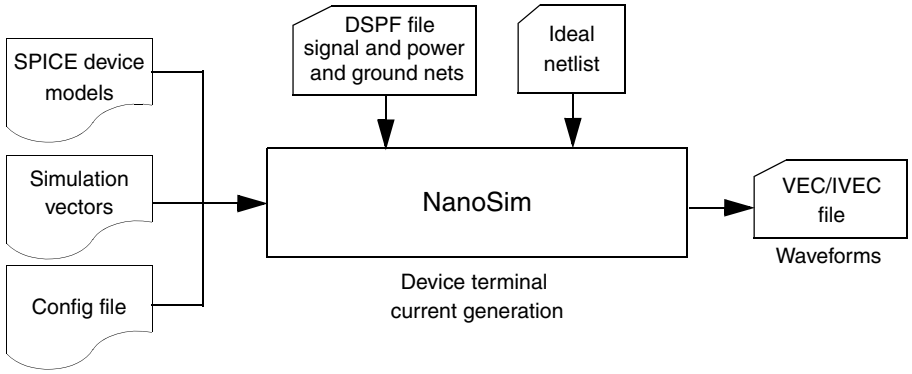


NanoSim Simulation Flow

Figure 9-3 shows the NanoSim simulation flow. You must provide the following data for PrimeRail to drive NanoSim successfully:

- Simulation vectors that define the behavior of the boundary ports for NanoSim
- Device models that are in the SPICE format to be used in the simulation
- NanoSim configuration file that configures specific simulation needs for the block being analyzed

Figure 9-3 NanoSim Simulation Flow



HSIM^{plus} Simulation Flow

PrimeRail allows you to perform top-level rail analysis using HSIM^{plus} to simulate transistor-level blocks within the design. This accurately solves for all necessary dynamic data (currents, voltages), which is crucial during top-level power and rail analysis.

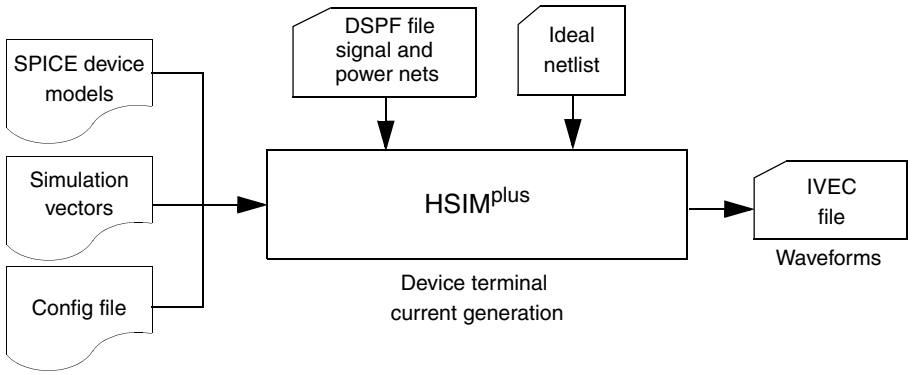
PrimeRail relies upon HSIM^{plus} for the transistor-level dynamic simulation results prior to solving the global matrix for top-level analysis.

The transistor-level designs are simulated using HSIM^{plus}, which solves for all device currents. These dynamic results are then fed back to the PrimeRail tool's top-level matrix solver for accurate top-level power and rail analysis results. PrimeRail provides the capability of analyzing designs containing mixed gate- and transistor-level blocks. The PrimeRail tool's transistor-level analysis relies on HSIM^{plus} for transistor-level SPICE simulation and StarRC for transistor-level device and parasitic extraction.

Figure 9-4 shows the HSIM^{plus} simulation flow. For optimum accuracy and performance with power and ground bus parasitics, you must provide the following files for PrimeRail to drive HSIM^{plus} successfully:

- Input vector stimulus/simulation vectors
- SPICE device models and the technology file
- HSIM^{plus} configuration files, including HSIM^{plus} commands and parameters that configure specific simulation needs for the block being analyzed

Figure 9-4 HSIM^{plus} Simulation Flow



The StarRC extraction process in PrimeRail generates the ideal input SPICE netlist and the extracted parasitic netlist (in the DSPF format) that contain both power and signal nets. HSIM^{plus} reads the data stored in these two files during the simulation process.

By default, PrimeRail supplies HSIM^{plus} with a Device List File (DLF) containing all devices connected to a particular power or ground net. HSIM^{plus} back-annotates all devices found in the DLF file and collects currents from devices that are connected to the net.

For more information about how PrimeRail interfaces with HSIM^{plus}, see the chapter on the HSIM^{plus}– PrimeRail interface in the *HSIM^{plus} Reference Manual*.

Preparing Input Files

Because PrimeRail employs the StarRC, NanoSim, and HSIM^{plus} technologies during the transistor-level dynamic analysis flow, you can use the same input files in PrimeRail as you use in running StarRC, NanoSim, and HSIM^{plus} tools. Some of the input files can be automatically generated through the `poTxPowerAnalysis` command, but some cannot. Each of the files should contain the necessary information for completing a successful StarRC, NanoSim, or HSIM^{plus} run. If any of the files are corrupted, the `poTxPowerAnalysis` command fails in executing the operation and cancels the process.

StarRC Extraction

To invoke the StarRC engine through the `poTxPowerAnalysis` command, you need to manually create the following files:

- StarRC command file
- StarRC mapping file
- StarRC GRD file

The following is an example of the StarRC command file based on the Hercules LVS database:

```
BLOCK:                              ADD4
MILKYWAY_DATABASE:                  ADD4_XREF
MILKYWAY_EXTRACT_VIEW:              YES
EXTRACTION:                         RC
XREF:                               YES
TCAD_GRD_FILE:                      xtor.nxtgrd
MAPPING_FILE:                       mapping.file
SKIP_CELLS:                         !*
NETLIST_FILE:                       add4.spef
NETLIST_FORMAT:                     SPEF
... ...
   EVACCESS_DIRECTORY:              /run_details/evaccess
   COMPARE_DIRECTORY:               /run_details/compare
```

See the “Calibre Connectivity Interface” and “Hercules Database Extraction Flow” sections in the *StarRC User Guide* for details on running transistor-level extraction from an LVS database.

For a detailed description about creating a command file, see the StarRC documentation.

NanoSim Simulation

For NanoSim, the files that you need to provide are

- NanoSim configuration file that contains a set of NanoSim commands
- Transistor model file
- File that contains information about input vectors

The following is a sample configuration file that can be used for a NanoSim simulation.

```
rm_dump_current VDD 0 80 0.05 prxtd_VDD.dlf ;
set_sim_eou sim=3 model=3 ;
use_sim_case 1  ;
set_diode_model * 1
```

PrimeRail requires the `rm_dump_current` command in the configuration file. If the configuration file you provide does not contain the `rm_dump_current` command, the `poTxPowerAnalysis` command automatically inserts it for each power and ground net.

```
rm_dump_current <start_Time> <end_time> <period> \
  [peak [= 0|1] compress[=0|1|2] format=0|1 ave=0|1]
```

Table 9-1 Argument Descriptions

Argument	Description
start_time	The time when the tool starts to save the transistor currents.
end_time	The time when the tool stops saving the transistor currents.
period	The length of time for saving transistor currents.
peak	Reports peak currents. Valid values: 1 – Reports peak currents 0 – Not to report peak currents (the default)

Table 9-1 Argument Descriptions (Continued)

Argument	Description
compress	Compresses the output file. Valid values: 0 – No compression (the default) 1 – Perform the UNIX compression. The output file is with the suffix ".Z". 2 – Perform the GZIP compression. The output file is with the suffix ".gz".
format	Creates the output file without saving the repeated values if they are not zero. Valid values: 1 – Keep the repeated values in the output file (the default) 2 – Remove the repeated values from the output file
ave	Saves the output file in the AVE format. Valid values: 0 – Not to write an AVE file 1 – Write an AVE file (the default)

When multiple output IVEC/VEC files exist in the directory, the GZIP-compressed file (.gz) has the precedence over the UNIX-compressed file (.Z), followed by the noncompressed file.

The tool automatically detects the compression format and processes the file later during the subsequent transistor-level dynamic rail analysis.

For a detailed description about creating a configuration file, see the NanoSim documentation.

HSIM^{plus} Simulation

For HSIM^{plus}, the files that you need to provide are

- Input vector stimulus/simulation vectors
- SPICE device models and the technology file

  PrimeRail will generate another SPICE input file for the top-level run (for example, PRTXRUN/prxtd_hsim.top) and add the necessary commands along with the original SPICE input file for the HSIM^{plus} simulation.

Do not include the ideal netlist and parasitic netlists in the original SPICE input file. HSIM^{plus} automatically adds these netlists based on the settings configured in the parasitic extraction section in the Tx Power Analysis dialog box (see Figure 9-7 on page 9-18). If these files are defined in the original SPICE input file, it might cause duplicate subckt definitions and incorrect back-annotation. A typical SPICE input file contains commands and options for circuit simulation, along with settings for input stimulus.

- HSIM^{plus} configuration files, including HSIM^{plus} commands and parameters

  You can define the HSIM^{plus} commands and parameters either in the input SPICE deck read into HSIM^{plus} or any file that is included within the input SPICE deck (.inc). HSIM^{plus} also reserves the hsim.ini file for defining any HSIM^{plus} parameters to be used during the simulation.

  In addition to the HSIM^{plus} commands and parameters, PrimeRail automatically adds the `HSIMPRIMERAIL` and `HSIMPRIMERAILTCL` parameters to activate HSIM^{plus} for generating IVEC files.

  - The HSIMPRIMERAIL Parameter

    The `HSIMPRIMERAIL` parameter is used to indicate whether or not the HSIM^{plus} and PrimeRail flow is activated.

    Syntax:

    ```
    .param HSIMPRIMERAIL=0|1
    ```

  - The HSIMPRIMERAILTCL Parameter

    The `HSIMPRIMERAILTCL` parameter is used to specify the name of the current vector waveform (IVEC) Tcl configuration file that is generated by HSIM^{plus} and subsequently passed to PrimeRail. PrimeRail also generates the IVEC CFG Tcl file and saves it as PRTXRUN/prxtd_hsim.cfg during transistor-level power analysis.

    Syntax:

    ```
    .param HSIMPRIMERAILTCL=<ivec_cfg_tcl_filename>
    ```

    The IVEC CFG Tcl file read in by `HSIMPRIMERAILTCL` supports the following syntax:

```
time -start <start_time> -stop <end_time> -tau \
<ivec_step_size>
net -name <net_name> -dlf <DLF_filename>
input -case <0|1>
output -pres <1|10> -type <0|1> -compress <0|1|2>
```

Table 9-2 Syntax Descriptions

Syntax	Description
time	start – Transient start time of current information written to *.ivec file (default: 0 ns) stop – Transient stop time of current information written to *.ivec file (default: stop time set by .tran statement) tau – Step size of currents written to *.ivec file (default: HSIMRATAU or 1ns) All times are defined in the unit of nanosecond.
net	name – Power net(s) to be analyzed (VDD, VSS, and so on) dlf – Power and ground netlist of connected device nodes (Device List File)
input	case – Automatically adjust case sensitivity to HSIM settings during IVEC generation. The possible values are 0 (case-insensitive) and 1 (case-sensitive). The default is 0.
output	pres – Print resolution time unit: 1 (default) = 1 ps; 10 = 10 ps type – Current type: 0 (max) or 1 (avg). The default is 1. compress – 0: No compression (default); 1:UNIX compress; 2:gzip

Note:

For the recommended, complete and minimal flows, HSIM^{plus} by default automatically adjusts all net names to match the case of the internal database settings. This is to account for running with case sensitive netlists in Spectre format where case-sensitivity is supported. The `-case` switch within the IVEC Tcl file is not necessary for these flows.

However, for the RA Re-use flow, HSIM^{plus} does not recall the case settings used during the simulation to generate the RA results. Therefore, you must provide HSIM^{plus} with the correct case switch in the input section of the IVEC CFG Tcl file. By default, the switch is set to case-insensitive (`-case 0`).

Example

An example of an IVEC CFG Tcl file that is passed to the `HSIMPRIMERAILTCL` parameter is provided below. The IVEC CFG Tcl file can only contain one time and one output section but can contain several net sections.

```
.param HSIMPRIMERAIL=1
.param HSIMPRIMERAILTCL=myfile.pr

myfile.pr:
----------
time -start 0 -stop 50 -tau 0.02
net -name VDD
input -case 0
output -pres 1 -type 1 -compress 1
```

The example file measures all average currents of devices connected to net VDD, starting at time 0 ns and ending at time 50 ns with a time step of 20 ps. The resulting *.ivec file will be compressed using UNIX compression.

For more information, see "Performing Transistor-Level Power Analysis With HSIM^{plus}" on page 9-16 or the chapter on the HSIM^{plus} and PrimeRail interface in the *HSIM^{plus} Reference Manual*.

Transistor-Level Power Analysis

Run the `poTxPowerAnalysis` command to simulate the circuit based on the device netlist and to record the device currents in a binary file by invoking the StarRC extraction engine and the NanoSim/HSIM^{plus} circuit simulator. The PrimeRail matrix solver (`poRailAnalysis`) then simulates the circuit by using the waveforms from NanoSim.

The transistor-level simulation flow in PrimeRail consists of the following three processes:

- Extraction process – Perform the StarRC transistor-level extraction for the signal nets of the circuit as well as the power and ground network.
- Simulation process – Perform the NanoSim or HSIM^{plus} simulation using the results from signal net extraction and derive the current waveforms for all device terminals connected to the power supply network.

- PARA view creation process – Import the resulting SPEF file from power and ground network extraction as well as the VEC/IVEC file resulting from NanoSim or the IVEC file resulting from HSIM^{plus} to a Milkyway PARA view. The tool accesses the PARA view for data during the subsequent dynamic power grid simulation.

You can execute these processes together in a single run or individually, depending on the settings given in the Tx Power Analysis dialog box (`poTxPowerAnalysis`), as shown in Figure 9-5.

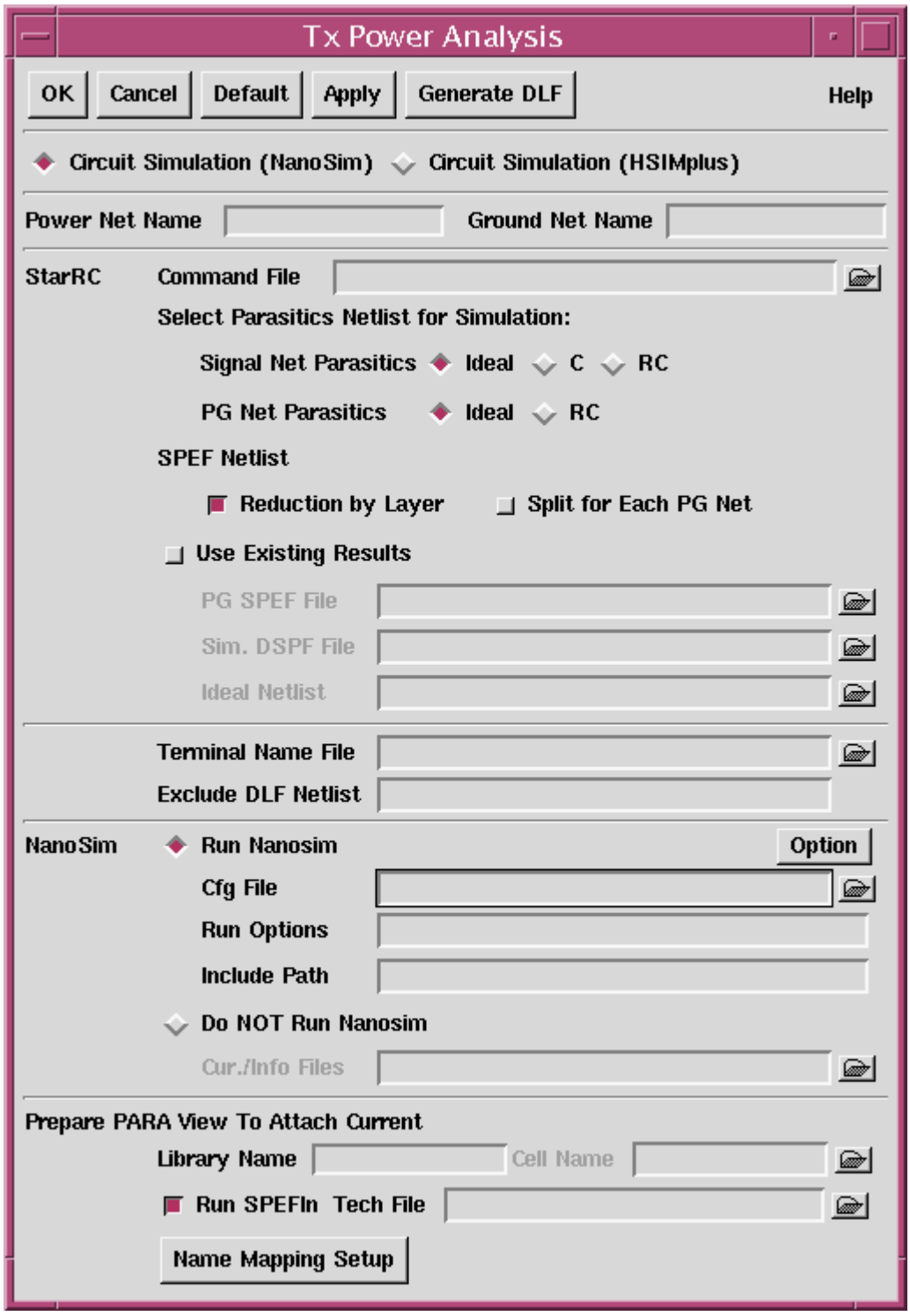
Performing Transistor-Level Power Analysis With NanoSim

This section describes the steps involved in capturing transistor currents with the `poTxPowerAnalysis` command.

1. Enter `poTxPowerAnalysis` in the command window or choose Tx-level Dynamic Analysis > Tx-level Analysis – Tx-level Power Analysis.

   The Tx Power Analysis dialog box appears.

Figure 9-5 Tx Power Analysis Dialog Box – NanoSim



2. Select NanoSim as the simulator used to run this simulation.
3. Enter the name of the power or ground net to be analyzed. You must enter at least one power or ground net. If you want to analyze multiple nets, separate them with a space. Wildcard (?*) usage is supported.
4. In the StarRC section, configure the options for the extraction process.
   - Command File – Enter the name of the skeleton command file for configuring StarRC extraction. You must provide a command file to be used. Otherwise, the StarRC extraction process will fail.

     For an example of the command file, see “Preparing Input Files” on page 9-5.
   - Select Parasitics Netlist for Simulation – Specify one of the following options:

     Signal Net Parasitics – Specify the extraction to be performed on the signal nets. The DSPF file resulting from this extraction is used in the subsequent NanoSim simulation process.

     Use the Ideal (the default) mode to extract only the connectivity information of signal nets. This mode is useful for calculating average waveforms. When selected, the tool generates an ideal netlist.

     Use the C mode to extract only the capacitance data of signal nets. When selected, the tool generates an ideal netlist and a Detailed Standard Parasitic Format (DSPF) file with signal capacitance data only.

     Use the RC mode to Extracts the connectivity information and the parasitic data (including resistance and capacitance) of signal nets. This mode is useful for calculating the parasitic effects of signal nets. When selected, the tool generates an ideal netlist and a DSPF file with signal parasitics data.

     PG Net Parasitics – Specify the extraction to be performed on the power and ground nets. Additional parasitic data of the power and ground nets is included in the DSPF file.

     Use the Ideal (the default) mode to extract only the connectivity information of power and ground nets. This mode is useful for calculating average waveforms.

     Use the RC mode to extract the parasitic data, including capacitance and resistance, of power and ground nets.
   - Specify the SPEF netlist – Configure additional behavior of the power and ground net extraction. PrimeRail always generates a PG SPEF file with all the necessary geometry information. The SPEF file is used to create a Milkyway PARA view, no matter which option you select in the “Select Parasitics Netlist for Simulation” section in step 5.

Reduction by Layer – Instruct StarRC to be run with the geometry preserving netlist reduction. This enforces preserving the power and ground net geometry as needed for rail analysis while still reducing the size of the extracted netlist as much as possible.

Split for Each PG Net – Instructs StarRC to generate one SPEF file for each power and ground net.

- Use Existing Results – Skip generating the files that already exist and generates only those that are missing if you are rerunning the simulation. Otherwise, PrimeRail automatically invokes the StarRC extraction engine and generates all the netlist files based on the extraction mode you choose.

  PG SPEF File – Enter the name of the SPEF file that already exists. As a result, the power and ground network extraction is skipped.

  Sim. DSPF File – Enter the name of the DSPF file that already exists. As a result, the signal net extraction is skipped.

  Ideal Netlist – Enter the name of the ideal netlist that already exists. As a result, the signal net extraction is skipped.

5. Enter the name of the terminal name file. Based on the source of the various input data to the transistor-level extraction flow, the device terminals listed in the VEC/IVEC file might be named differently than those in the PARA view. If so, by default it leads to name mismatching between the VEC/IVEC file and the PARA view, which prohibits subsequent rail analysis.

   Use the terminal name file to resolve name mismatching of the device terminal names.

   The tool supports two sets of device terminal naming conventions. It will first try the Hercules device terminal naming convention during name mapping; for example:

```
M DRN GATE SRC BULK
Q COLL BASE EMIT BULK
R A B
L A B
C A B
D ANODE CATHODE
```

   If the name mapping fails, PrimeRail will try the second naming convention; for example:

```
M: D G S B
Q: C B E S
R: POS NEG
L: POS NEG
C: POS NEG
D: POS NEG
```

   The second naming convention is used in the StarRC CCI flow. If name mapping fails again, you must specify the terminal device names in the device terminal file.

6. Enter the name of the device list file (DLF) to be discarded during the analysis flow. The device list file contains the information that PrimeRail uses to identify the connected transistors before analyzing the dynamic voltage drop or electromigration effects.

   For more information about the device list file, see “Name Mapping” on page 9-20.

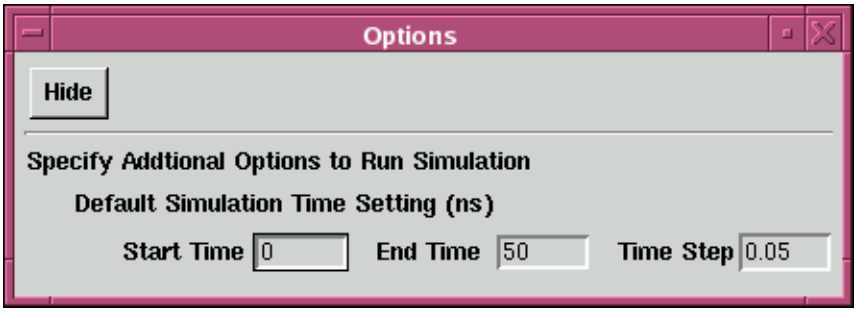
7. In the NanoSim section, select Run NanoSim and specify the necessary options for running a NanoSim simulation.

   - Cfg File – Enter the name of the configuration file that contains a set of NanoSim commands. You must provide a configuration file to run a transistor simulation.

     For an example of the configuration file, see “Preparing Input Files” on page 9-5.

   - Option – Click and specify the time window and resolution used for dumping the IVEC file. The Options dialog box appears, as shown in Figure 9-6.

Figure 9-6 Specifying Timing Windows During Transistor-Level Power Analysis



   Start Time – Enter the time to start the simulation. The default is 0 ns.

   End Time – Enter the time to stop the simulation. The default is 50 ns.

   Time Step – Enter the time step for the simulation. The default is 0.05.

   Click Hide to apply the changes you make and return to the Tx Power Analysis dialog box.

   The timing setting you specify will be added to the `rm_dump_current` command in the PRTXRUN/prxtd_nanosim.cfg file. For example,

   ```
   rm_dump_current VDD 0 50 0.05 PRXTRUN/prxtd_VDD.dlf
   ```

   - Run Options – Enter additional command-line options you want to pass to NanoSim for simulation.

   - Include Path – Specify the search path for NanoSim “include” statements.

   - Do NOT Run NanoSim – Use this option if you want to skip the simulation process. Use the Cur./Info Files text box to specify the VEC/IVEC file that includes the current waveforms to be used during rail analysis. If you have multiple files to specify, separate them with a space.

8. In the Prepare PARA View To Attach Current section, specify the necessary options for creating a Milkyway PARA view.

   - Library Name – Enter the name of the library that contains the PARA view to be created.
   - Cell Name – Enter the name of the PARA view to be created.
   - Run SPEFIn – Select to load a StarRC SPEF file from the power and ground net extraction process to create a Milkyway PARA view. The SPEF file can be from StarRC or you can specify one to be used.

     If the PARA view already exists and you do not want to update it, deselect this option. This is useful when you want to skip the process of loading a SPEF file.
   - Tech File – Enter the name of the Milkyway technology file (*.tf) that will be used to create a new library if the library given does not exist.

     If you do not have a preexisting Milkyway technology file, run `poTxCreateTechFile` to augment the creation of a Milkyway technology file. For a detailed procedure of creating a technology file, see the online Help for the command.

     During the PARA view creation process, PrimeRail imports the VEC/IVEC files—either generated by NanoSim or those you provide—to the Milkyway PARA view. This allows PrimeRail to access the data stored in the files later during transistor-level dynamic rail analysis.
   - Name Mapping Setup – Click to open the Name Mapping Setup dialog box, where you specify the settings for creating a name mapping file based on the information available in the database.

     For more information, see Figure 9-9 on page 9-22.
9. Click OK or Apply.

Performing Transistor-Level Power Analysis With HSIM^{plus}

PrimeRail relies upon HSIM^{plus} for the transistor-level dynamic simulation results prior to solving the global matrix for top-level analysis. To run HSIM^{plus} simulation in PrimeRail, two additional parameters (`HSIMPRIMERAIL` and `HSIMPRIMERAILTCL`) are required in the HSIM-RA (Reliability Analysis) Phase I simulation. If the parameters are not available, PrimeRail automatically adds the parameters to activate HSIM^{plus} for generating IVEC files. For a detailed description of the parameters, see “HSIM^{plus} Simulation” on page 9-7.

HSIM^{plus} supports the following flows when interfacing with PrimeRail:

- Recommended Flow: Runs the HSIM^{plus} simulation with back-annotation, compression, and reduction of power and ground net parasitics. Post-Layout Acceleration (PLX) is optional.

  This flow provides users with the available post-layout feature of power net reliability analysis (PWRA) power net compression and reduction within HSIM^{plus} without having to run a complete HSIM^{plus} power net reliability analysis (PWRA). HSIM^{plus} back-annotates and simulates with the power nets in place, arriving at much more accurate current waveform results.

  The HSIM^{plus} PWRA parameter `HSIMSPFPWNET` controls the level of power net reduction performed on the power net parasitics. When set, this parameter checks out the hsim-pra license.

- Complete Flow (Optional): Runs the HSIM^{plus} simulation with HSIM^{plus} power net reliability analysis (PWRA) and PLX in addition to the recommended flow.

  In addition to the recommended flow above, the complete flow takes advantage of the HSIM^{plus} PLX feature and also runs phase 1 of HSIM^{plus} PWRA for storing all necessary results to run HSIM^{plus} reliability analysis.

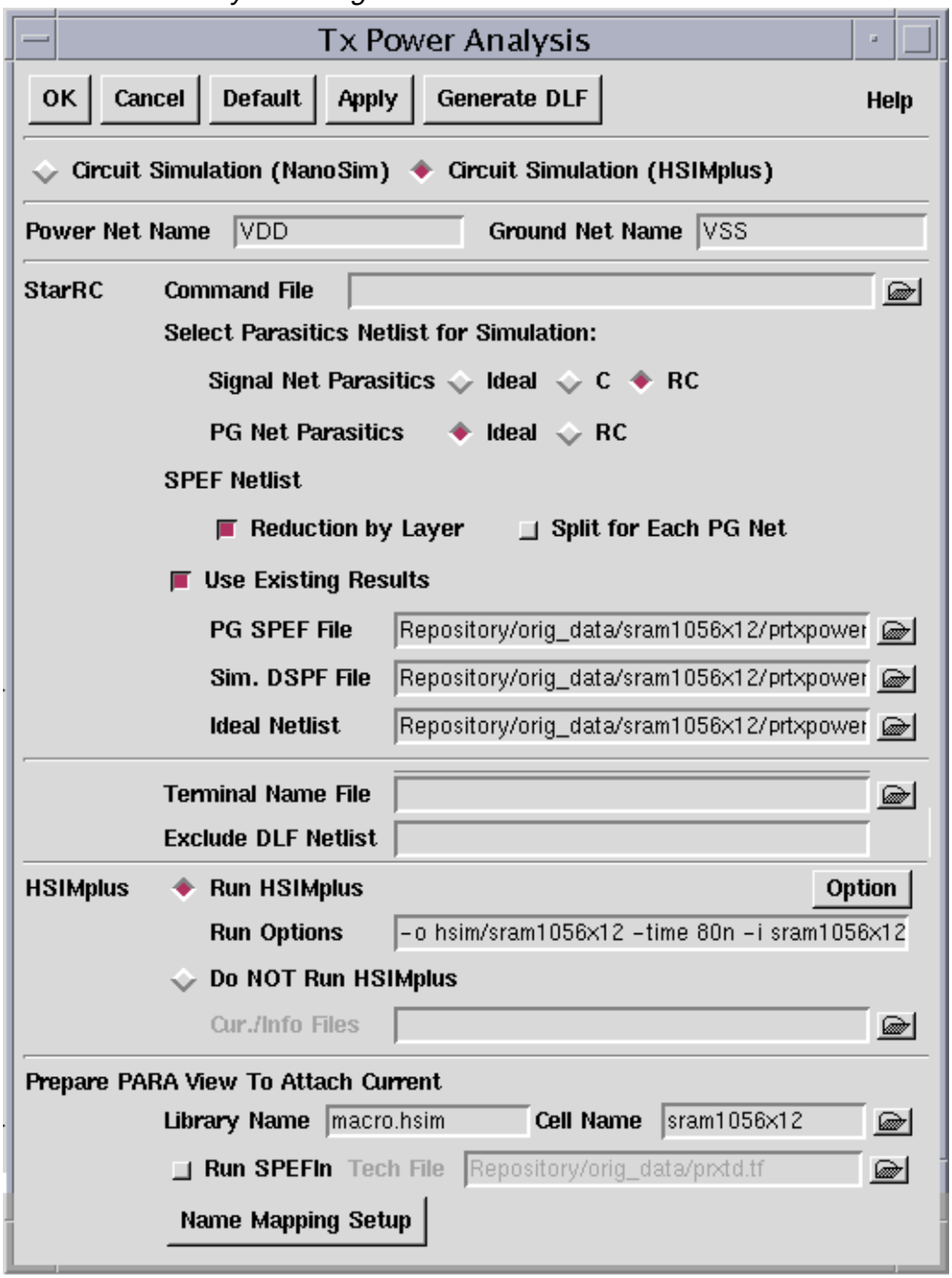
- Minimal Flow: Runs the HSIM^{plus} standalone simulation with no back-annotation, compression, or reduction of power and ground net parasitics. Device List files (DLF) are required for power nets whose currents are desired, because they are not back-annotated from the extracted DSPF netlist in this flow.

  Use the minimal flow if you are not familiar with and/or do not have access to HSIM^{plus} power reliability analysis features, such as, `HSIMSPFPWNET`, for power net reduction. In the minimal flow, power net parasitics are not back-annotated onto the simulation netlist and DLF files are mandatory for any net defined within the IVEC Tcl file. DLF files and IVEC Tcl file are generated by PrimeRail during the transistor-level power analysis command.

The following section describes the steps involved in capturing transistor currents with the `poTxPowerAnalysis` command.

1. Enter `poTxPowerAnalysis` in the command window or choose Tx-level Dynamic Analysis > Tx-level Analysis – Tx-level Power Analysis to open the Tx Power Analysis dialog box.
2. In the Tx Power Analysis dialog box, select Circuit Simulation (HSIMplus). The options related to the HSIM^{plus} simulation are displayed in the dialog box, as shown in Figure 9-7.

Figure 9-7 Tx Power Analysis Dialog Box – HSIM^{plus}



3. Enter the name of the power or ground net to be analyzed. You must enter at least one power or ground net. If you want to analyze multiple nets, separate them with a space. Wildcard (?*) usage is supported.

4. In the StarRC section, specify the options needed to run StarRC extraction. For a detailed description of the options related to StarRC simulation, see Step 4 on page 9-13.

5. In the HSIM^{plus} section, select Run HSIM^{plus} and specify the necessary options for running an HSIM^{plus} simulation.

   - Run Options – Enter additional command-line options you want to pass to HSIM^{plus} for simulation. The following is an example of the syntax:

     ```
     -o hsim/sram1056x12 -time 80n -i sram1056x12_top.sp
     ```

     In the above example, HSIM^{plus} will use the sram1056x12_top.sp file as the SPICE input file and use hsim/sram1056x12 as the prefix for all the output files. The simulation will be continued for 80 nanoseconds.

   - Option – Click and specify the time window and resolution used for dumping the IVEC file. The Options dialog box appears (see "Specifying Timing Windows During Transistor-Level Power Analysis" on page 9-15).

     Start Time – Enter the time to start the simulation. The default is 0 ns.

     End Time – Enter the time to stop the simulation. The default is 50 ns.

     Time Step – Enter the time step for the simulation. The default is 0.05.

     Click Hide to apply the changes you make and return to the Tx Power Analysis dialog box.

     The timing setting you specify will be added to the IVEC CFG file; for example:

     ```
     time -start 0 -stop 50 -tau 0.05
     ```

   - Do NOT Run HSIMplus – Use this option if you already have the pre-generated IVEC files and want to skip the simulation process. In the Cur./Info Files text box, specify the IVEC file that includes the current waveforms to be used during rail analysis. If you have multiple files to specify, separate them with a space.

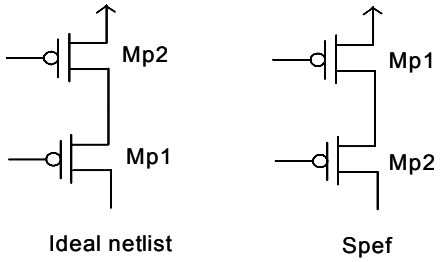
6. In the Prepare PARA View To Attach Current section, specify the necessary options for creating a Milkyway PARA view. For a detailed description about the options, see Step 8 on page 9-16.

7. Click OK or Apply.

Name Mapping

The process of name mapping is to back-annotate the names in the ideal netlist to those in the SPEF file. When running `poTxPowerAnalysis`, the tool automatically solves all possible name mapping issues between ideal netlists and SPEF files. However, more mapping failures are expected if your ideal netlist is not generated through the StarRC extraction process with the `poTxPowerAnalysis` command.

Run `poTxNameMapping` to quickly check for name mapping issues without running `poTxPowerAnalysis`. When the design being analyzed has swap devices, you need to prepare a Hercules CDB file that contains the information of swabbable devices and mappings between schematic and layout names. Be sure that the syntax in the CDB file defines the “matched” clause for the “device” type with “composite.” For more information about the Hercules CDB file, please see the Hercules documentation.



Swap Devices

When the name mapping process is done with the `poTxNameMapping` command, two output files will be saved to the PRTXRUN directory; they are NM debug file (_nm.fail) and NM reference file (_ref.nm).

NM Debug File (_nm.fail)

PrimeRail writes the name mapping failure information to this file when name mapping process is complete. In this file, the ivecName keyword is the name of an instance or a device from the IVEC file and the “matched macro” keyword from the SPEF file. If name mapping fails the tool reports the substring of an instance name in the SPEF file to match with what ivecName is (that is, the name in the IVEC netlist).

For example, the following lines are printed in the NM debug file:

```
ivecName = XU8/XU7/XCKBXD16G1B7I309/D3_P
matched macro = XU8/XU7/XCKBXD16G1B7I309
child instances: MU21_0 MU21_1
```

This means that no D3 child instance or device is found in the SPEF file. Only MU21_0 and MU21_1 are in the SPEF file.

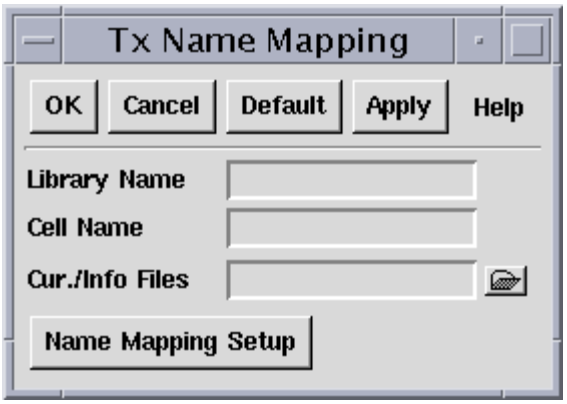
NM Reference File (_ref.nm)

PrimeRail saves the matched IVEC and HN name pairs in this file.

To perform name mapping,

1. Enter `poTxNameMapping` to open the Tx Name Mapping dialog box, as shown in Figure 9-8.

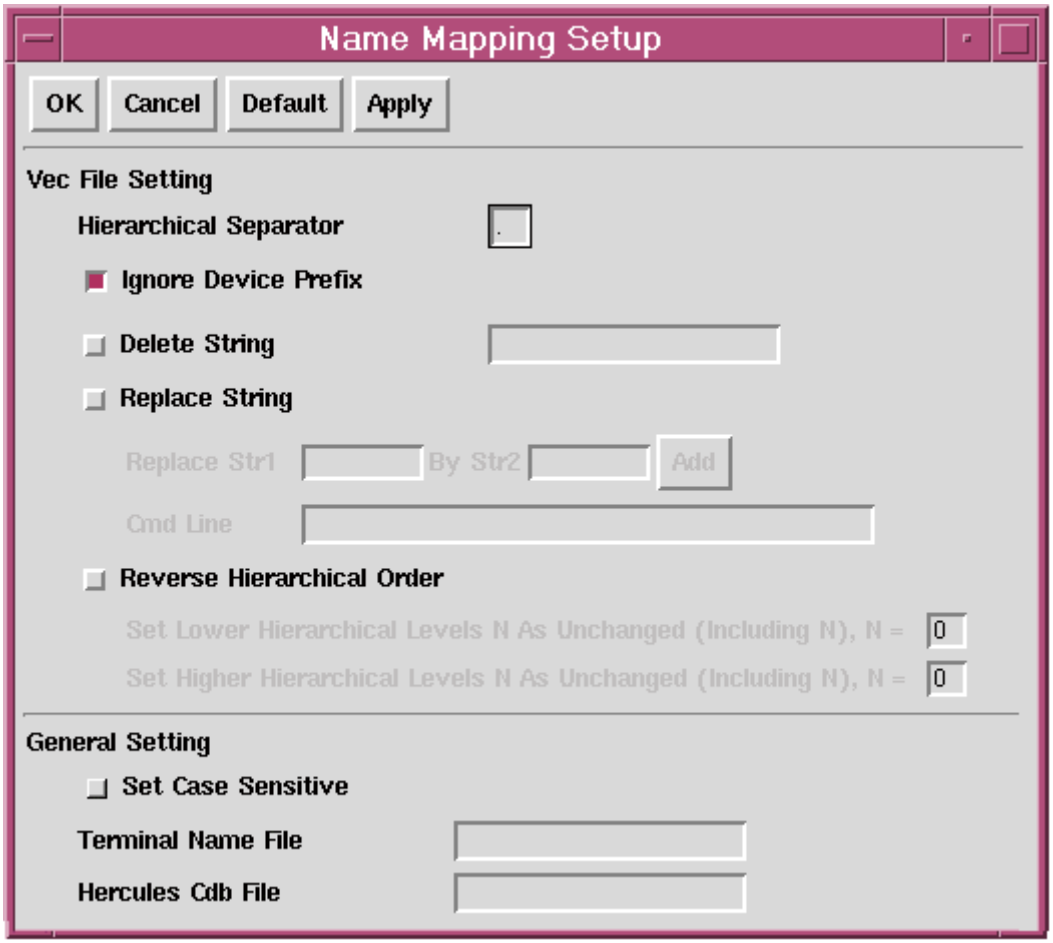
Figure 9-8 Tx Name Mapping Dialog Box



2. Enter the name of the library and cell being analyzed.
3. Enter the name of the file that contains the current information.
4. Click Name Mapping Setup to configure additional settings.

   The Name Mapping Setup dialog box opens, as shown in Figure 9-9.

Figure 9-9 Name Mapping Setup Dialog Box



- In the Vec File Setting section,

  Hierarchical Separator – Enter the hierarchy separator used in the VEC/IVEC file. The default is a period (.).

  Ignore Device Prefix – NanoSim adds a prefix to the name of the device node during the simulation. You can choose to ignore the prefix so that the nodes are named consistently in the VEC/IVEC file and the PARA view.

  For example, NanoSim prefixes the character X to the name of node X119, so the node is named XX119 in the VEC/IVEC file. However, the node continues to be named X119 in the PARA view.

  Delete String – Enter any element names you want to remove from the PARA view.

  Replace String – Enter the element name to be replaced in the Str1 text box and assign a new name in the Str2 text box. Click Add to add the strings you enter in the Cmd Line box.

Reverse Hierarchical Order – Reverse the hierarchical order of element names saved in the PARA view.

Note that when the Delete String, Replace String, and Reverse Hierarchical Order options are selected, the Delete String option has precedence over the other two options, followed by Replace String and then Reverse Hierarchical Order.

- In the General Setting section,

  Set Case Sensitive – Select if the VEC/IVEC file and the PARA view are to be treated as case-sensitive. The default is off.

  Terminal Name File – Enter the name of the terminal name file that is used to resolve the name mismatching of the device terminal names. This is useful if the device terminals listed in the NanoSim VEC/IVEC file are named differently than those in the PARA view.

  Hercules Cdb file – Enter the name of the Hercules CDB file. You need to provide a CDB file if the design being analyzed contains swap devices.

  Click OK to apply the changes and close the Name Mapping Setup dialog box.

5. In the Name Mapping dialog box, click OK or Apply.

Generating Device List Files

By default, PrimeRail automatically generates a device list file for each power and ground net based on the ideal SPICE netlist for circuit simulation. PrimeRail uses this device list file to identify the connected transistors before analyzing the dynamic voltage drop or electromigration effects.

The following is an example of the device list file (*.dlf):

```
*_NET VDD
XI8.XI18.X7.M0 D
XI8.XI18.X7.M1 S
XI8.XI18.X7.M4 G
XI8.XI18.X6.M5 S
```

Alternatively, if you want to generate the device list file without completing the entire transistor-level analysis flow, click the Generate DLF button at the top of the Tx Power Analysis dialog box (see Figure 9-5 on page 9-12).

Checking Calculated Current Waveforms

When you do transistor-level power analysis, you can save the calculated current waveforms at the transistor level to an output FSDB file. This FSDB current waveform file not only stores the current waveforms in the same way that the IVEC/VEC file does, but it also contains the information when there are multiple currents connected to one port.

This FSDB current waveform file also supports the threshold usage. The threshold used in the FSDB file is equal to 1^{-4} voltage threshold (in millivolts), as defined in the Rail Analysis dialog box.

To save current waveforms for each device terminal or instance port in the FSDB format, enter the following in the command window:

```
poDumpTxPowerDBToFsdb "vdd.fsdb" "VDD"
```

where *vdd.fsdb* is the name of the output FSDB file to be created, and *VDD* is the name of the net being analyzed.

Transistor-Level Rail Analysis

Use the `poRailAnalysis` command to perform a dynamic rail analysis at the transistor level.

Before you conduct a transistor-level dynamic rail analysis, make sure the Milkyway PARA view generated by the `poTxPowerAnalysis` command is open. The rail analysis reads the power and ground net extraction and current waveform data from the PARA view and assembles the global RC tree loaded by the given current waveforms. The `poRailAnalysis` command then simulates the resulting RC tree according to the time window and time step you specify. The tool captures the voltage waveforms in either FSDB or WDB format, and you can view the waveforms in a standard waveform viewer.

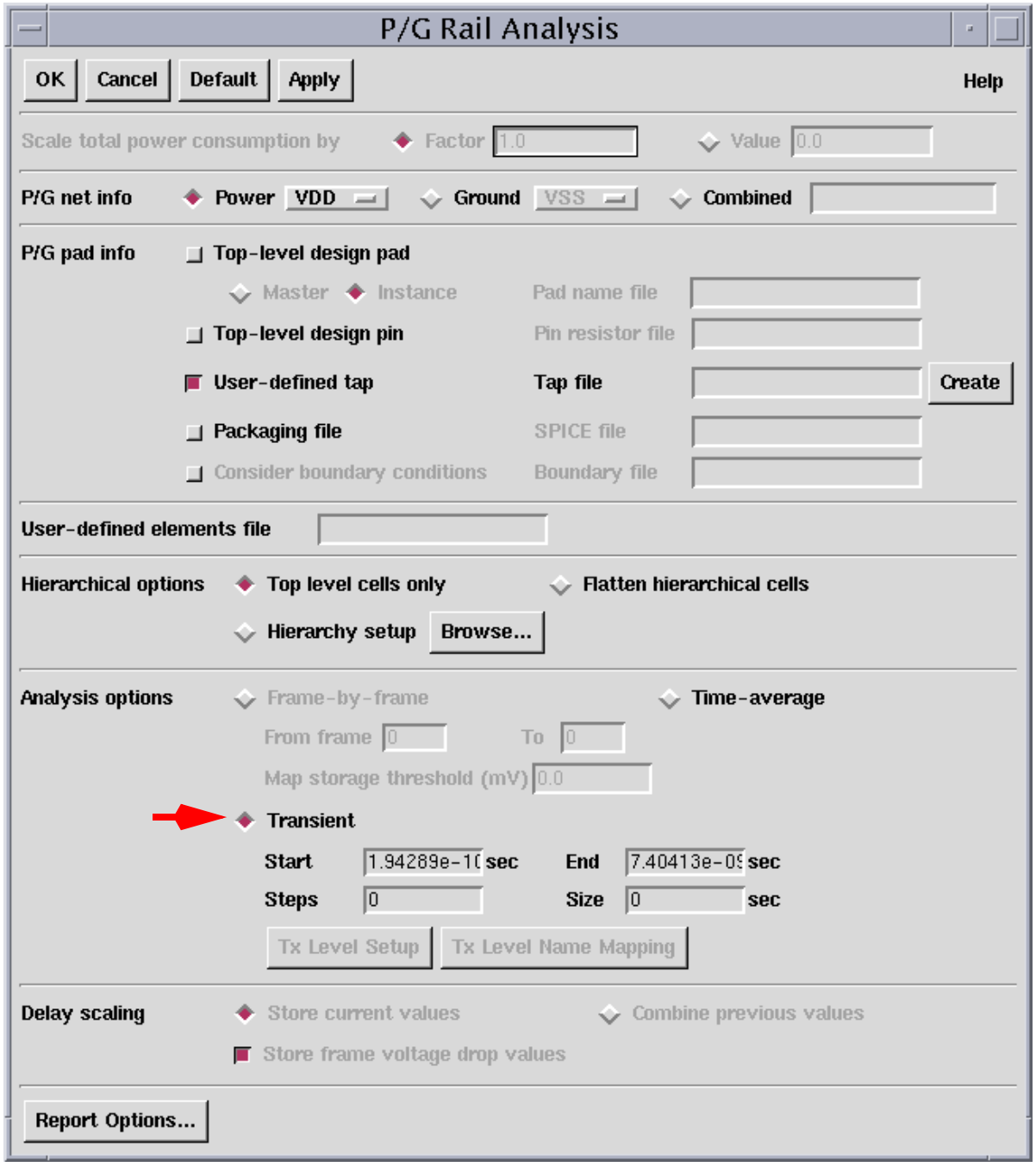
Because the `poRailAnalysis` command is shared with cell-level dynamic and time-average analysis flows, not all the command options are applicable in the transistor-level dynamic rail analysis.

To simulate the transistor network,

1. Enter `poRailAnalysis` or choose Tx-level Dynamic Analysis > Tx-level Analysis – Rail Analysis.

   The P/G Rail Analysis dialog box appears, as shown in Figure 9-10.

Figure 9-10 P/G Rail Analysis Dialog Box—Transistor-Level Analysis



2. In the “P/G net info” section, select the type of net to be analyzed. Select the name of the net from the menu.

3. In the “P/G pad info” section, specify how the power is to be supplied to the chip from the off-chip components of the system.

   - Top-level design pad – The voltage sources can be top-level pad cells; the information is provided as the pad cell instance names or master names in an ASCII file.
   - Top-level design pin – Pins from the CEL or FRAM view are used as the voltage source.
   - User-defined tap – Specify the voltage source location through tap points. To create a tap file, click the Create button. The tap file contains power and ground net names, coordinates, and layer numbers on which the voltage source is used.

     The user-defined tap file also allows for more complex modeling usage. For more information, see online Help for the `poGenUserDefineTap` command or “User-Defined Taps” on page 7-25.
   - Packaging file – Specify the SPICE file containing packaging parasitics that define the ideal voltage sources to be used during rail analysis.

     For more information, see “Including Package Parasitics” on page 12-3.

   You can use some of the options, such as Top-Level Design Pin and User-Defined Tap, in combination to investigate whether the rail analysis results can be improved in the existing design. For a standard flow, select only the Top-Level Design Pin option and deselect User-Defined Tap.

   For a detailed description about specifying ideal voltage sources, see “Ideal Voltage Source Locations for Rail Analysis” on page 7-23.

4. User-defined elements file – Enter the name of the file that includes user-defined structures consisting of virtual capacitors or resistors. PrimeRail will include these data in the simulation of the transistor-level block for nonstandard design schemes.

   Use this option when you are conducting a what-if analysis. For the transistor-level analysis flow, PrimeRail supports inserting non-physical capacitors in the design.

   For more information about what-if analysis, see “What-If Analysis” on page 12-21.

5. Analysis options – Select Transient to conduct a transistor-level analysis.

   Start / End – You can have the tool write out the currents of a vector device for a specific subwindow of peak current values and peak time, captured by the NanoSim simulation engine. To do this, specify the start and end time for which the currents are to be saved.

   PrimeRail will execute a sanity check on any timing window given before entering the simulation. If the time you specify exceeds the time range from the VEC/IVEC file, the command returns an error message. If any time windows overlap with each other, PrimeRail automatically merges the overlapped time windows for peak currents.

   Steps – Enter a value in the Steps box for the time steps in which the simulation is to be run.

Size – Enter a value in the Size box for the step size, in nanoseconds (ns), to be used during power grid simulation.

Note:
By default, the Tx Level Setup and Tx Level Name Mapping buttons are not available. To enable them, set the `poTxOldDB` switch before invoking the `poRailAnalysis` command.

6. Report Options – Click to specify the necessary options in the Report Options dialog box that opens.

   For a description about the options on this dialog box, see Figure 7-12 on page 7-32.

7. In the P/G Rail Analysis dialog box, click OK or Apply.

   When analysis is complete, you can view the voltage drop and electromigration violations by using the `pgMap` command.

   For more information about viewing transistor-level dynamic analysis results, see "Viewing Transistor-Level Dynamic Analysis Results" on page 9-29.

Log Message

The following is a sample log message from the transistor-level dynamic rail analysis:

```
Reading user defined pad location under VDD ...
Open Tx-level current waveform file for net VDD
HN Summary:
  total cells: 50
  total cell instances: 929
  total macro cells: 48 0 (model)
  total macro cell instances: 103 0 (model)
  total nets: 2
  total ports: 1030
  total signals: 1032
  hierarchy depth: 6

Loading PG Net VDD ...

Load Parasitic Data  Memory: 86.355 MB  CPU: 0 seconds  Elapse: 0 seconds

 mpu VDD
RAIL BC: EE_6_264400_113100 (3) touches block 0
RAIL BC: (4) top block pin VDD (0 362.600000 113.000000) is added as
boundary
condition
RAIL BC: (5) top block pin VDD (0 0.000000 254.000000) is added as
boundary
condition
RAIL BC: (6) top block pin VDD (0 0.000000 174.800000) is added as
boundary
condition
RAIL BC: (7) top block pin VDD (0 0.000000 114.100000) is added as
```

```
boundary
condition
RAIL BC: (8) top block pin VDD (0 0.000000 38.200000) is added as
boundary
condition
RAIL BC: (9) top block pin VDD (0 362.600000 186.800000) is added as
boundary
condition
RAIL Analysis: adding top design pins as boundary condition

Setup Boundary Conditions  Memory: 86.355 MB  CPU: 0 seconds  Elapse: 0
seconds


Setup Current  Memory: 86.355 MB  CPU: 0 seconds  Elapse: 0 seconds

RAIL Analysis: create waveform file
There is no pg.spec in library DESIGN4.
Fail to read reference library PG Spec files
WARNING : Fail to initialize library characterization data. DECAP cells
will
be decoupled between nets.

** Capacitance report ******************************
Total core capacitance 2.105e-3 nf

Setup Compressed Matrix, number of equations 5137 (0) ...
RECYCLE 212992 bytes of array memory
Matrix solver: Total number of elements 6156
RECYCLE 24576 bytes of array heap memory
RECYCLE 73728 bytes of array heap memory
RECYCLE 24576 bytes of array heap memory
RECYCLE 73728 bytes of array heap memory
RECYCLE 24576 bytes of array heap memory
Matrix solver: Total capacitance value: 2.105e-3 nf

Setup Matrix  Memory: 86.355 MB  CPU: 0 seconds  Elapse: 0 seconds

Transient Analysis 0: start 4.05e3 (ps), end 50e3 (ps), step size 50
(ps),
steps 919
 reference time: 0 (s)

One Step Evaluation  Memory: 86.355 MB  CPU: 0 seconds  Elapse: 1 seconds

Begin Transient Analysis ...
 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Done

Power domain VDD network voltage report

PEAK transient voltage:
    Min voltage rise (mV) = 0
```

```
    Max voltage rise (mV) = 0
    Min voltage drop (mV) = 0
    Max voltage drop (mV) = 6.364e3
AVG transient voltage:
    Min voltage drop (mV) = 0
    Max voltage drop (mV) = 558.2

Power domain VDD instance voltage report

Top 5 PEAK transient voltage drop values at instance ports:
#1: 6.364e3 (mV) at XBUS_CTRL/XMEM_DRV2/XI4\MP1@2/DRN (0_402) (Dev1) time
49.875e-9
#2: 6.364e3 (mV) at XBUS_CTRL/XMEM_DRV2/XI6\MP5@2/DRN (0_349) (Dev1) time
49.875e-9
#3: 6.326e3 (mV) at XBUS_CTRL/XMEM_DRV1/XI4\MP1/DRN (0_18) (Dev1) time
49.875e-9
#4: 6.326e3 (mV) at XBUS_CTRL/XMEM_DRV1/XI6\MP5@2/DRN (0_385) (Dev1) time
49.875e-9
#5: 6.25e3 (mV) at XBUS_CTRL/XMEM_DRV2/MP3/DRN (0_132) (Dev1) time
49.875e-9
Analysis Result File: Default:Default:VDD, size: 250559 Bytes
RAIL DB: save waveform file /remote/na1v0home/davidtg/rail_qa/tutorial/
tx/
mpu/DESIGN4/RAIL/mpu:1_11
RAIL Analysis: link waveform file /remote/na1v0home/davidtg/rail_qa/
tutorial/tx/mpu/DESIGN4/RAIL/mpu:1_11 to test2.fsdb

RAIL Analysis  Memory: 86.355 MB  CPU: 2 seconds  Elapse: 3 seconds

RECYCLE 180224 bytes of array memory
RECYCLE 376832 bytes of array memory
```

Viewing Transistor-Level Dynamic Analysis Results

When transistor-level dynamic analysis is complete, use the `pgMap` command to check for violations on voltage drop and current density, as described in the following sections:

- Displaying the Voltage Drop Map
- Viewing Dynamic Voltage Drop Waveforms
- Displaying the Electromigration Map
- Querying Voltage Drop and Current Density Values
- Reporting Voltage Drop and Current Density Violations
- Viewing Animated Voltage Drop Maps

Displaying the Voltage Drop Map

When dynamic power network simulation at the transistor level is complete, you can view the voltage drop violations by drawing a voltage drop map.

To display the voltage drop map of a net,

1. Enter `pgMap` or choose Tx-level Dynamic Analysis > Display – Display Tx-level Dynamic Maps.
2. In the Display Map dialog box that appears (see Figure 8-1 on page 8-3), select the analysis result you want to load from the Results menu. Click Load.
3. In the Map Options section, choose Voltage Drop from the menu. Specify either the average (AVG) or maximum (MAX) voltage waveforms to be displayed in the voltage drop map. PrimeRail displays the average waveforms by default.

   Overshoot – Display voltage overshoot and ground undershoot. Overshoots usually occur due to the inductance of a circuit. This option is applicable only to the voltage drop map.

   For example, when you want to display dynamic voltage waveforms for a net, select MAX to display the peak voltage drop and ground bounce of the waveform. Select Overshoot to display the peak voltage overshoot and ground undershoot of the waveform.

   - Map Configuration – Click to specify additional options for map display in the Map Configuration dialog box that appears.

     For a detailed description of the options in the Map Configuration dialog box, see Figure 8-7 on page 8-17.

   - Upper bound/Lower bound – Enter the values for the upper and lower voltage drop limits. The tool draws the voltages higher than the upper bound in red and disregards those lower than the lower bound.

     Auto-range – Use the maximum voltage as upper bound and the minimum voltage as lower bound.
4. In the Display Options section, specify the following:

   - Show Text – Display voltage drop or current density violations on the map by selecting one of the following options:

     Value – Show the voltage drop value as an absolute value. This option is on by default.

     Ratio – Show the voltage drop value as a ratio to a specified threshold.

     Density – This option is not applicable in displaying a voltage drop map.

   - Label – Display the design name, analysis type, net name, and key analysis characteristics in the map.

- Pads – Highlight the pads used as ideal voltage sources in rail analysis.
- Legend – Display the upper and lower bound values of the metal and via layers on the map. Click Select Layer to choose the layer whose upper and lower bound values are to be shown in the map.
- Violations In Red – Indicate violations in red on the map.
- Discard Bulk Layer – Ignore the bulk layer specified in the text box when displaying the voltage drop map.
- Peak Position In Blink – Show a blinking rectangle where the maximum value of voltage drop or current density occurs, based on the type of map to be displayed.
- With Solid Fill Pattern as Default – Draw the map with the shapes filled with a solid color.
- Floating Metals In Grey – Show floating metals on the map.

5. Specify other options in this dialog box as necessary.

   For a detailed description of the options in this dialog box, see online Help for the `pgMap` command.

6. Click Apply.

   The voltage drop map is drawn in the cell editing window. In the map, a device terminal is represented by a white cross box. Zoom in to examine a specific location in the voltage drop map.

Viewing Dynamic Voltage Drop Waveforms

When the voltage drop map is displayed, click the Waveform button at the top of the Display Map dialog box (or enter `pgWaveformQuery` in the command window) and click a device terminal in the voltage drop map. The dynamic waveform of the power and ground pin on the selected device is displayed in the Voltage Waveform Display window that appears (see Figure 8-10 on page 8-23).

The device name of the selected transistor is also reported in the command window, including the instance name where the device resides (for example, XADDER/XI101/MP12).

If you want PrimeRail to instantly write the name and location of a device port to the command window, click a spot in the waveform displayed in the Voltage Waveform Display window.

The following is an example of the result:

```
XBUS_CTRL/XMEM_DRV1/XI4\MP1:DRN (Number:19)
```

To clear the waveforms of a transistor, select the transistor in the list and click Clear. Click Hide to close the window.

The following is an example of the voltage drop waveform output results:

```
Searching P/G net in para view...
Design Hierarchy (instance master net type (models) library)
ff ff VDD TBox DESIGN3
Reading extraction of UserDefined.VDD in ff_n50 under /psd/DESIGN3 ...
Cell ff_n50, net UserDefined.VDD: 603 ports, 3239 nodes,
5905 edges, 9144 rectangles
Cell ff_n50, net UserDefined.VDD: Total memory for poRLCTree = 807832
bytes

Transient voltage drop summary
    Min voltage drop (mV) = 0
    Max voltage drop (mV) = 63.3
    Geometry [(17790  68250) (17790  68250)]
Top 5 transient voltage drop values at instance ports:
#1: 63.3 (mv) at Xdreg_reg[30]/MP5.SRC time 32.125e-9
#2: 63.241 (mv) at Xdreg[25]/MP5.SRC time 32.125e-9
#3: 62.838 (mv) at Xdreg[10]/MP5.SRC time 32.125e-9
#4: 62.509 (mv) at Xdreg[28]/Mp144.SRC time 32.125e-9
#5: 62.502 (mv) at Xdreg[15]/MP5.SRC time 32.125e-9
Max happens at [(17790  68250) (17790  68250)] inside the block ff_n50

Load VD Result  Memory: 43.016 MB  CPU: 0 seconds  Elapse: 0 seconds

Successfully loaded voltage drop map
INFO: set default voltage Lower/Upper bounds [-63.3 ~ 0] for frame0

Display VD Map  Memory: 56.875 MB  CPU: 0 seconds  Elapse: 0 seconds
```

Displaying the Electromigration Map

When dynamic analysis at the transistor level is complete, use `pgMap` to display average (AVG), root mean square (RMS), or maximum (MAX) current density violations in a map.

To display the electromigration map,

1. Enter `pgMap` or choose Transistor-level Dynamic Analysis > Display – Display Tx-level Dynamic Maps.
2. In the Display Map dialog box that appears (see Figure 8-1 on page 8-3), choose the analysis result from the Results menu for displaying the current density violations. Click Load.

3. In the Map Options section, select EM from the menu. Specify whether you want to display dynamic currents in average (AVG), root mean square (RMS), or maximum (MAX).

   Click Map Configuration to specify additional settings for map display.

   In the Map Configuration dialog box, select the metal or via layer to be displayed.

   Color Configuration – Click to define the color of each step to be shown on the map in the Display Color Setup dialog box (see Figure 8-8 on page 8-17).

   EM Rules Configuration – Click to define the lower and upper current density bounds in the EM Rules Configuration dialog box.

   For more information about the options in this dialog box, see online Help for the `pgMap` command.

   Click Hide to apply the changes you make and return to the Map Configuration dialog box. When you finish configuring settings in the Map Configuration dialog box, click Hide to apply the changes you make and return to the Display Map dialog box.

4. Display Options – Select the options related to the map you specified in step 3.

   For a detailed description of the options, see the online Help for the command.

5. Specify other options as necessary.

6. Click Apply. PrimeRail draws the electromigration map in the cell editing window.

   Before you view the rail analysis results of another cell, click Clear to remove the results of the current cell. If you need to view the results at a later time, be sure to save the data to an error cell or file.

   To reset the cell editing window to its normal status, click Clear and then Cancel.

   For more information about displaying the electromigration map, see the online Help for the `pgMap` command.

Querying Voltage Drop and Current Density Values

When the voltage drop or electromigration map is displayed, you can use the map to check for the voltage drop or current density values of a net.

To query values of the net, click Query at the top of the Display Map dialog box and then click the net being studied in the cell editing window. The tool writes the values to the log and the command window. If the maximum value is available, the tool also reports the time, in nanoseconds, when peak voltage or current occurs.

For more information, see “Querying Voltage Drop and Current Density Values” on page 8-22.

Reporting Voltage Drop and Current Density Violations

If you want PrimeRail to report an error cell or file on voltage drop or current density violations, click Report Setup at the bottom of the Display Map dialog box (see Figure 8-1 on page 8-3). The VD Report Setup or EM Report Setup dialog box opens, depending on whether the voltage drop or electromigration map is displayed.

For more information, see “Reporting Voltage Drop and Current Density Violations” on page 8-23.

Viewing Animated Voltage Drop Maps

As with cell-level dynamic analysis, you can use the `pgMoviePlayer` command to display an animated voltage drop map to pinpoint the critical cycles and perform a more detailed analysis to solve the problems. You can choose to view the voltage drop distribution for each time point, find out which cycles cause IR drop problems, and then run the detailed analysis only on those problematic cycles.

For more information, see “Viewing Animated Voltage Drop Maps” on page 8-26.

10

Using Macro Models

Nonstandard cells, such as I/O pads and hard macros, can have significant effects on voltage drop and electromigration at the full-chip level. The power consumption of these cells or the power and ground routing that exists inside the cells might allow current to flow through the cells and cause voltage to be applied to other regions of the chip. It is therefore necessary to be able to consider the internal power and ground network and power consumption of these types of cells during the top-level time-average rail analysis.

This chapter discusses how to model and import hard macros used in the full-chip analysis flow, as described in the following sections:

When to Use Macro Models

To improve performance, it is recommended that you use different macro models in time-average and dynamic analysis flows.

Table 10-1 lists the recommended macro model types and required data for generating each macro model in time-average and dynamic analysis flows.

Table 10-1 Using Macro Models

	Recommended Macro Model	Required Data
Time-average analysis	Static white box model	CONN view and current source file
Dynamic analysis	Dynamic white box model (DWM)	CONN view and the analysis results from the `poTxPowerAnalysis` run
	DWM-lite	CONN view and memory data sheet

Preparing Data for Hard Macros

PrimeRail allows you to combine the values of the routing and the power usage into a single construct that rail analysis can access during the full-chip flow. The construct consists of the following two parts:

- Connectivity (CONN) view: Represents, in a viewable manner, the power and ground networks inside the cell in question.
- Current source file (CSF): A file that records where current (and hence power) is drawn inside the cell in the ASCII format.

The cells that require CONN views to correctly re-create the resistance network are as follows:

- Power cells
- Cover cells
- Hard macro cells
- I/O pad cells

Note:
Corner pad cells should be treated as I/O pad cells.

In PrimeRail, a soft macro is any cell that has been placed and routed. It can contain standard cells, hard macros, or other soft macros. The CEL view of a soft macro is used to create both the white box and gray box macro models. Do not create either a CONN view or a current source file (CSF) for a soft macro.

Creating Connectivity Views

PrimeRail provides various methods for creating connectivity (CONN) views. Which method to use depends on the cell data size, the accuracy of text labeling on shapes, and layers inside the cell.

This section includes the following topics:

- Creating CONN Views
- Skipping Cells During CONN View Generation

Creating CONN Views

Run the `poConnViewPreparation` command to automatically invoke the Hercules connectivity engine to generate connectivity (CONN) views for nonstandard cells, either flat or hierarchical, but with less memory usage.

Note that the `poConnViewPreparation` command is designed to replace the `poCreateConnViewByHerc` command. When creating CONN views for hard macros, the `poConnViewPreparation` command provides the “skip cell” feature that reduces data size and improves performance of the CONN view generation process, without losing the accuracy on IR drop analysis. Creating a white box macro model by integrating current sources is also supported.

When you are creating a full-chip CONN view or an IP block CONN view for rail analysis, it is sometimes better not to generate CONN views for standard cells to remove the fingers. After the fingers are removed, the current sources located at the fingers are snapped to the center of standard cell power and ground pins. Skipping bit cells for memories saves memory usage in the downstream processing steps.

When a cell is skipped during CONN view creation but not during current source file generation, running the `poConnViewPreparation` command removes the cell content from the CONN view, but the current sources at the diffusion contacts in the cell are snapped to the center of cell instance boundaries that interact with top-level power and ground metals.

Because the tool cannot distinguish the cell types from GDSII data, the Skip Cell feature is available only to standard cells. For other cell types, current sources are snapped to the cell boundaries only when the center points of the power and ground pin rectangles are inside the cell boundaries.

The tool also allows you to use a replay file to choose which cells to be skipped. For example, if you want to skip all the cells in the design “top” except cell A, enter the following in the replay file:

```
poConnProcSkipCells "connViewPrep"
 "resetSkipCells_CONN:top"
poConnProcSkipCells "connViewPrep" "skipCells_CONN:top:*"
poCommProcSkipCells "connViewPrep" "skipCells_CONN:top:!A"
```

Note:

You should close the library and cell of interest before running this command.

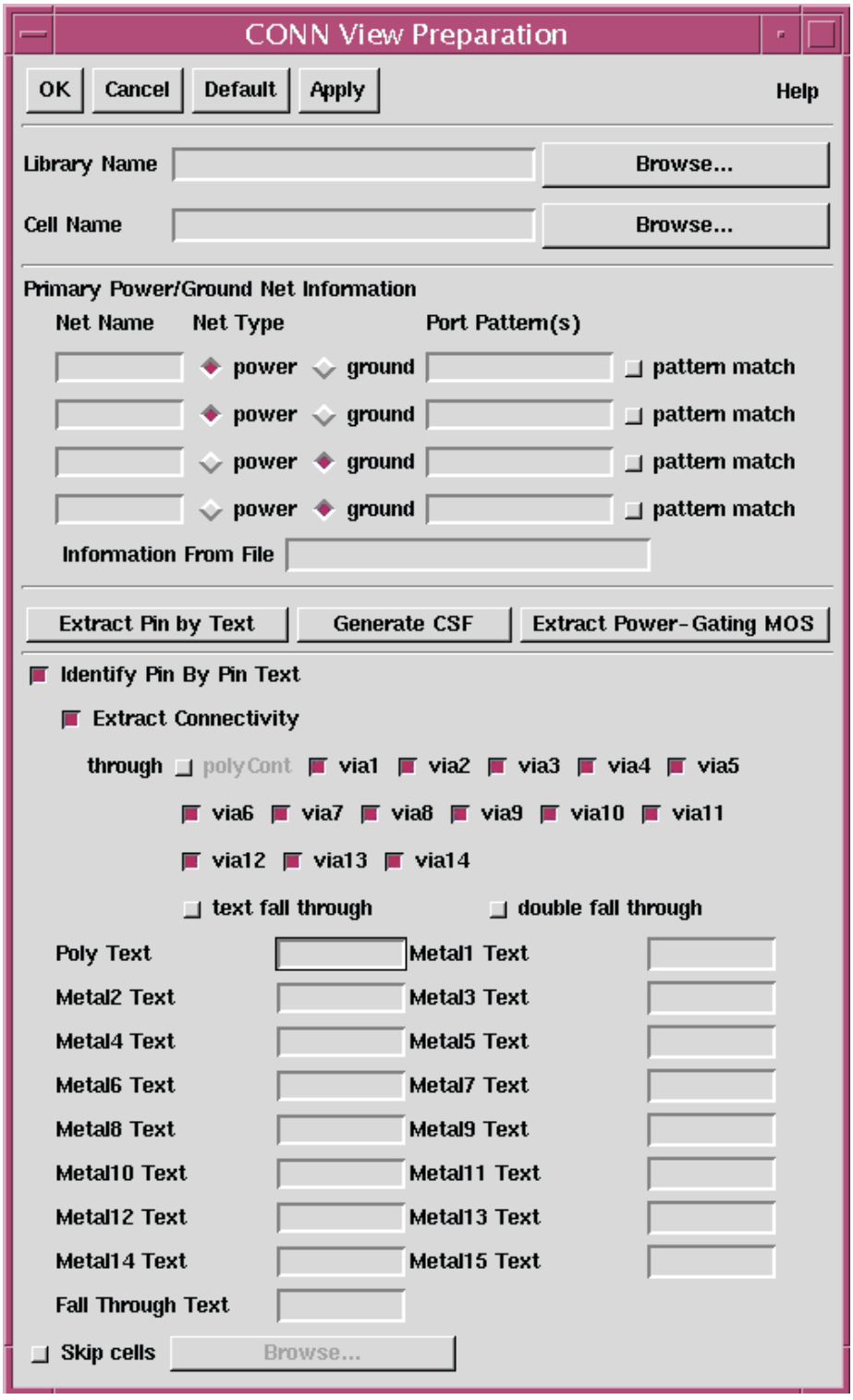
The number of cells printed in each Hercules command is limited to 300 to prevent the Hercules parser from stopping.

To create CONN views using the `poConnViewPreparation` command,

1. Enter `poConnViewPreparation` or choose Design Setup > CONN View Generation – Prepare CONN View.

   The CONN View Preparation dialog box opens.

Figure 10-1 The CONN View Preparation Dialog Box



2. Enter the name of the library or click Browse to select a name from the list that appears.
3. Enter the single cell name for which you want to generate the CONN view, or click Browse to select a name from the list that appears.
4. Click the Extract Pin By Text button to specify information for extracting pins by text. When clicked, the available options are displayed in the lower section of the Connectivity View Preparation dialog box.
5. If you want to generate current source files during CONN view generation, click the Generate CSF button to specify options for generating current source files. When clicked, the available options are displayed in the lower section of the CONN View Preparation dialog box.

   For more information, see "Generating CSFs During CONN View Generation" on page 10-8.
6. Specify other options as necessary. For a detailed description about the options, see the online Help for the command.
7. Click OK or Apply.

Skipping Cells During CONN View Generation

Because the Skip Cell option is available on both the CONN view generation panel and the Generate CSF panel in the CONN View Preparation dialog box (see Figure 10-1 on page 10-5), there are four combinations of generating CONN views and current source files for each child cell:

- No skipping for both CONN view and CSF generation – If the Skip Cells option is deselected on both panels, the tool will generate CONN views and current source files for all the cells in the design.

  This is the default setting.
- Skipping both CONN view and CSF generation – If the Skip Cells option is selected and a cell is chosen on both panels, the tool will not generate CONN views and current source files for this cell. This is helpful when you are analyzing bit cells in a memory design.
- Skipping CONN view generation but not CSF generation – If you select the Skip Cells option (and choose a cell) on the CONN view generation panel, but deselect the option on the Generate CSF panel, the tool will not generate CONN views for the chosen cell but will snap all the current sources inside the cell to the center of its power and ground pins.

  Note that this setting applies to standard cells only. Applying this setting to other cell types can have unpredictable results.

- Skipping CSF generation but not CONN view generation – If you select the Skip Cells option (and choose a cell) on the Generate CSF panel, but deselect the option on the CONN view generation panel, the tool will generate only CONN views for the chosen cell; no current source file will be generated. This setting usually applies to pad cells.

If a huge number of cells are to be skipped, use wildcards to specify the cell names in the replay file. For example,

```
poConnProcSkipCells "connViewPrep" "skipCells_CONN:TOP:A*"
poConnProcSkipCells "connViewPrep" "skipCells_CSF:TOP:B*"
```

The first command line means to skip all child cells with a cell name starting with A during CONN view generation. The second line means to skip all child cells with a cell name starting with B during current source file generation.

If more cells are to be skipped than not skipped in the design, use the inverse expression to input the cell names in the replay file. For example,

```
poConnProcSkipCells "connViewPrep" "skipCells_CSF:TOP:*"
poConnProcSkipCells "connViewPrep" "skipCells_CSF:TOP:!A*"
```

Then the tool will skip all child cells during current source file generation except the child cells with a cell name starting with A.

Generating Current Source Files for GDSII Macros

By default, the power consumption of a macro cell is assumed to be evenly provided by all available power pins. To improve rail analysis accuracy, you might want to distribute the current demands nonuniformly, by creating a current source file and loading it during rail analysis.

You do not need to create any current sources for standard cells or I/O cell's CONN views.

To have PrimeRail generate the current sources of a hard macro, use the `poGenCurrSource` or `poConnViewPreparation` command.

If you want to calculate currents consumed by transistors inside the hard macros, use the `poTxGenCurrSource` command (see "Calculating Saturation Currents for Transistors" on page 10-15).

This section includes the following topics:

- Calculating Currents for Transistors
- Writing Out Current Source Data
- Removing Current Source Data

Generating CSFs During CONN View Generation

Use the `poConnViewPreparation` command to generate current source files in PrimeRail.

The two major parts in the current source file are current source locations and current values. The current source locations can be determined by the locations of diffusion contact cuts to the approximate MOS transistor terminal locations. You must specify the mask name of P and N diffusion layers, so the tool can invoke Hercules to output all diffusion contact cut locations to a log file during the CONN view creation process. Based on the log file, the tool generates current source files with uniform current distributions over all device locations. If the mask names of diffusion layers are not available, you can choose to generate current sources at the Via1 locations.

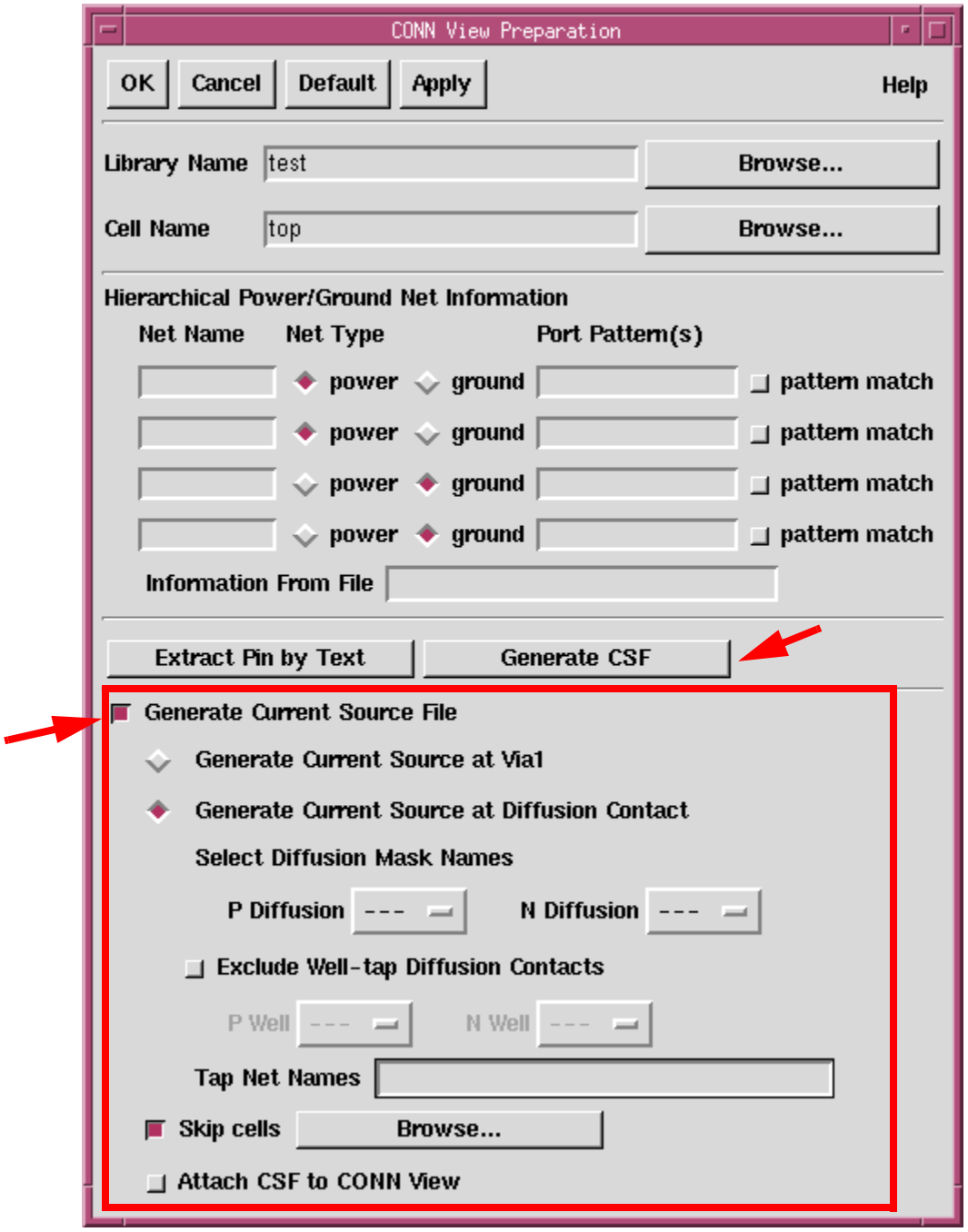
You must provide the correct mask names of diffusion layers if you want to generate current sources at the locations of diffusion contact cuts. Only uniform current distributions are available because the tool does not run circuit simulation during CONN view creation. However, the resulting current source file is close to the one from the saturation current mode using the `poTxGenCurrSource` command, because the number of diffusion contact cuts is proportional to the width of MOS devices.

To create current source files using the `poConnViewPreparation` command,

1. Enter `poConnViewPreparation` or choose Design Setup > CONN View Generation – Prepare CONN View.

   In the CONN View Preparation dialog box that opens (see Figure 10-2 on page 10-9), specify the library and cell information.
2. Click the Generate CSF button to specify options for generating current source files during CONN view generation. When clicked, the available options are displayed in the lower section of the CONN View Preparation dialog box.

Figure 10-2 CONN View Preparation Dialog Box—Generate CSF



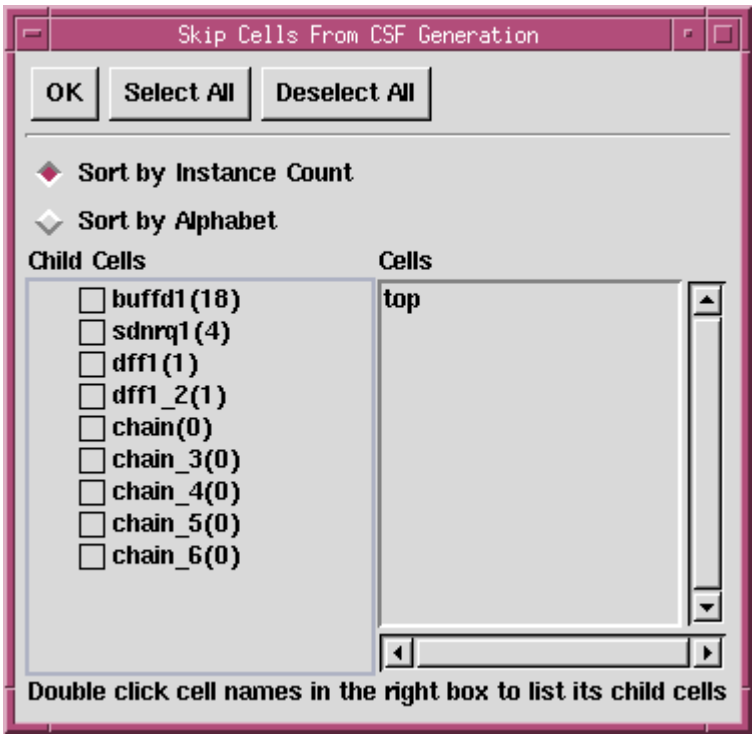
3. Select Generate Current Source File to enable the current source file generation process. Note that the generated current source file is net-based, meaning each power and ground net has a corresponding current source file.

4. Select Generate Current Source At Via1 to generate current sources at the Via1 locations. When selected, the current source file name is in the form *net_name-via1*.csf.
5. Select Generate Current Source at Diffusion Contact to generate current sources at the locations of diffusion contact cuts. When selected, the current source file name is in the form *net_name-contact*.csf.
6. Under Select Diffusion Mask Names, specify the diffusion mask names for locating MOS transistor terminals. Select this option when you want to generate the current source file at the locations of diffusion contact cuts.

   Select the mask names from the menus for p-type and n-type diffusions. The tool searches for keywords of “pdiff,” “ndiff,” and other user-defined mask names in the Milkyway technology file and lists them in the menu.
7. Select Exclude Well-tap Diffusion Contacts to exclude the well-tap diffusion contacts for generating more accurate current source locations.

   Select the mask name from the menu. The tool searches for keywords of “pwell,” “nwell,” and other user-defined mask names in the Milkyway technology file and lists them in the menu. The tool can work with well layers of P-well only, N-well only, or both.
8. In the Tap Net Names box, enter the tap net names to generate the current source files with locations on the non-transistor-terminal diffusion contacts separately. This option is useful when the design being studied uses a different power or ground net (other than the main power or ground net) to connect to the non-transistor-terminal diffusion contact cuts.
9. Select “Skip cells” and click Browse to specify the cell to be excluded from the current source file generation.

The Skip Cells From CSF Generation dialog box opens.



- Sort by Instance Count – List the child cells in the order of cell instance count (from high to low).
- Sort by Alphabet – List the child cells in the order of cell name.

In the Child Cells box, select any child cells to be excluded. The tool displays the name of the selected cell in the Cells box. This cell name is the same as the input cell name.

To select all the child cells as skip cells, click Select All. You can deselect all the selected cells by clicking Deselect All.

Click OK to save the change you make and close the dialog box.

10. Select “Attach CSF to CONN View” to attach the generated current source files to the CONN view.
11. Enter other information as necessary.
12. Click OK or Apply.

Generating CSFs Using poGenCurrSource

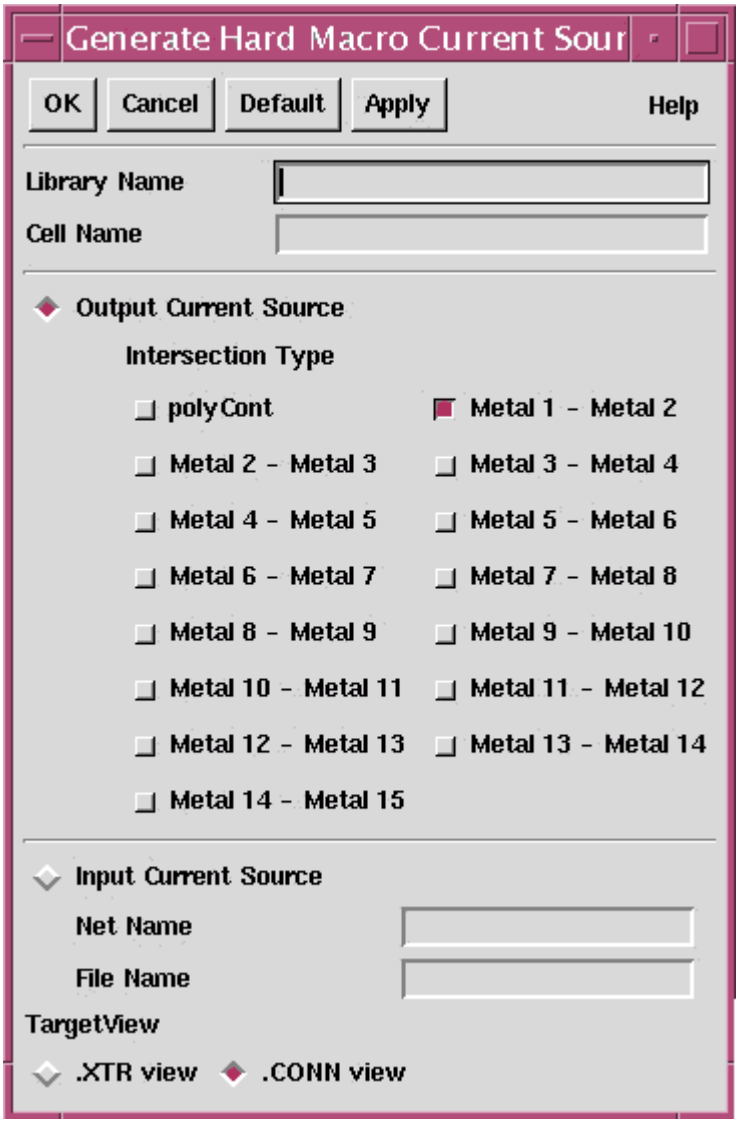
After you stream a hard macro into a CEL view (using the `auStreamIn` command), you can use the `poGenCurrSource` command, which writes the default current distribution for all power and ground nets to the attached files in the CONN view.

To generate current sources for a macro,

1. Enter `poGenCurrSource` or choose Design Setup > Current Source File Generation – Uniform Distribution CSF.

   The Generate Hard Macro Current Source dialog box appears.

Figure 10-3 Generate Hard Macro Current Source Dialog Box



2. Enter the name of the library.
3. Enter the name of the cell for which you want to generate the current source files. The cell must have a CONN view attached.

4. Choose which metal-to-metal intersections to use for a current location. These are intersections in the power and ground network for the individual nets.

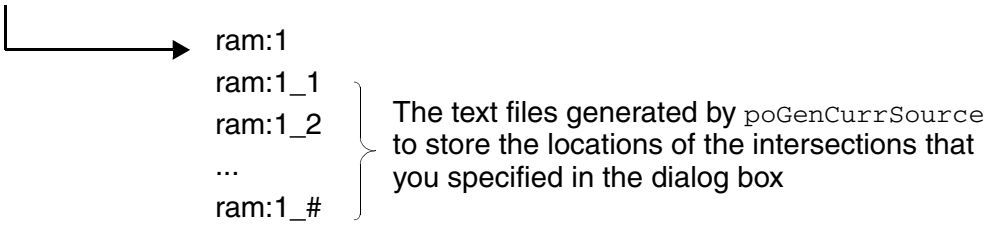
   For instance, if a metal1 VDD rail intersects with a metal2 VSS rail, this location is not used for a current source inside either net. You can choose multiple interactions.

5. In the Target.View section, choose to save the outputs to the XTR or CONN directory of the specified library.

6. Click OK or Apply.

When executing the command, PrimeRail reports that it is processing geometry for each of the power and ground nets in the CONN view of the cell. For example, if the block name is ram, the created files are as follows:

The view of the power and ground geometries used when you want to open and view the CONN view of the cell (ram.CONN)



A Current Source File Sample

The header information at the beginning of the current source file includes the following information, showing the current source count in bold:

```
**********************************
       * Current Source Report
       * Macro Type : GDS2
       * Cell Name :
       * Bounding Box : [-7416 -192160]
       * Length Precision : 1000
       * Net Name : VSS
       * Supply Voltage : N/A
       * Power Consumption : N/A
       * Equivalent Current : N/A
       * Current Source Count : 13138
       **********************************
       16 175140 336140 1
       16 342540 335460 1
       ...
```

Lines of current sources

Note that

- The number of lines after the second line of asterisks (*) matches the number specified in the Current Source Count field
- The Cell Name field is not required in this file because the file is named by the name of the cell itself, such as cellname:1, cellname:2, and so on
- The Supply Voltage, Power Consumption, and Equivalent Current boxes are not applicable in this sample, because they can be specified when you perform the rail analysis (with the synopsys_pr_setup.e file) and the Input Cell Instance Power File option of the `poCalculatePower` command

You specify the current source as follows:

```
Layer# x-coordinate y-coordinate current-drawn
```

The value of `current-drawn` at each location is scaled based on the cell instance power usage you specify.

Loading Existing Current Source Files

You can modify the output current source file that is generated with the `poGenCurrSource` command and load the file back to the tool.

Note that you cannot modify the syntax format of the output current source file. If you do, the tool will not recognize the user-defined entry in the file.

To load a current source file,

1. Enter `poGenCurrSource` or choose Design Setup > Current Source File Generation – Uniform Distribution CSF to open the Generate Hard Macro Current Source dialog box (see Figure 10-3 on page 10-12).
2. Specify the names of the library and the cell.
3. Select Input Current Source and enter the name of the net whose current sources you want to read in. Then enter the name of the current source file you want to load.
4. In the Target.View section, choose to save the outputs to the XTR or CONN directory of the specified library.
5. Click OK or Apply.

Calculating Currents for Transistors

In addition to `poGenCurrSource`, PrimeRail provides the `poTxGenCurrSource` command to calculate how much current is consumed by all of the transistors powered by the specific power and ground nets in the design with the following methods:

- Calculating Saturation Currents for Transistors
- Distributing Uniform Currents to Transistors

Different methods used for generating current source files (CSF) lead to different accuracies that are applied to the final white box model. The accuracy from low to high is uniform distribution and then saturation current.

Calculating Saturation Currents for Transistors

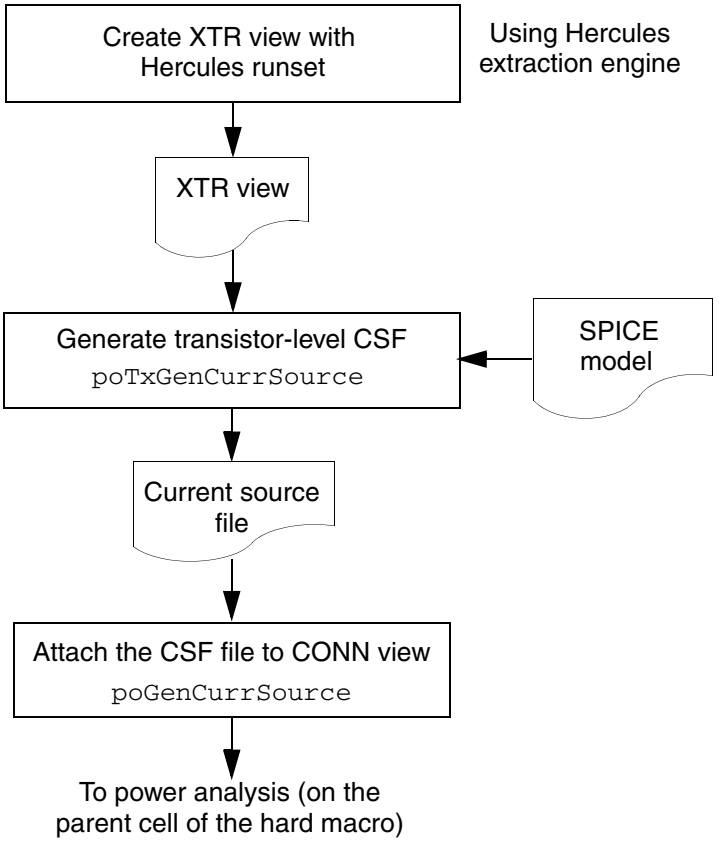
Use the `poTxGenCurrSource` command to calculate how much current is consumed by all of the transistors powered by the specific power and ground nets in the design based on the saturation method. The command does not require the power consumption value of hard macros, but you need to specify the calculation results later during the top-level power analysis.

PrimeRail uses an XTR view to obtain the device-net connectivity in relation to the specific power and ground nets, the geometric parameters (length and width), and the physical location of the transistors. Using the XTR view also allows PrimeRail to determine the connectivity between the transistors and the signal nets and to find the transistors driven by the signal nets.

PrimeRail uses a SPICE model to calculate the gate capacitance values of the devices (found in the XTR view) to be driven by each transistor being analyzed. You can scale the current consumption values for devices inside the specific cell masters or cell instances.

Figure 10-4 shows the flow of generating saturation current sources at the transistor level with the XTR view.

Figure 10-4 Creating Saturation Currents for Transistors



This is the sequence in Figure 10-4:

1. Create XTR view with Hercules runset – The standard StarRC transistor-level extraction flow for generating the XTR view of a hard macro.

2. Generate transistor-level CSF – Run the `poTxGenCurrSource` command to calculate the current consumption value of transistors inside the hard macro.

   An ASCII file is created in the working directory for each of the power and ground nets you specify in the Generate Transistor Current Source dialog box (see Figure 10-5).

3. Attach the CSF file to CONN view – Run the `poGenCurrSource` command to attach the generated current source file to the CONN view of the cell (see "Loading Existing Current Source Files" on page 10-15).

Creating XTR View With Hercules Runset

Create the XTR view of the macro for which you want to generate the saturation currents. This flow follows the standard StarRC transistor-level extraction flow.

Enter the following syntax at the UNIX prompt to use Hercules for generating the XTR view:

```
% hercules -rcxt RUNSET.ev
```

The generated XTR view contains complete information about geometric parameters for the extracted devices.

For more information about creating the XTR view with the Hercules runset, see the *StarRC User Guide*.

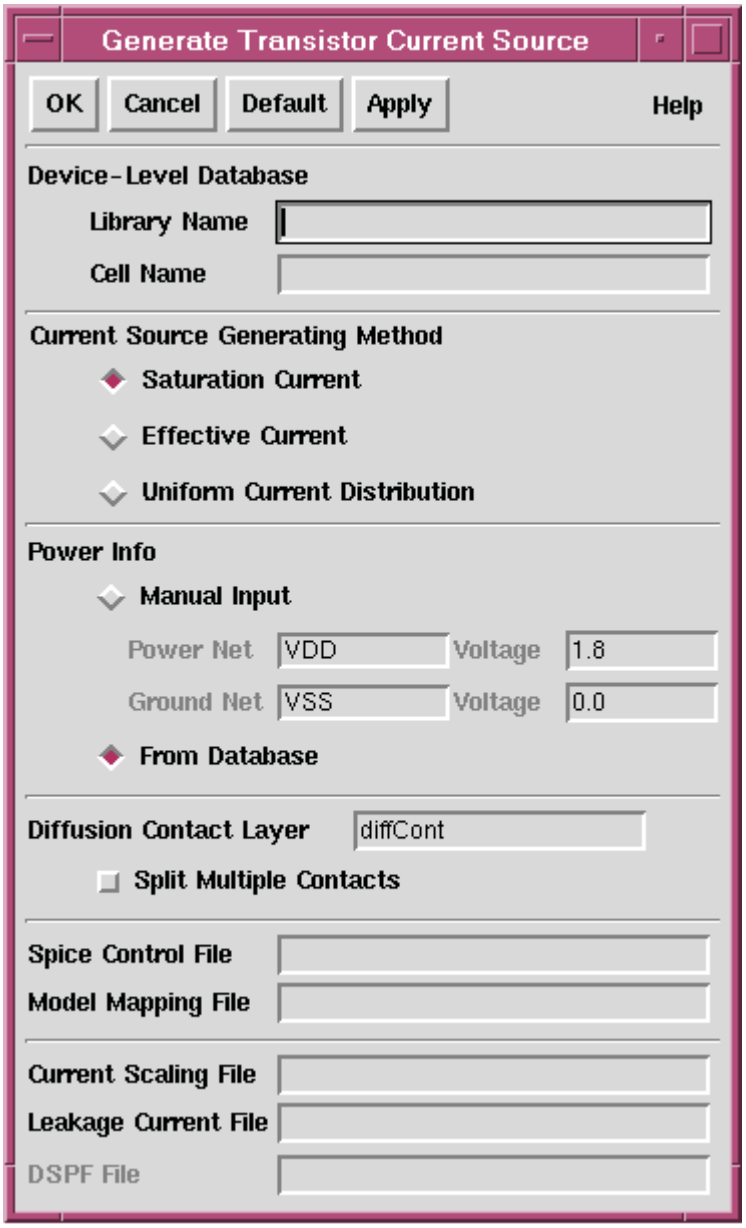
Creating Transistor-Level Current Source Files

Use the `poTxGenCurrSource` command to generate saturation currents for transistors.

1. Enter `poTxGenCurrSource` to create the transistor-level current source file for the hard macro.

   The Generate Transistor Current Source dialog box appears.

Figure 10-5 The Generate Transistor Current Source Dialog Box



2. Enter the name of the library where the Milkyway XTR view is created with the Hercules runset.
3. Enter the name of the top-level cell for which you want to generate transistor currents.
4. Choose Saturation Current as the method for calculating saturation currents.

   Note:
   The effective current method is available only when you have the beta license key.

5. Specify the power information, by selecting Manual Input and entering the names of the power and ground nets. Specify the voltage value of the power net. The default value for the power net is 1.8, and the default value for the ground net is 0.0.

   Alternatively, select From Database to use the power information that is saved and loaded into the database using the `poLoadRailSetup` command (see "Preparing Rail Setup Data" on page 5-2).

6. In the Diffusion Contact Layer field, enter the name of the layer into which multiple diffusion contacts in a single source or drain area will be merged. The default layer name is diffCont.

   You need to specify the correct diffusion contact for the command to calculate the current.

   Select Split Multiple Contacts if you want to divide the transistor currents among these contacts.

7. Enter the name of the SPICE control file that contains the SPICE model for the transistors in the database.

8. In the Model Mapping File field, enter the name of the mapping file that contains the transistor model mapping information. The transistor cell name is automatically mapped to the corresponding SPICE model for each layout model type.

   Syntax

   ```
   transistor_cell model_name
   ```

   where transistor_cell is the name of the transistor cell and supports wildcards (?*). The model_name is the name of the transistor model and has to be an exact string—multiple-to-one mapping.

   Note:
   The transistor_cell is case-sensitive, and model_name is not.

   Example

   ```
   p*                 p_mos
   n* !nor* !nand*    n_mos
   ```

9. Enter the name of the current scaling file that defines how to scale the transistor currents for cells or instances. The file supports multiple entries of scaling rules.

   The following is the syntax of the file:

   ```
   CELL: cellName1 cellName2 ... cellNameN scaling_factor
   INST: instName1 instName2 ... instNameN scaling_factor
   ```

   Wildcards (*) and negation (!) symbols are both accepted to define cellName and instName.

Example

```
CELL: ram16x2048_ioblock* 10.0
CELL: ram16x2048_ctrl 5.0
INST: layout_instance_14* !layout_instance_145* 0.1
INST: layout_instance_1452/M3* 0.001
```

Note:

The instance scaling rule has higher priority than the cell scaling rule and the last entry has higher priority than the first entry.

Caution:

Because the output CSF does not contain the information about the relationship between the cells and the different current sources, you must perform scaling during the current calculation. Scaling cannot be done without this information.

10. Enter the name of the file where leakage current rules are written. If a scaling file is provided with the scaling rules specified in step 9, the command scales currents only after leakage current is available.

    The leakage current file supports the following three different formats, with the order of precedence from high to low:

    - Instance-based
    - Model- and size-based (unit current)
    - Model-based

    Syntax

    ```
    MODEL <model_name> <leakage_current>
    INSTANCE <instance_name> <leakage_current>
    MODEL_PMS <model_name>
      <leakage_current_per_micron_square>
    ```

    Wildcard (*) symbols are accepted to define model_name and instance_name. The default current unit is amp.

    Example

    ```
    INSTANCE: 0/35/* 2.5e-10
    MODEL_PMS: p 3e-11
    MODEL: n 1e-10
    ```

11. In the DSPF File box, enter the name of the netlist in the detailed standard parasitic format (DSPF) that contains net loading capacitance information to be loaded to the XTR view of the hard macro.

    Note:

    The DSPF netlist is required only when you generate currents with net switching activities by selecting the effective current method in step 4.

12. Click OK or Apply.

    PrimeRail automatically calculates the device's currents. The saturation current values are also scaled based on the information you specify about cell master and cell instance names and scaling factors.

    The `poTxGenCurrSource` command automatically creates ASCII output files with the generated current source data in the current working directory. The output file is named as follows:

    ```
    BLOCK.NET.macro
    ```

    The file contains the same syntax as that for the standard current source file data created by the `poGenCurrSource` command.

    Important:
    Write down the reported power consumption values for the block as a whole. Specify this value in the cell instance power file that is specified with the Input Cell Instance Power File option in the Power & Current Analysis dialog box with the `poCalculatePower` command when you perform power analysis on the parent of the hard macro in question.

Attaching the Current Source File to the CONN View

After the transistor-level current source file is generated, attach it to the CONN view of the cell, using the `poGenCurrSource` command.

1. Enter `poGenCurrSource` or choose Design Setup > Generate Macro Current Source to open the Generate Hard Macro Current Source dialog box (see Figure 10-3 on page 10-12).
2. Specify the name of the library and the cell.
3. Select Input Current Source, and enter the name of the net whose current sources you want to read in. Then enter the name of the current source file to be loaded.
4. Click OK or Apply.

Caution:
You must attach the current source file to the CONN view for each power and ground net in the design that has a current source file.

Distributing Uniform Currents to Transistors

If you do not want to perform current calculation, you can have PrimeRail distribute a uniform value of current to each transistor in the design. The uniform current value is set to 1 ampere.

Note:

You need to prepare an XTR view before running the `poTxGenCurrSource` command. For more information about creating an XTR view, see "Creating XTR View With Hercules Runset" on page 10-17.

To set the current value uniformly,

1. Enter `poTxGenCurrSource` to open the Generate Transistor Current Source dialog box (see Figure 10-5 on page 10-18).
2. Enter the name of the library where the Milkyway XTR view is created with the Hercules runset.
3. Enter the name of the top-level cell for which you want to generate transistor currents.
4. Choose Uniform Current Distribution as the calculation method. This will assign 1 ampere of current to each transistor location.
5. Click OK or Apply.

   When current source file is generated, attach it to the CONN view of the cell, using the `poGenCurrSource` command. For the step-by-step instruction, see "Attaching the Current Source File to the CONN View" on page 10-21.

Writing Out Current Source Data

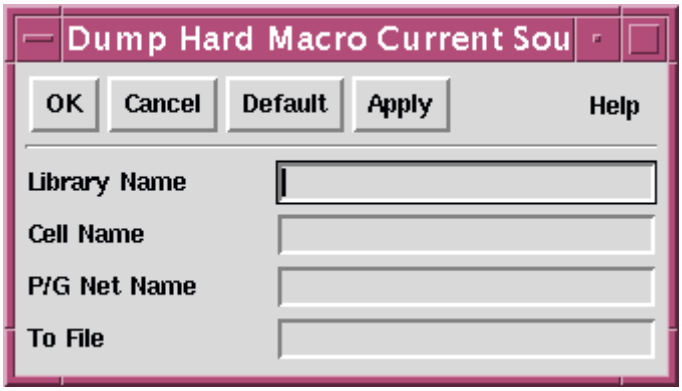
Use the `poDumpCurrSource` command to write current sources of a power or ground net from the attached CONN view to an ASCII file. This enables you to verify if the information is generated correctly before you proceed to power and rail analysis.

To write current source data of a net,

1. Enter `poDumpCurrSource` or choose Diagnosis > Report – Dump Macro Current Source.

   The Dump Hard Macro Current Source dialog box appears.

Figure 10-6 The Dump Hard Macro Current Source Dialog Box



2. Enter the name of the library and the hard macro cell.
3. Enter the name of the power or ground net whose current source data is to be written out to a file.
4. Enter the name of the file to which the current source data of the net is written.
5. Click OK or Apply.

Removing Current Source Data

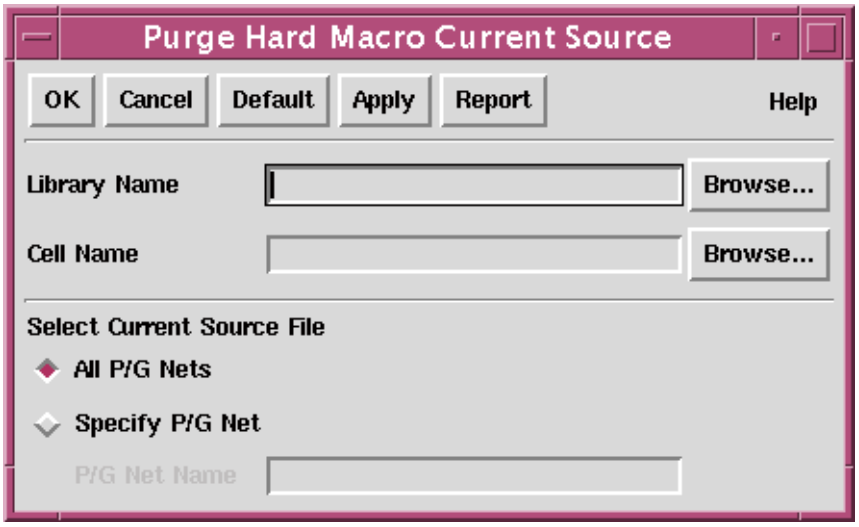
Use the `poPurgeCurrSource` command to delete current sources of macro cells for a power or ground net. Because PrimeRail saves current sources in the files that are attached to the CONN view, make sure the CONN view is open before you run this command.

To remove current source files of a power or ground net,

1. Enter `poPurgeCurrSource` in the command window.

   The Purge Hard Macro Current Source dialog box appears.

Figure 10-7 The Purge Hard Macro Current Source Dialog Box



2. Enter the name of the design library.
3. Enter the name of the cell that contains the nets whose current source data you want to delete.
4. Select All P/G Nets if you want to delete the current source data of all the power and ground nets in the cell. If you want to delete data from a specific power or ground net, select Specify P/G Net and enter the name of the net in the P/G Net Name text box.
5. Click OK or Apply.

Static White Box Model

Accurately modeling hard macro blocks in the full-chip reliability analysis is important because hard macros are generally a significant portion of the whole chip in terms of chip area and power consumption.

You can use the time-average white box model that is created by the `poExtractPGParasitics` command to analyze the dynamic reliability effects of a design at the full-chip level. When running the `poExtractPGParasitics` command, the tool first checks if the RAIL view contains any hard macro model information, like the CONN view and the current source for the static macro model, or the DWM (or DWM-lite) model. If no hard macro model information is available, the tool reports it as a black box in the log file.

Different from the dynamic macro model generated with the DWM-lite or DWM flow, the time-average white box created by the `poExtractPGParasitics` command contains current source information only. You need to run the `poCalculatePower` command in order to generate the current waveforms.

- When performing a time-average power analysis, running the `poCalculatePower` command creates a triangle current waveform for the power value that is calculated by PrimeTime PX. However, if the power value is user-specified, running the `poCalculatePower` command will create a constant current waveform for the power value.
- You can then run the `poRailAnalysis` command to distribute the generated current waveform to each of the current source locations based on current source data stored in the CONN view.
- When performing a vector-based power analysis, running `poCalculatePower` will not create current waveforms for the power value calculated by PrimeTime PX. An error message is printed to the log file, like the following:

  ```
  0 hard macros (out of total 187) have DWM
  178 macro instance without current waveform
  ```

  When you specify a power value to be used with the `poCalculatePower` command, the tool will create a constant current waveform for the power value you assign. You can then run `poRailAnalysis` to distribute the generated current waveform to each of the current source locations based on current source data stored in the CONN view.

Dynamic White Box Model

PrimeRail provides the dynamic white box model (DWM) technology that allows you to capture the analog effects in a hard macro. When transistor-level power analysis is complete, you can generate a macro model for a hard macro by applying the DWM technology and use the model during full-chip analysis with reusability. Each macro model is identified by cell name and power and ground net names and contains parasitics and current waveform information.

Dynamic white box model (DWM) is a Milkyway data structure stored in the RAIL view. Because the rail analysis in PrimeRail is performed on the net-by-net basis, the DWM holds a model for each individual power and ground net. Each model contains the following information—power and ground net capacitance, intrinsic capacitance, current locations, and one or multiple modes. Because the power consumption of a building block is dominated by some control signal, it is possible to categorize the current waveforms of the circuit into several different waveform signatures based on different operation modes, like read mode, write mode, stand-by mode, and so on. While the DWM is generated, operation modes have to be determined according to the macro manufacturer's data sheet and/or timing and power libraries. Each mode in the DWM model contains current waveform information, which will be automatically triggered and applied to the rail analysis according to the VCD input. Similarly, it can be automatically adopted in the vector-free flow, or assigned by users manually.

PrimeRail allows you to store different operation modes (for example, read or write mode for an SRAM) in the macro model. The operation mode is determined by the state of I/O ports of the design, and each mode results in different current waveforms. When performing a full-chip rail analysis (see "Performing Full-Chip Rail Analysis" on page 12-37), PrimeRail automatically picks up the waveforms of the chosen mode based on the given VCD input.

When you are creating a mode in a DWM model, PrimeRail also allows you to specify multiple trigger signals (a bus or a bundle of signals) for the mode being created. Creating asynchronized modes is also supported. A synchronized circuit design relies on a single control signal (usually known as a clock) to trigger the operation. Asynchronized designs do not require such a signal but rather are active whenever the input signal changes. For example, an SRAM contains one synchronized write port and one asynchronized read port.

PrimeRail also supports handling multiple PVT corners of DWM (or DWM-lite) models. You can generate the DWM (or DWM-lite) models based on different PVT settings by specifying a corner in the DWM configuration file. The tool then runs the dynamic power and rail analysis by obtaining the corner information from the PrimeTime PX report and automatically picks up the closest corner stored in the DWM model. Note that unlike with standard cells, no scaling (interpolation and extrapolation) is performed for DWM models.

If the existing macro model is generated with PrimeRail version prior to Z-2007.03, you have to clean up the database using the `poTxPurgeDWM` command before generating DWMs with multiple-corner support. Otherwise, if generating DWMs on top of an existing DWM, the new one will keep replacing the old model, and there will be only one default corner in the library.

PrimeRail provides two ways for generating a DWM model; they are described in the following sections:

- Generating Macro Models, Using the DWM-Lite Flow
- Generating Macro Models, Using the DWM Flow

Generating Macro Models, Using the DWM-Lite Flow

PrimeRail provides the DWM-lite flow that allows you to generate DWMs without performing the transistor-level analysis. The DWM-lite flow provides a more straightforward way of generating DWMs with a very small deviation of accuracy than the actual DWM model.

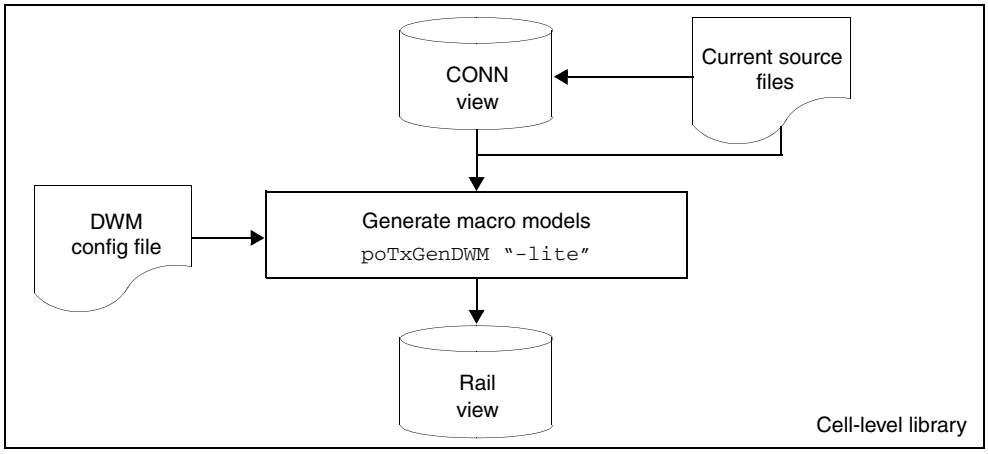
Running the DWM-lite generation flow involves one Scheme command, `poTxGenDWM`, and two input files—the DWM configuration file and the current source file (CSF) that is used in the time-average analysis.

By providing a total current waveform, DWM-lite distributes and scales the waveform to all the device locations based on what is described in the current source file. For memory cells, it is recommended that you run the `poCONNViewPreparation` command to create the

CONN view and current source files. It is also recommended that the switch of skipping the bit-cells inside the memory cell is enabled. For details, see “Creating Connectivity Views” on page 10-3.

Figure 10-8 illustrates the DWM-lite data flow.

Figure 10-8 DWM-lite Data Flow



Because the macro models generated with DWM-lite flow contain the total current waveform, running `poCalculatePower` does not generate current waveforms. The command will use the total current waveforms in the macro model directly and will ignore the power values either calculated by PrimeTime PX or assigned by the user. The `poRailAnalysis` command will distribute this current waveform to each of the current source locations based on the current source data saved in the CONN view.

Before Generation

The tool requires a configuration file and a current source file before you can generate a hard macro model using the DWM-lite flow.

Preparing a DWM Configuration File

The DWM configuration file contains the basic information about the hard macro being modeled, primary input and output ports, and mode configuration. You can run the `poTxCreateDWMConfig` command to automatically create a DWM configuration file based on the information stored in the LM view or the .db file. Alternatively, you can manually create one if you want the flexibility of controlling how DWMs are generated by modifying the configuration file.

To automatically create a configuration file,

1. Choose Macro Modeling > DWM-lite – Create DWM Config File from the menu or enter `poTxCreateDWMConfig "lite"` in the command window.

2. In the DWM Config File Creation dialog box (see Figure 10-12 on page 10-37) that appears, specify the options as necessary. Note that by default the output DWM configuration file is named as dwm_lite.cfg for DWM-lite analysis.

   For more information about the options in this dialog box, see online Help for the command or Figure 10-12 on page 10-37.

3. Click OK or Apply.

Preparing a Current Source File

For running the DWM-lite flow, you need to provide a current source file. Then specify the name of the current source file and the parasitic capacitances in the DWM configuration file, using the following syntax:

```
Power Net       : VDD {csf=VDD.csf, C_total=10pf}
Ground Net      : VSS {csf=VSS.csf, C_density=1e-9}
```

Table 10-2 Keywords Used in the Current Source File

Keyword	Description
csf	The current source file names associated with the corresponding power and ground nets
C_total	The total capacitance for power and ground nets, in the unit of Farad
C_density	The density of the capacitance, in the unit of Farad/ um^2

You can choose to either specify the capacitance or capacitance density (or both) for the power and ground nets. When both of them are specified, C_total is honored.

If no “csf” option is specified in the DWM-lite configuration file, PrimeRail automatically uses the current source file attached to the CONN or XTR view, depending on how the current source file is generated.

For the total current waveform, you can provide piecewise linear total current waveforms in the DWM configuration file, or some parameters that are specified in the manufacturer’s data sheet and let PrimeRail emulate a current waveform automatically. These two methods are described in the following sections.

- Providing Total Current Waveform in the PWL Table

  Use the keyword PWL waveform to assign the total current waveform to a mode. To comment a line, start the line with one of the following symbols: asterisk (*), slash (/), semicolon (;) or pound sign (#). Each value in the configuration file is case-insensitive.

  ```
  * Basic information
  Library Name  : DWM
  ```

```
Cell Name     : ram_16x2048
Power Net       : VDD {csf=VDD.csf, C_total=10pf}
Ground Net      : VSS {csf=VSS.csf, C_density=1e-9}

Supply Voltage : 1.8V
Max Frequency : 200MHz

* IO information
Input          : A[10:0]
Input          : CLK
Input          : WEB
Input          : CEB
Inout          : D[15:0]

* Operation mode information
Define Mode   : Read Mode
Trigger       : CLK  rise
Condition     : (CEB & WEB)
PWL waveform: (0,0) (1ns, 0.1mA) (2ns, 0)

Define Mode   : Write Mode
Trigger       : CLK  rise
Condition     : (CEB & !WEB)
PWL waveform    : (0,0) (1ns, 0.2mA) (2ns, 0)
```

Table 10-3 Keywords Used in the DWM Configuration File

Keyword	Description
Library name	The name of the library where the transistor-level analysis is run. The library contains the PARA view (where the `poTxPowerAnalysis` result is saved) and the RAIL view (where the DWM data is to be saved).
Cell name	The name of the hard macro cell that has been analyzed by the `poTxPowerAnalysis` command.
Power net and ground net	The names of the power and ground nets that need to be modeled, separated with a space or tab.
Input, output, and inout	The names of the input ports, output ports, and bidirectional ports.
Define mode	Define the operation mode to be read or write.
Trigger	Define which I/O port triggers the operation mode. A switching direction of rise (r) or fall (f) is followed by the port name.

Table 10-3 Keywords Used in the DWM Configuration File

Keyword	Description
Condition	Define the state when the mode is to be activated. It is represented by an in-fix Boolean expression. Tokens (', '), NOT(!), AND(&), and OR (|) are supported.
PWL waveform	Piecewise-linear total current waveform used for generating DWM-lite modes.

The current waveform will be distributed based on what is described in the current source file. PrimeRail automatically checks the current direction and assigns it correctly to the power and ground net.

- When No Total Current Waveform Is Available

  When the total current waveform is not available, you can generate the current waveform by pre-characterizing the cell data. The only parameters you have to provide are cycle time, peak current, average current, time-average current, and/or access time. You can easily obtain these parameters from the manufacture's data sheet.

  Note:
  If the peak current parameter is not specified, PrimeRail automatically assigns one based on the other parameters. A warning message will also be issued.

  The following is an example of the configuration file. To comment a line, start the line with one of the following symbols: asterisk (*), slash (/), semicolon (;) or pound sign (#). Each value in the configuration file is case-insensitive.

```
* Basic information
Library Name  : DWM
Cell Name     : ram_16x2048
Power Net: VDD {csf=VDD.csf, C_total=10pf}
Ground Net: VSS {csf=VSS.csf, C_density=1e-9}

Supply Voltage        : 1.8V
Max Frequency         : 200MHz

* IO information
Input                 : A[10:0]
Input                 : CLK
Input                 : WEB
Input                 : CEB
Inout                 : D[15:0]

* Operation mode information
Define Mode           : Read Mode
Trigger               : CLK  rise
Condition             : (CEB & WEB)
```

```
Cycle Time            : 2ns
Peak Current          : 50mA
Average Current       : 2mA
Access Time           : 1.8ns

Define Mode           : Write Mode
Trigger               : CLK  rise
Condition             : (CEB & !WEB)
Cycle Time            : 1.6ns
Peak Current          : 40mA
Average Current       : 1.6mA
Static Current        : 0.2mA
```

Table 10-4 Keywords Used in the Configuration File

Keyword	Description
Library name	The name of the library where the transistor-level analysis is run. The library contains the PARA view (where the `poTxPowerAnalysis` result is saved) and the RAIL view (where the DWM data is to be saved).
Cell name	The name of the hard macro cell that has been analyzed by `poTxPowerAnalysis`.
Power net and ground net	The names of the power and ground nets that need to be modeled, separated with a space or tab.
Input, output, and inout	The names of the input ports, output ports, and bidirectional ports.
Define mode	Define the operation mode to be read or write.
Trigger	Define which I/O port triggers the operation mode. A switching direction of rise (r) or fall (f) is followed by the port name.
Condition	Define the state when the mode is to be activated. It is represented by an in-fix Boolean expression. Tokens (', '), NOT(!), AND(&), and OR (|) are supported.
Cycle time	The minimum cycle time that a memory cell can operate. Sometimes it is also referred as “clock cycle time” in other documents.
Peak current	The peak current of a mode.

Table 10-4 Keywords Used in the Configuration File

Keyword	Description
Average current	The average current while the memory cell is operated at the operating frequency as specified in the DWM configuration file. It can also be referred as “AC current,” “total power,” or “total power per 100 Mhz” in other documents. Because sometimes different units may be applied, use “average current @ max frequency” instead.
Static current	An optional field to specify the time-average (leakage) current of the operation mode.
Access time	An optional field for designs with output port(s) only. The access time means the longest possible delay to validate an output when the trigger starts. It can also be referred to as “clock to output delay” in other documents. That whether the access time is defined results in different current waveforms.

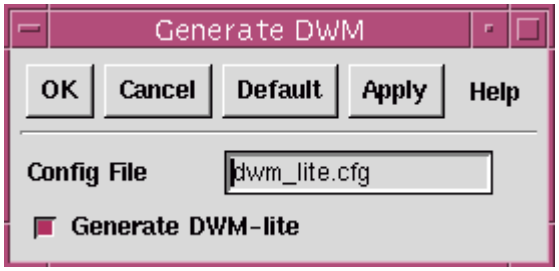
Generating DWM-Lite Models

To run the DWM-lite flow to generate a hard macro model:

1. Enter `poTxGenDWM` or choose Macro Modeling > DWM Generation – Generate DWM.

   The Generate DWM dialog box opens.

Figure 10-9 Generate DWM Dialog Box



2. Enter the name of the configuration file that contains the necessary information for modeling a hard macro.

   By default, the configuration file is named as dwm_lite.cfg.

3. Select DWM-lite and click OK or Apply.

Generating Macro Models, Using the DWM Flow

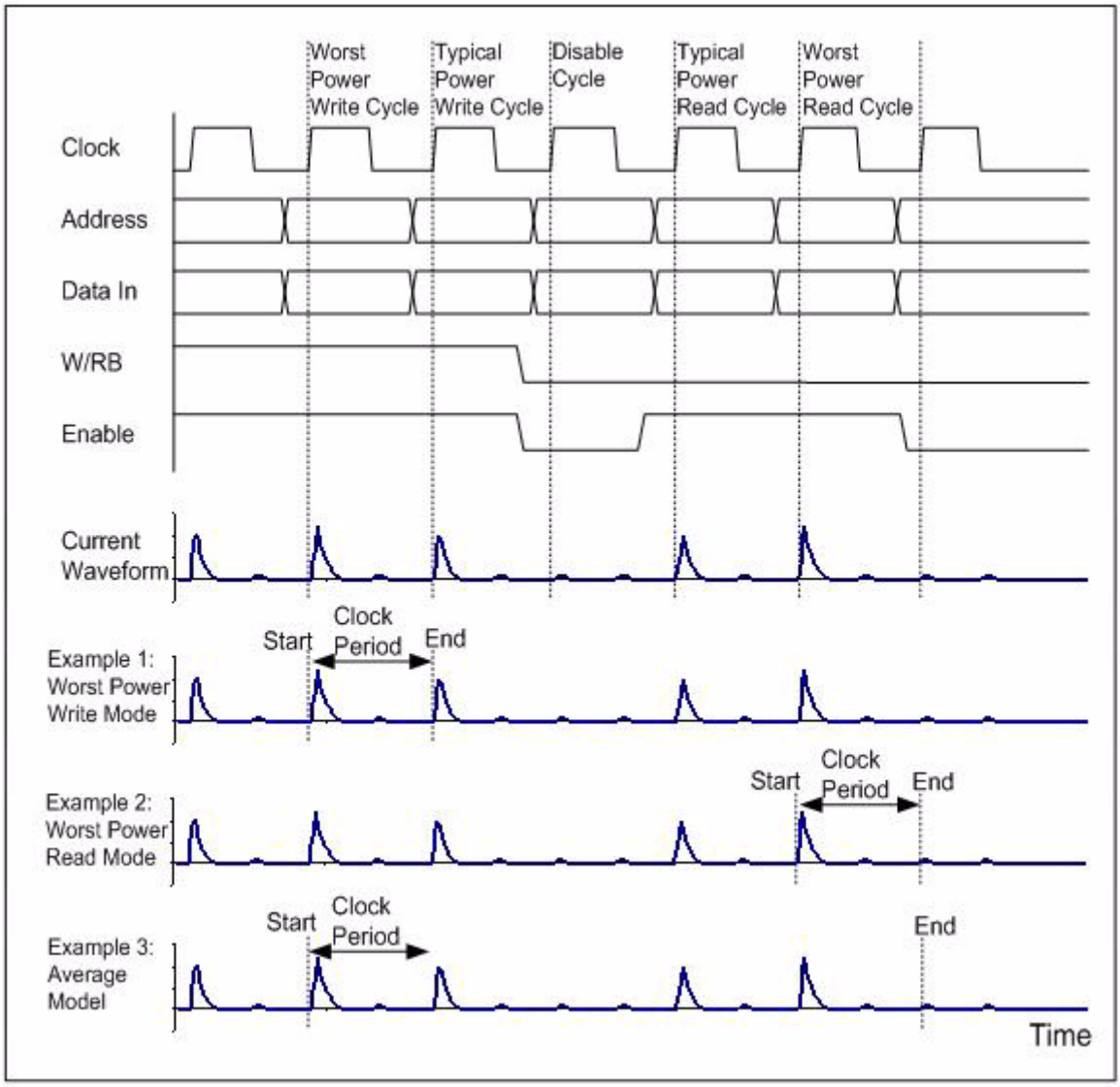
Because the macro model generated through the DWM flow contains current waveforms for each of the current source locations, the `poCalculatePower` and `poRailAnalysis` commands will use the current waveforms directly. All the power values calculated by PrimeTime PX or assigned by users will be ignored.

Dynamic White Box Model Technology

Before modeling a hard macro in PrimeRail, you need to perform transistor-level power analysis to capture the dynamic current waveforms at each device terminal that is connected to power and ground nets for the complete simulation time. PrimeRail then cuts the captured dynamic current waveforms into segments based on the start time and end time you specify in the configuration file.

Figure 10-10 explains how PrimeRail records and saves the dynamic waveforms to the resulting macro model.

Figure 10-10 Characterizing Waveforms for Multiple Operation Modes

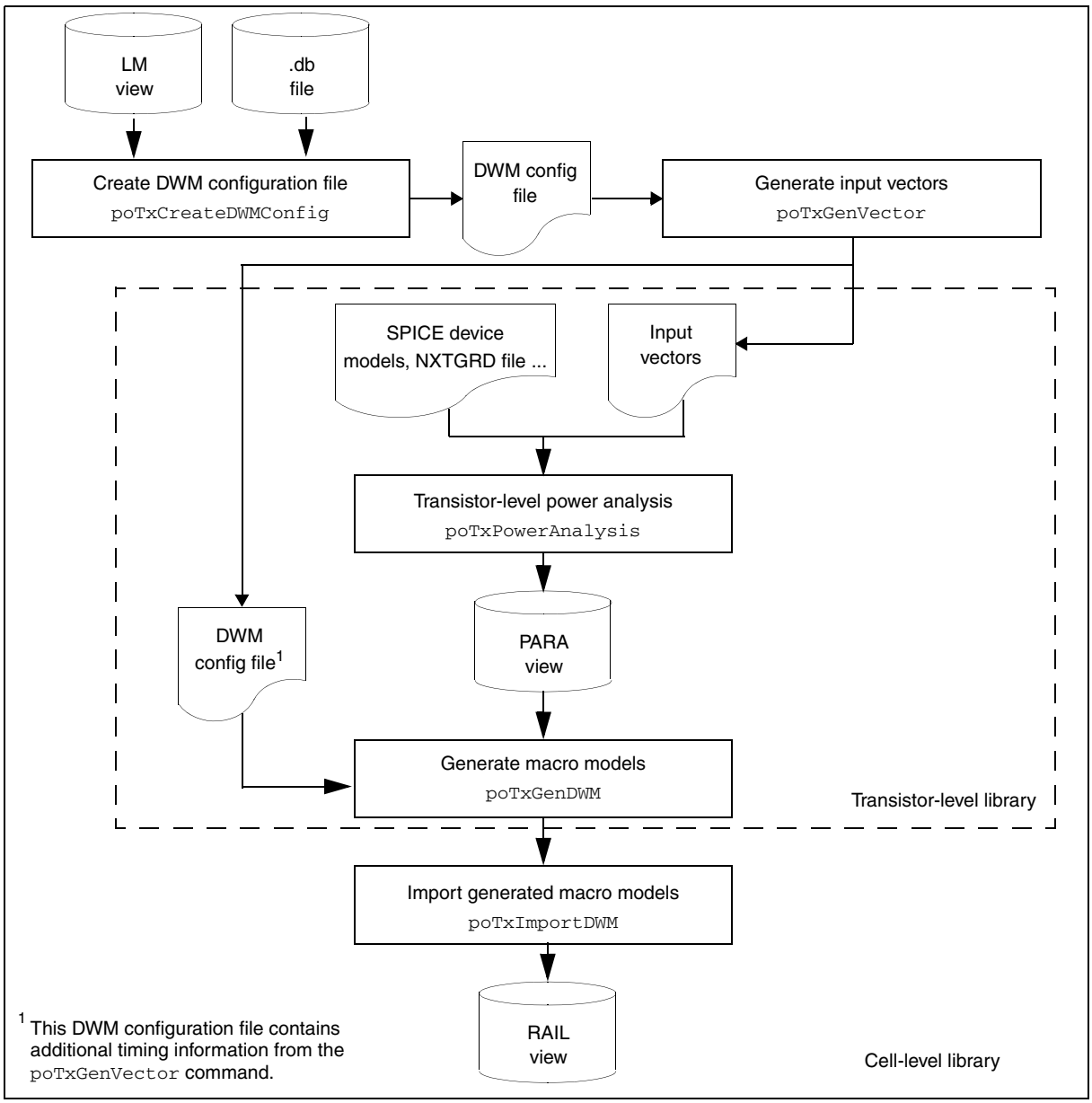


In Figure 10-10, five operations are performed to demonstrate the current waveform variation of a block. The operations are worst-power write cycle, typical-power write cycle, disable cycle, typical-power read cycle, and worst-power read cycle. The generated macro model contains the worst-power write mode and worst-power read mode.

DWM Generation Flow

Figure 10-11 illustrates the files and steps involved in the dynamic hard macro model generation flow.

Figure 10-11 DWM Design Flow



Creating a DWM Configuration File

Generating a hard macro model in PrimeRail requires a configuration file that contains basic information about the hard macro being modeled, primary input and output ports, and mode configuration. PrimeRail provides the `poTxCreateDWMConfig` command that automatically

creates a DWM configuration file based on the information in the LM view or .db files. Alternatively, you can manually create one if you want the flexibility of controlling how DWMs are generated by modifying the configuration file.

Because LM views are created through the Synopsys .lib files, which are usually provided by the IP vendor or the standard cell creator, some information in the .lib file is required to correctly generate a DWM configuration file, such as I/O pins, operation modes, and power numbers. The `poTxCreateDWMConfig` command reads the information defined in the internal_power( ) section, including the input pin name, the when condition, and the rise_power, fall_power, or power table.

The following is an example of the internal_power() section in the .lib file:

```
internal_power(){
           when : "!CEN & WEN";
           rise_power(ram16x2048_energy_template) {
                 index_1 ("0.5 2.0");
                 values ("38.59, 38.59")
           }
           fall_power(ram16x2048_energy_template) {
                 index_1 ("0.5 2.0");
                 values ("0, 0")
           }
}
```

PrimeRail only captures internal powers that are triggered by clock pins. The `poTxCreateDWMConfig` command ignores the operations with power values of zero or those much smaller than other modes. These small power modes will be printed as command lines with their power number as written in the LM view. It is nice to have power numbers in the library; the numbers can be rough estimations but cannot be totally meaningless. In the case that the vendor does not characterize the power number correctly, you can choose to manually modify the configuration file. It is always recommended that you double check the configuration file before proceeding to the next step.

Automatically Generating a DWM Configuration File

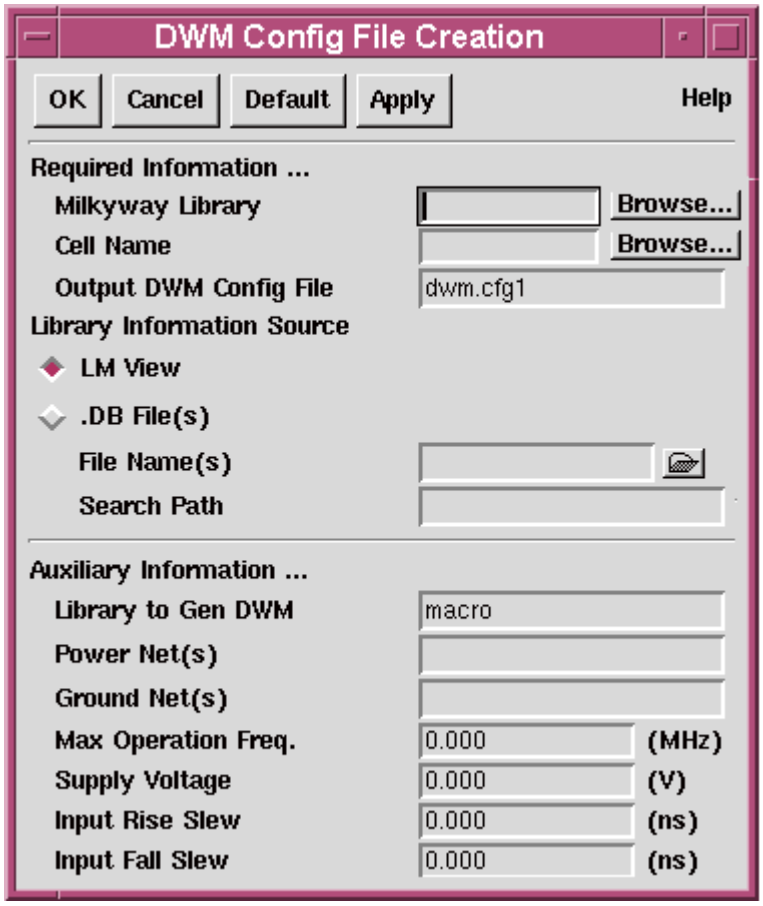
PrimeRail provides the `poTxCreateDWMConfig` command to automatically grab information from the Milkyway library and create the DWM configuration file with the necessary information filled. Similar to standard cells, the timing arc and power information of a macro cell are saved in the LM view (created by importing the .lib file) or the .db files.

To have the tool automatically generate a DWM configuration file for a macro cell:

1. Enter `poTxCreateDWMConfig` or choose Macro Modeling > DWM Generation – Create DWM Config File.

   The DWM Configuration File Creation dialog box appears.

Figure 10-12 DWM Config File Creation Dialog Box



2. Enter the name of Milkyway library where the macro cell is located.
3. Enter the name of the macro cell.
4. Enter the name of the DWM configuration file to be generated. By default, the file is named as *dwm.cfg1*.
5. In the Library Information Source section, choose if the information is to be read from the LM view or the .db file. If the information is available in the .db file, enter the name and search path of the .db file being used.
6. In the Auxiliary Information section, specify the supplementary information that is not in the LM view or the .db file but is required by the macro model generation. You can choose not to fill in the fields since generating a configuration file does not require this supplementary information. However, it is recommended that you manually enter the information to the configuration file once it is generated in order to successfully proceed to the following steps of running the `poTxGenVector` and `poTxGenDWM` commands.

- Library to Gen DWM – The library where the DWM model will be generated. Because PrimeRail captures the information from the transistor-level power analysis results for the DWM generation, this field should be the name of the library where the transistor-level analysis is run or is to be run.

  Note that it is possible that you have not executed the `poTxPowerAnalysis` command at this stage, and thus no transistor-level library is available. If this is the case, leave this field blank.

  If you are running the DWM-lite generation, enter the name of the LM view library as specified earlier in that the DWM-lite macro model will be created in the reference library.

- Power Net(s) – The names of the power nets that need to be modeled, separated with a space or tab.
- Ground Net(s) – The names of the ground nets that need to be modeled, separated with a space or tab.
- Max Operation Freq. – The maximum operation frequency, in megahertz.
- Supply Voltage – The supply voltage, in volts.
- Input Rise Slew – The time duration when input signals rise. When set to zero, the tool uses the default input slew from NanoSim (or HSIM). Setting the input rise slew is optional.
- Input Fall Slew – The time duration when input signals fall. When set to zero, the tool uses the default input slew from NanoSim (or HSIM). Setting the input fall slew is optional.

7. Click OK or Apply.

   The tool creates the DWM configuration file in the working directory. You can then run the `poTxGenDWM` command to create a DWM model. Note that if neither “supply voltage” nor “temperature” is specified in the configuration file, running the `poTxGenDWM` command will create a model that does not support multiple PVT corners.

   For more information about creating a macro model, see “Generating DWM Models” on page 10-44.

Manually Creating a DWM Configuration File

The DWM configuration file contains the information required to model a hard macro in PrimeRail. In addition to running the `poTxCreateDWMConfig` command to have the tool automatically generate a DWM configuration file for a macro cell, you can manually create a configuration file.

The following is an example of the configuration file. To comment a line, start the line with one of the following symbols: asterisk (*), slash (/), semicolon (;) or pound sign (#). Each value in the configuration file is case-insensitive.

```
* Basic information
Library Name     : DWM
Cell Name        : ram_test
Power Net        : VDD
Ground Net       : VSS

Supply Voltage: 1.8V
Temperature      : 27.00
Max Frequency    : 200MHz
Rise Time        : 50ps
Fall Time        : 50ps

* IO information
Input            : A[10:0]
Input            : CLK
Input            : CEB
Input            : WEB
Input            : OEB
Inout            : D[15:0]

* Operation mode information
Define Mode      : Read Mode
Trigger          : CLK      rise
Condition        : (!CEB & WEB &!OEB)
Start Time       : 25ns
End Time         : 35ns

Define Mode      : Write Mode
Trigger          : CLK      rise
Condition        : (!CEB & !WEB)
Start Time       : 5ns
End Time         : 25ns
```

Table 10-5 Keyword Explanation for the DWM Configuration File

Keyword	Description
Library name	The name of the library where the transistor-level analysis is run. The library contains the PARA view (where the `poTxPowerAnalysis` result is saved) and the RAIL view (where the DWM data is to be saved).
Cell name	The name of the hard macro cell that has been analyzed by `poTxPowerAnalysis`.
Power net and ground net	The names of the power and ground nets that need to be modeled, separated with a space or tab.

Table 10-5 Keyword Explanation for the DWM Configuration File (Continued)

Keyword	Description
Temperature	The PVT corner of the DWM to be generated. The default values for P, V, and T are 1.00, 0.00 (in unit volts), and 270 (in degree Celsius), respectively. The value of the P corner is fixed to 1.00. This is because the tool currently does not handle P corners during power and rail analyses. Note: You can create a configuration file for a corner at a time.
Input, output, and inout	The names of the input ports, output ports, and bidirectional ports.
Define mode	Define the operation mode to be read or write.

Trigger	Define which I/O port triggers the operation mode. A switching direction of rise (r) or fall (f) is followed by the port name. For example:
	`Trigger : WE rise`
	When analyzing an asynchronized design and want to specify multiple signals, define the Trigger keyword in the following format:
	`Trigger : Ar[11:0] 12`
	The first part is a bus expression and the last part tells how many bits are toggling.
	The following are examples of input signals to explain the operation of typical asynchronous memories:
	Aw[11:0] – Address for the write port
	Dw[7:0] – Data in for the write port
	WE – Write enable, positive edge trigger
	Ar[11:0] – Address for the read port
	Qr[7:0] – Data out for the read port
	When WE is switching from low to high, the address and data on Aw and Dw ports will be latched and written to the addressed memory cell. It is synchronized by WE, so the write mode is a synchronized mode. However, there is no one single control signal to control the read port. As soon as the address on Ar port is changing, the Qr port output corresponding data right away.
	See “Setting Asynchronized Mode” on page 10-41 for more information.

Table 10-5 Keyword Explanation for the DWM Configuration File (Continued)

Keyword	Description
Condition	Define the state when the mode is to be activated. It is represented by an in-fix Boolean expression. Tokens (', '), NOT(!), AND(&), and OR (|) are supported.
Start time, end time	Define how the current waveform is to be captured when a hard macro is modeled. See "Dynamic White Box Model Technology" on page 10-33 for more information.

Setting Asynchronized Mode

The main difference between synchronized and asynchronized modes is the trigger. The former has a single trigger signal; the latter allows multiple signals. The following are examples of specifying multiple signals for asynchronized mode in the Trigger field:

```
Define mode:    asyn_read
Trigger :       Ar[11:0] 12
```

The first part of the Trigger field is a bus expression; the last part specifies how many bits are toggling.

For synchronized mode, you must specify the switching direction for the trigger, as in the following example:

```
Trigger :       WE rise
```

For asynchronized mode, trigger switching is neither considered nor allowed. The tool only counts how many bits out of the triggers are toggling; different toggling counts result in different power consumptions. Therefore you can generate different modes with different toggling counts for some switching groups, as in the following example:

```
Define mode:    asyn_read_tog12
Trigger :       Ar[11:0] 12
Define mode:    asyn_read_tog8
Trigger :       Ar[11:0] 8
Define mode:    asyn_read_tog4
Trigger :       Ar[11:0] 4
```

While running power and rail analyses, PrimeRail will check the VCD input and select the corresponding relatively pessimistic mode. For example, if there are 7 input signals toggling in a short period of time, mode asyn_read_tog8 will be triggered.

It is required to generate at least one mode for a trigger group. If there is only one mode for a trigger group, this mode has to toggle all the signals in this trigger group to maintain pessimism.

You can specify the switching group in the following syntax.

```
Trigger :       A[0] A[1] A[2] A[3] 4
Trigger :       CEB A[10:0] 4
```

The keyword *Condition* is supported for asynchronized modes.

Here is a sample DWM configuration file:

```
* Basic information
Library Name:         DWM
Cell Name  :          ram_8x4096
Power Net  :          VDD
Ground Net :          VSS
Supply Voltage:       1.8V
Max Frequency:        200MHz

* IO information
Input            : Aw[11:0]
Input            : Dw[7:0]
Input            : CE
Input            : WE
Input            : Ar[11:0]
Output           : Qr[7:0]

* Operation mode information
Define Mode:          Write
Trigger    :          WE   rise
Condition  :          CE
Define Mode:          Read_Tog_12
Trigger    :          Ar[11:0] 12
Condition  :          CE
Define Mode:          Read_Tog_8
Trigger    :          Ar[11:0] 8
Condition  :          CE
Define Mode:          Read_Tog_4
Trigger    :          Ar[11:0] 4
Condition  :          CE
```

Automatic Input Vector File Generation

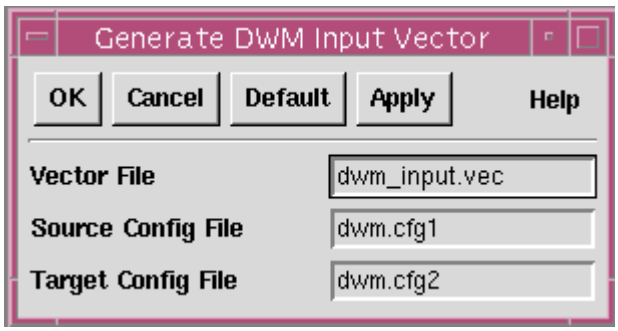
Use the `poTxGenVector` command to automatically generate the input vector file that is needed by the NanoSim simulation. When this vector file is created, you can use it during transistor-level power analysis by entering the NanoSim -t command in the Run Option text field of the Tx Power Analysis (see Figure 9-5 on page 9-12).

To automatically generate input vector files,

1. Enter `poTxGenVector` or choose Macro Modeling > DWM Generation – Generate Input Vector.

The Generate DWM Input Vector dialog box appears, as shown in Figure 10-13.

Figure 10-13 Generate DWM Input Vector Dialog Box



2. Enter the name of the vector file to be generated. By default, the file is named as dwm_input.vec.
3. In the Source Config File text field, enter the name of the DWM configuration file that you created previously. PrimeRail reads the information from this configuration file and generates an input vector file for the NanoSim run. PrimeRail also updates the timing window information for each mode in the DWM configuration file, and writes the information to a new file specified in the Target Config File text field.

   By default, the file is named as dwm.cfg1.
4. In the Target Config File text field, enter the name of the configuration file to which PrimeRail writes the updated timing window information. When not specified, PrimeRail overwrites the original information with the updated information.

   By default, the file is named as dwm.cfg2.
5. Click OK or Apply.

   The difference between the source and target configuration files is the start time and end time of the timing window. Because this command creates an input vector file automatically, it has to indicate the timing window to record the current waveform according to the way PrimeRail generates the input vectors. If start time and end time are specified in the source configuration file, they will be ignored and updated in the target configuration file.

   A signal is called deterministic if it is defined as a trigger or control signal in any operation mode. The program automatically generates the sequence that triggers every operation mode for deterministic signals. For the non-deterministic signals, PrimeRail arbitrarily chooses an initial state and changes every bit of them before every operation mode. This policy leads to a high possibility of triggering the maximum power consumption operation for all modes.

Generating DWM Models

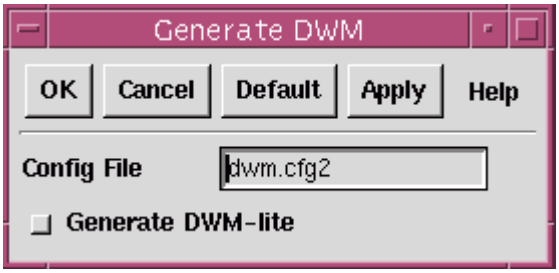
When the transistor-level power analysis is complete and the DWM configuration file is available, run `poTxGenDWM` to create a DWM model for a hard macro cell in the design. During the creation, the command uses the parasitic and waveform data stored in the PARA view generated by `poTxPowerAnalysis` for modeling and then saves the data in the RAIL view for top-level dynamic rail analysis.

To model a hard macro:

1. Enter `poTxGenDWM` or choose Macro Modeling > DWM Generation – Generate DWM.

   The Generate DWM dialog box appears, as shown in Figure 10-14.

Figure 10-14 Generate DWM Dialog Box



2. Enter the name of the configuration file that contains the necessary information for modeling a hard macro. By default, the file is named as dwm.cfg2.
3. Click OK or Apply.

   Note:
   Do not select the Generate DWM-lite option unless you are running the DWM-lite flow. For more information about running the DWM-lite flow, see “Generating Macro Models, Using the DWM-Lite Flow” on page 10-26.

The generated macro models are saved in the RAIL view. When you generate all the hard macro models, run the `poTxImportDWM` command to import the RAIL view from the transistor-level run library to the cell-level reference library where the CONN view of the macro cell is located. For more information on how to import the generated hard macro models before performing a full-chip analysis, see “Importing Macro Models” on page 12-2.

When the generated macro models are imported to the cell-level reference library, PrimeRail uses the generated macro models in the full-chip analysis flow, including waveform generation, power and ground net extraction, and rail analysis.

Log Message

The following is an example of the log message when the macro model generation is done.

```
; poTxGenDWM "dwm1.cfg"
```

```
Generate DWM: Finish parsing DWM config file dwm1.cfg

DWM Summary:
 P/V/T corner: 1.00 / 1.50 / 27.00
 cell/net names: sram1056x12 VDD
 total PG wire capacitance: 14.743pF
 total intrinsic capacitance: 5.845pF
 # of current sources: 30656
  mode (peak/avg total current): Read Mode (-24.68/-2.36mA within 6.41ns)
   mode (peak/avg total current): Write Mode (-37.32/-3.10mA  within
6.41ns)

DWM Summary:
 P/V/T corner: 1.00 / 1.50 / 27.00
 cell/net names: sram1056x12 VSS
 total PG wire capacitance: 21.215pF
 total intrinsic capacitance: 4.89pF
 # of current sources: 30037
 mode (peak/avg total current): Read Mode (33.71/1.90mA within 6.41ns)
   mode (peak/avg total current): Write Mode (36.80/2.49mA within 6.41ns)

DWM: finish generate DWM for cell sram1056x12
```

Browsing the Generated Macro Models

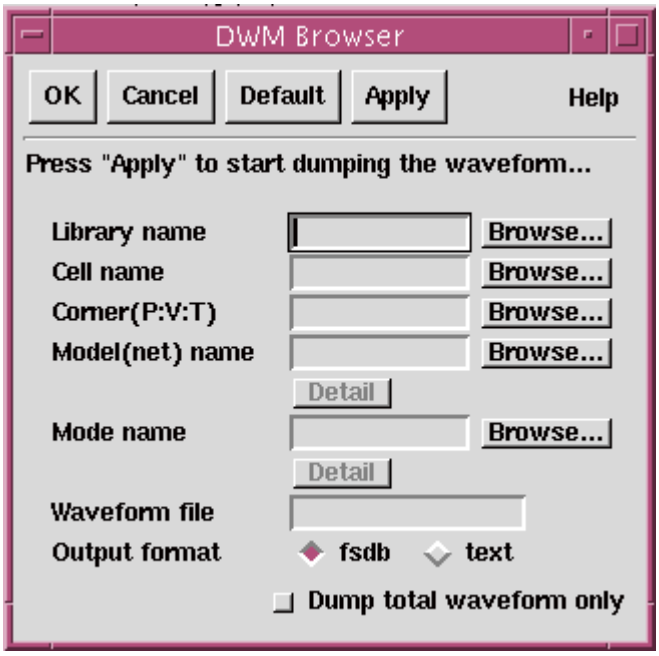
Run the `poTxBrowseDWM` command to check the information inside a generated dynamic macro model.

To browse and check the generated macro models,

1. Enter `poTxBrowseDWM` or choose Macro Modeling > Auxiliary Commands – Browse DWM.

   The DWM Browser dialog box appears, as shown in Figure 10-15.

Figure 10-15 DWM Browser Dialog Box



2. Enter the names of the library and cell where the DWM is located.
3. Enter the corner of the cell to be opened. If the cell contains multiple PVT corners, click Browse and select a corner in the Browse PVT Corner dialog box that opens. The Corner (P:V:T) text box will be empty if the model does not contain a PVT corner.

   Click Hide to return to the DWM Browser dialog box.
4. Enter the name of the macro model to be checked. Alternatively, you can click the Browse button and choose a model from the Browse Model dialog box that appears. If you want to write out the summary information for the model, click Detail.
5. Enter the operation mode to be checked. Alternatively, you can click the Browse button and select an operation mode from the Browse Mode dialog box that appears. If you want to write out the summary information for the operation mode, click Detail.
6. If you want to write the current waveforms of a mode to a file, enter a name in the Waveform File text field.
7. Specify the format of the output waveform file to be either FSDB or ASCII. Check the “Dump total waveform only” option if you want to write out only total current waveform for a mode. Otherwise, all the individual current waveforms will be written to the output file.
8. Click Apply.

Deleting the Generated Macro Models

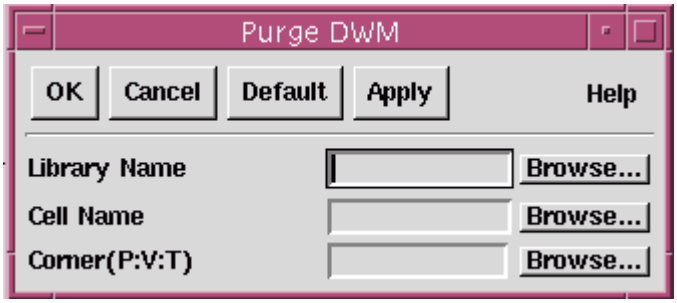
Run the `poTxPurgeDWM` command to purge the DWM database from the Milkyway database.

To remove the generated dynamic macro models from the database,

1. Enter `poTxPurgeDWM` or choose Macro Modeling > Auxiliary Commands – Purge DWM.

   The Purge DWM dialog box appears, as shown in Figure 10-16.

Figure 10-16 Purge DWM Dialog Box



2. Enter the names of the library and the cell from which the dynamic macro models are to be removed.
3. Enter the value of the PVT corner to be purged. If the cell contains multiple PVT corners, click Browse and select one in the Browse PVT Corner dialog box that opens. If a corner is selected, the tool will purge only the specified corner in the DWM. If no corner is specified, all the corners in the DWM will be purged. The Corner (P:V:T) text box will be empty if the model does not contain a PVT corner.

   Click Hide to return to the Purge DWM dialog box.
4. In the Purge DWM dialog box, click OK or Apply.

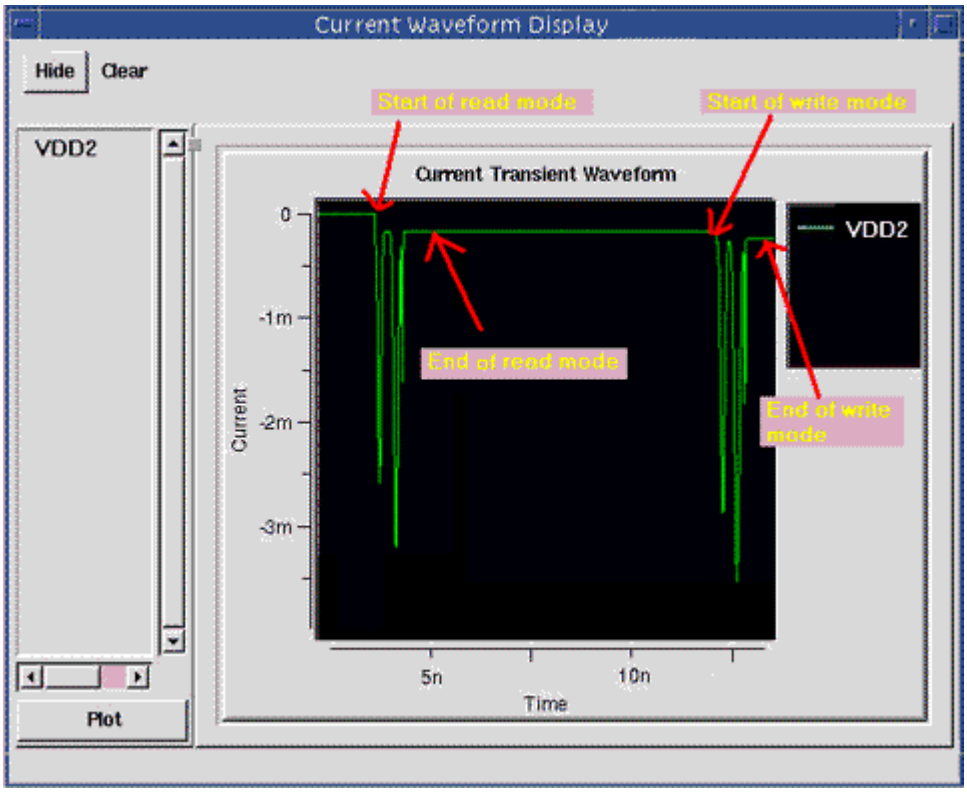
DWM Leakage Handling

When performing rail analysis, you specify the start time, end time, and step size for simulating the power and ground network. The generated waveforms are thus cut into several segments by modes. When the time reaches the end of the waveform of this mode, the rail analysis engine extends the waveform until the next mode is triggered. During the DWM (and DWM-lite) flow, PrimeRail treats the tail of current waveforms as leakage.

As shown in Figure 10-17, a design contains a read mode and a write mode; both modes have 5 ns long of waveforms. The current at the end point of the read mode is 0.2 micro ampere (uA), and that at the end point of the write mode is 0.3 uA. During the rail analysis,

the read mode and the write mode are exited at 8 ns and 23 ns, respectively. Then a constant 0.2 uA of leakage current between 13 ns and 23 ns is generated, and a constant current of 0.3 uA after 28 ns.

Figure 10-17 Handling Leakage Current During the DWM Flow



Note:

If the capturing DWM waveform ends at a point where the circuit is still active, the tail of the waveform will contain active currents. In this case, a huge fake “leakage” current will be projected to the rail analysis, which leads to incorrect analysis results.

Because no mode is triggered at the beginning of rail analysis, the leakage of the time period before the first mode is triggered is undefined. A zero current will be assigned to this period.

Importing Macro Models

After you run the `poTxGenDWM` command to characterize a hard macro cell as a macro model, run the `poTxImportDWM` command to import the macro models that are generated by `poTxGenDWM` to the cell-level reference library. You must import the generated macro models before you perform a full-chip analysis.

For more information about how to import hard macro models during full-chip analysis, see “Importing Macro Models” on page 12-2.

11

Analyzing Power Management Cells

A power management (PM) cell acts like a switch in the design with an internal resistive network. It takes one power and ground net as an input and provides this to an output power and ground net in a different name. The input net obtains power and ground net from a pad cell while the output net provides this power and ground net to the core region or another specific portion of the full-chip design.

You can define the resistance of the connection between the input net and the output net based on the specific power management cell being used. This resistance is virtual and there is no actual geometry involved with creating this resistance.

PrimeRail provides two ways of analyzing power management cells. One is power-on mode analysis that is to be run in the time-average analysis flow. The other is in-rush analysis (also called rush current analysis) in the dynamic analysis flow.

This chapter includes the following topics:

Capabilities and Limitations

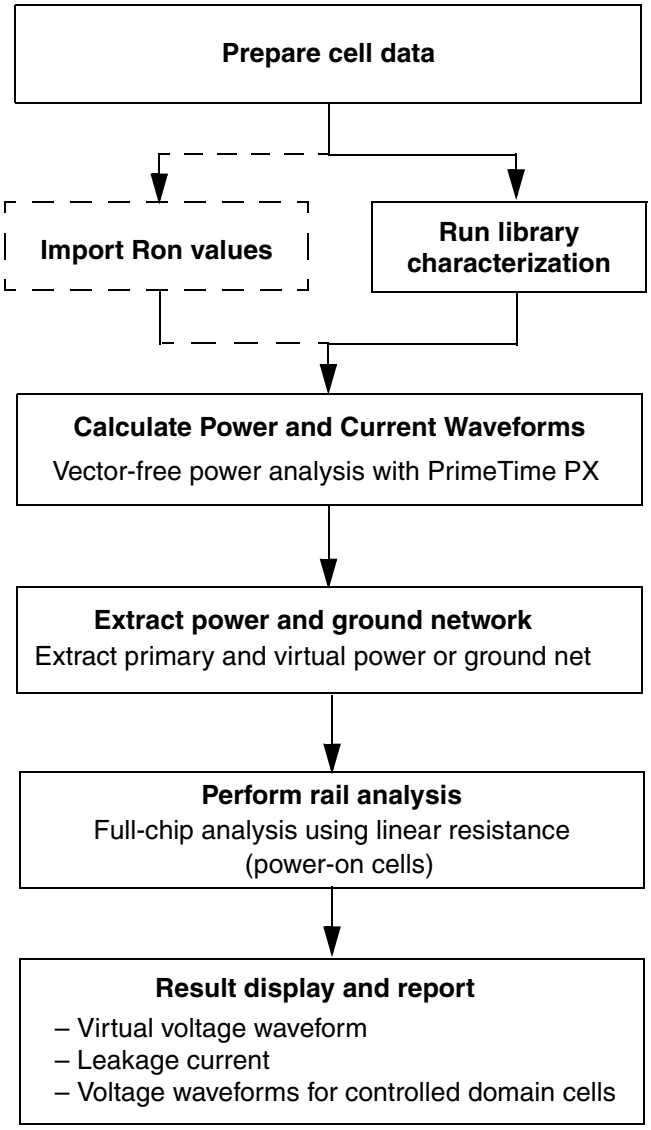
PrimeRail works with power management cells in the following manner:

- There can be multiple power management cells.
- There can be multiple instances of power management cells.

Power-On Mode Analysis

Figure 11-1 describes the design flow of the power-on analysis in the time-average flow. Different from the rush current analysis in the dynamic flow, library characterization is not required when performing a power-on mode analysis. If you do not want to run library characterization on power management cells, run the `pgImportRonValue` command to import the R_{on} values to the tool. The R_{on} values are the ON resistance values on IV curves.

Figure 11-1 Power-On Mode Analysis Design Flow



Preparing Cell Data

In the power-on mode analysis, you can choose to run library characterization for modelling the power management cells for generating characteristics of the multithreshold-CMOS cells and their constraints. If library characterization results are not available or you do not plan to run library characterization for acquiring IV characteristics, run the `pgImportRonValue` command for importing R_{on} values to the tool. The R_{on} values are the ON resistance values on IV curves.

Here is the syntax:

```
pgImportRonValue PMcellMastName controlPinName vddPinName \
  virtualvddPinName RonValue
```

The R_{on} value is in unit of Ohm.

Note:

When importing the R_{on} values, you need to open the reference library where the power management cell resides. If you want to remove the imported values, run the `pgPurgeCharacterize` command and select all the options on the dialog box (including the PM Cell option) to clean up all the analysis results that are related to the power management cell.

For details about library characterization, see Chapter 6, “Library Characterization for Cell-Level Dynamic Analysis,” in this user guide.

Calculating Power and Current Waveforms

When power management cells are modeled, run time-average power analysis and generate time-average current waveforms using the `poCalculatePower` command. For time-average analysis, all power management cells will be at ON state.

For details about calculating power current waveforms, see “Calculating Power and Current Waveforms” on page 7-13.

Extracting Power and Ground Network

Run the `poExtractPGParasitics` command to extract the resistance and capacitance data for the power and ground nets and virtual power and ground nets of the PM cell.

For more information about extracting parasitic data, see “Extracting Power and Ground Parasitics” on page 7-5, in this user guide.

Performing Rail Analysis with PM Cells

The last step in the power-on mode analysis is to run the `poRailAnalysis` command to analyze the reliability effects at the full-chip level.

To perform rail analysis,

1. Enter `poRailAnalysis` or choose Cell-level Analysis > Rail Analysis – Rail Analysis to open the P/G Rail Analysis dialog box.

2. In the “P/G net info” section, select Combined and enter the names of the nets to be analyzed.

   If the cell being analyzed is of cell type macro, select Power or Ground and specify a net to be analyzed. Selecting this option will also attach the virtual net to the main net.

3. In the “Hierarchical options” section, select “Flatten hierarchical cells” to flatten the hierarchical cells.

4. In the “Analysis options” section, select Time-Average.

   Specify other options as necessary.

   For more information about running the `poRailAnalysis` command, see the Help for the command.

5. Click OK or Apply.

For more information about rail analysis, see “Cell-Level Rail Analysis” on page 7-22.

Sample Script

The following is a sample script that lists the standard steps for running a multithreshold-COMS analysis with IC Compiler output rail setup data.

```
;# Scheme
geOpenLib
setFormField "Open Library" "Library Name" "mw_ctop"
formOK "Open Library"

geOpenCell
setFormField "Open Cell" "Cell Name" "primerail"
formOK "Open Cell"

poPurgeRail
formOK "Purge RAIL View"

poLoadRailSetup
formOK "Load Rail Setup"

poLinkPGNetlist

poReportRailSetup

poCalculatePower
formOK "Power & Current Analysis"

poExtractPGParasitics
setFormField "Extract PG Parasitics" "PG Net Name" "VDD VDDY VDDX VDDG
VDDGS VDDMS VDDXS VDDYS VSS"
formOK "Extract PG Parasitics"
```

```
poRailAnalysis
setFormField "P/G Rail Analysis" "Top-level design pin" "1"
setFormField "P/G Rail Analysis" "User-defined tap" "0"
setFormField "P/G Rail Analysis" "Analysis Option" "Time-average"
setFormField "P/G Rail Analysis" "Hierarchical Option" "Flatten
hierarchical cells"
setFormField "P/G Rail Analysis" "Extract Net" "Combined"
setFormField "P/G Rail Analysis" "Combine Nets" "VDD VDDY VDDX VDDG VDDGS
VDDMS VDDXS VDDYS VSS"
formOK "P/G Rail Analysis"

menuQuit
formYes "Dialog Box"
formButton "Save Cells" "saveAll"
formOK "Save Cells"
```

In-Rush Analysis

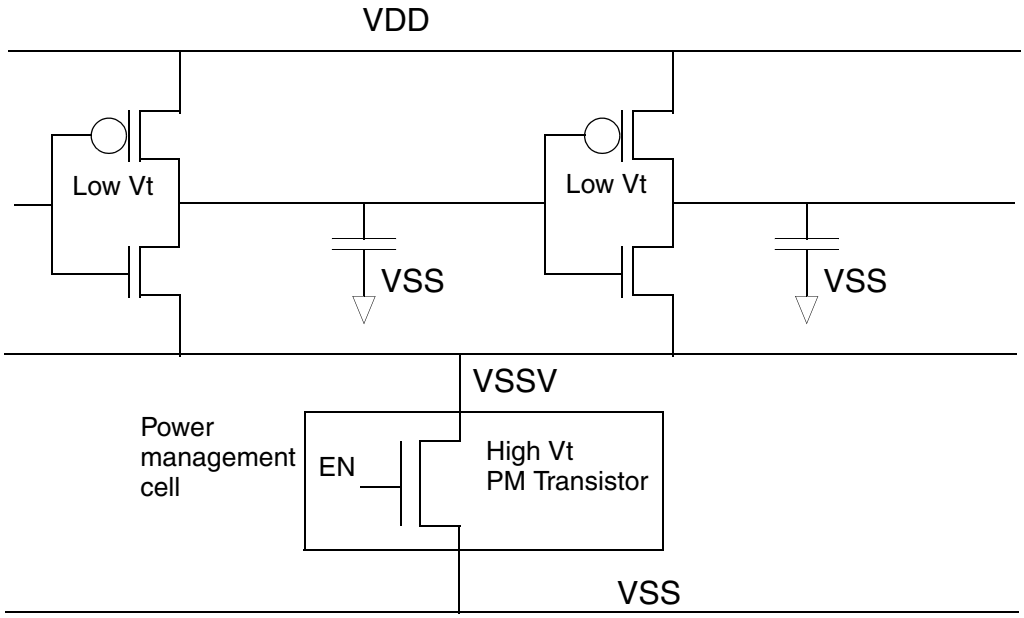
For CMOS circuits, total power consists of dynamic and time-average components. Dynamic power is proportional to the square of supply voltage and therefore lowering the supply voltage is the most effective way of reducing the dynamic power consumption. With scaling of the supply voltage, transistor threshold voltage also needs to be scaled in order to satisfy the performance requirements. Unfortunately, such scaling leads an exponential increase in transistor leakage current.

Multithreshold CMOS (MT-CMOS) technology is one of the design techniques for reducing leakage power. In multithreshold-CMOS circuits, power management cells (PM cells) with high threshold voltage (V_t) transistors are used to reduce the leakage power of power-managed functional blocks when they are inactive. However, the number of the power management cells in a real design can be large and turning them on (or off) all at the same time can draw lots of currents from the power grid. You must carefully control the power-up sequence of those power management transistors to minimize simultaneous switching noise when the circuits are turned on for normal operation.

PrimeRail provides the in-rush analysis feature to control the simultaneous switching noise by turning on these circuits sequentially. Accordingly, the number of power management cells, their drive strength, and timing sequence of the control signals are the design parameters that must be optimized and verified through accurate circuit simulation.

Figure 11-2 is an example of a multithreshold-CMOS circuit where a high V_t power management transistor is used for the ground net. It can also be applied to a power net.

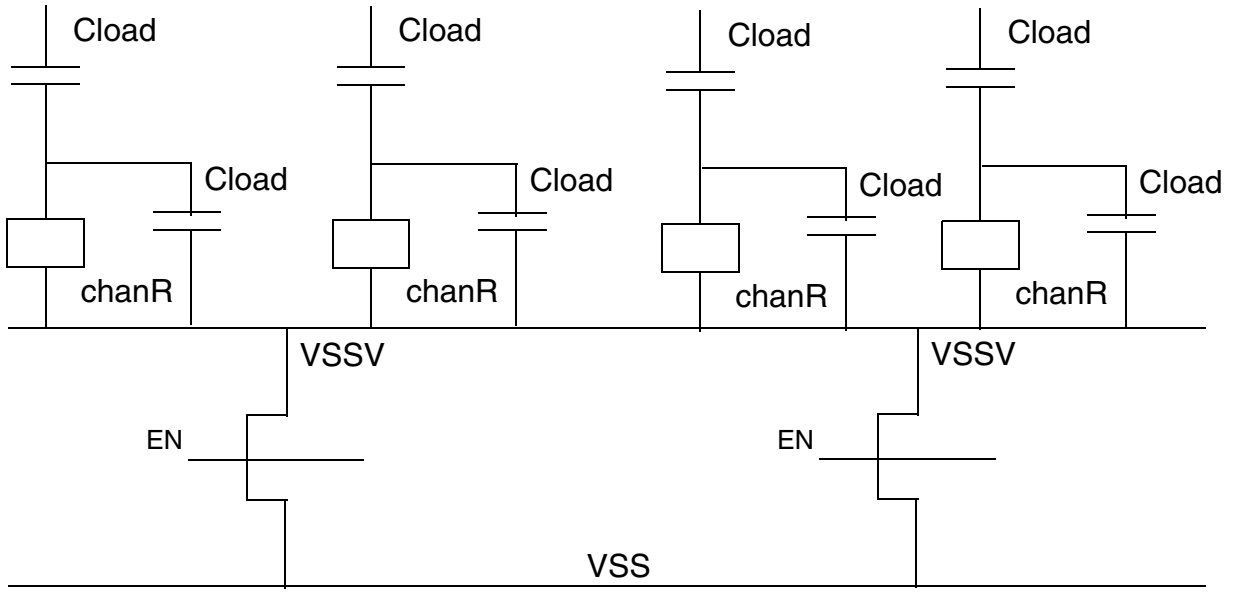
Figure 11-2 A Multithreshold-CMOS Circuit



When power management transistor (High Vt) is turned off in the inactive mode, the virtual ground net (VSSV) as well as internal logic nodes will be set to logic state 1. When the power management transistor is turned on for a normal operation, the virtual ground net (VSSV) reaches the VSS voltage level eventually. The time for the VSSV net to reach VSS is called “wake-up” time. The logic nodes will be at 0 or 1, which is dependent on the primary inputs. For the internal logic nodes of state 0, discharging current would flow through the circuit’s NMOS transistors, the power management transistor, then eventually to VSS. For the internal logic nodes charging to 1, charging current would flow from VDD to those nodes. A significant amount of short-circuit current also flows from VDD to VSS through the power management transistor. The sum of the discharging current and the short-circuit current is called “rush current.” This rush current behavior and wake-up time needs to be analyzed using PrimeRail.

Because the prediction of the individual node-discharging pattern is very computation intensive, PrimeRail provides the following heuristically scheme to estimate the rush current, as well as wake-up time, as shown in Figure 11-3.

Figure 11-3 The heuristically scheme to estimate the rush current and wake-up time



The scheme is to estimate how much the rush current is after power management cells are turned on. The total charge from each internal logic node is calculated by the following formula:

```
supply x Cload
```

where supply is the supply voltage value and Cload is the load capacitance of fan-out gate of a standard cell. If the filler cells are used as decoupling capacitors, then the decoupling capacitance value is used as Cload for filler cells.

Because initially all internal logic nodes are at supply voltage, discharging through multithreshold-CMOS cell would be the major concern when multithreshold-CMOS cells are turned on. Later, both PMOS and NMOS path would be open for some nodes, then “short circuit” current kicks in. For this reason, a scale factor is provided to compensate the missing “short circuit current” effect.

Using the above simple model, the relative discharging rate of all internal signal nodes can be used to provide a rough estimation on the “short circuit” effect. The maximum voltage difference among all standard cells at any given time is recorded. The bigger the maximum voltage difference is and the longer the maximum voltage difference stays above the large value, the bigger the “short circuit” effect will be. By this way, the tool provides the information of the uneven discharging pattern of controlled power and ground network. The typical channel resistance value of the standard or decoupling capacitance cells are used to calculate the discharging rate.

Limitation:

Note that no current flows through any VDD ports in the scheme model because the “short circuit” current of standard cells in the multithreshold-CMOS turn-on sequence is intentionally ignored. This is the limitation of the current flow.

Design Flow

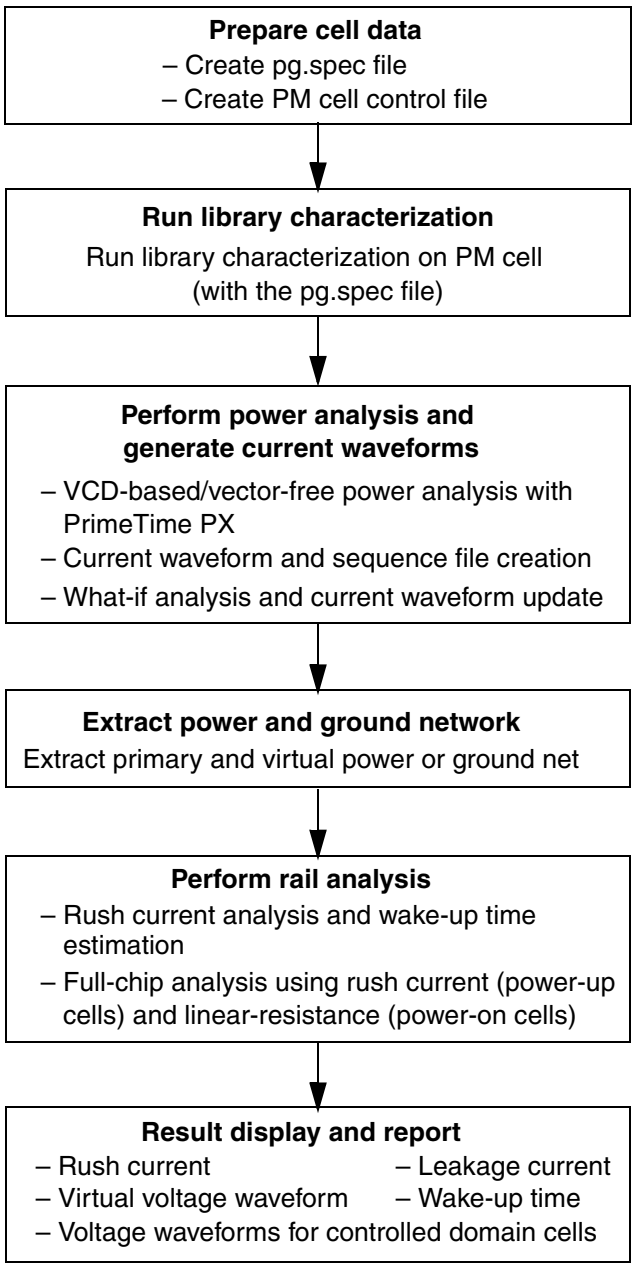
The first step in the in-rush analysis is to model the power management cells. To model the power management cells, you first need to run library characterization in order to generate the characteristics of the multithreshold-CMOS cells and their constraints. The resulting data will be used by the tool during runtime to capture rush current and wake-up time.

If you import R_{on} values for power management cells and do not run library characterization for a power-up sequence analysis, PrimeRail prints the following warning message:

```
Power Management Analysis Report

  Controlled voltage domain 2: VDD

WARNING : Voltage domain contains Ron channels. PM events are ignored and
rush current analysis result may be questionable.

    0 power management cell channels are used to control 769 std cells.
    Total OFF leakage current from PM cells is 0 (uA).
    Total ON leakage current from std cells is 2.124 (uA).
    No event is found, consider this voltage domain as ON.
```

Figure 11-4 illustrates the steps involved in running an in-rush analysis.

Figure 11-4 The Design Flow for the In-Rush Analysis Flow



Preparing Cell Data

Before characterizing the power management cells, you need to provide a PG spec file (that is, a SPICE netlist) for the header/footer cell (multithreshold-CMOS cell) and a PM cell control file.

- Preparing a PG Spec File
- Creating a PM Cell Control File

Preparing a PG Spec File

Before characterizing the cells, you need to add some power management cell specific statements to the PG spec file. This PG spec file is backward compatible, which means if the reference library contains both standard cells and power management cells, the same PG spec file with the added statements can be used.

The following is an example of the PG spec file, describing the constraints of the power management cell, with both header and footer cell depicted.

```
definePowerPort "TVDD" 1.08
defineGroundPort "TVSS" 0.0

defineDefaultPowerPort "TVDD"
defineDefaultGroundPort "TVSS"

defineGroundPowerMapping "TVSS" { "TVDD"  }

definePMCell {
    cell_name : PM_HEADER1
        iv_curve {
            leakage : 3.1e-10
            init_condition : (CTL1 0.0) (VDD 0.0) (TVDD 1.08)
            final_condition : (CTL1 1.08) (VDD 1.08) (TVDD 1.08)
            source_drain_pair : TVDD VDD
            control_pin : CTL1
        }
   }
}
definePMCell {
    cell_name : PM_FOOTER1
        iv_curve {
            leakage : 4.3e-10
            init_condition : (CTL2 0.0) (VSS 1.08) (TVSS 0.0)
            final_condition : (CTL2 1.08) (VSS 0.0) (TVSS 0.0)
            source_drain_pair : TVSS VSS
            control_pin : CTL2
        }
   }
}
```

Note that it is unnecessary to specify init_condition or final_condition for the internal pin, because the internal pin is not included in the sub-circuit pin name list. PrimeRail will issue an error message if a pin specified in init_condition, or final_condition is not included in sub-circuit pin name list.

For 2-control pin power management cells (that is, with two IV curves), use the similar method to define an internal pin for each IV curve.

The initial and final conditions to specify those two IV curves should be mutually exclusive. That means those two IV curves should be characterized independently. For example, a power management cell has two inputs, IN1 and IN2, to control two power management transistors 1 and 2. During in-rush analysis, IN1 is rising while IN2 is at static zero, so the power management transistor 1 is turned on. Some time later, IN1 will stay at static one, and IN2 is rising in order to turn on power management transistor 2. To characterize IV curves for those two power management transistors, the initial and final conditions for power management transistor 1 should be specified as:

```
init_condition (IN1 0) (IN2 0) (VVDD 0) (VDD 1.5)
final_condition (IN1 1.5) (IN2 0) (VVDD 1.5) (VDD 1.5)
```

The initial and final conditions for power management transistor 2 should be as:

```
init_condition (IN1 0) (IN2 0) (VVDD 0) (VDD 1.5)
final_condition (IN1 0) (IN2 1.5) (VVDD 1.5) (VDD 1.5)
```

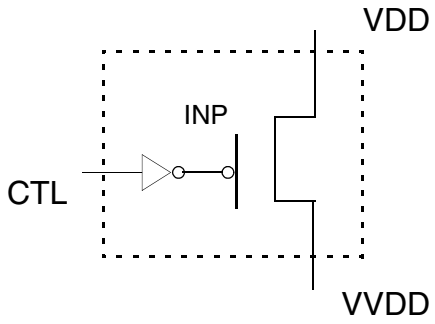
Note that although in the actual in-rush analysis, when the second power management transistor is turned on, pin IN1 already stays at static one. In this case, the pin should be specified with static zero condition when the second IV curve is characterized.

Internal Delays During Rush Current Analysis

If power management cells contain internal control logics, a delay from the outside input pin to the actual internal input of the power management cell can also be characterized. To consider internal delay/slew during rush current analysis, add the `internal_pin` option (with the node name from the SPICE subckt) in the PG spec file. Use the `internal_delay_thresholds` and `control_pin_input_slew` options if you want to customize the `internal_pin` option.

Figure 11-5 shows an example where the header cell has an inverter before the actual power management cell input.

Figure 11-5 An example of characterizing internal delays



The following is an example of the PG spec file, describing the constraints of the power management cell, with both header and footer cell depicted. The lines in bold are to characterize internal delays.

```
definePowerPort "TVDD" 1.08
defineGroundPort "TVSS" 0.0

defineDefaultPowerPort "TVDD"
defineDefaultGroundPort "TVSS"

defineGroundPowerMapping "TVSS" { "TVDD"  }

definePMCell {
  cell_name : PM1
    iv_curve {
     internal_pin : INP
      internal_delay_thresholds : (0.05 0.1 0.5 0.9 0.95)
      control_pin_input_slew : (1.0e-12 1.0e-11 1.0e-10
       + 1.0e-9 2.0e-9 4.0e-9)
init_condition : (CTL 0) (VVDD 0) (VDD 1.5)
      final_condition : (CTL 1.5) (VVDD 1.5) (VDD 1.5)
      source_drain_pair : VDD VVDD
      control_pin : CTL
    }
    ...
  }
}
```

Table 11-1 Keyword Explanation for the PG Spec File

keyword	Description
internal_pin	The internal node name in SPICE sub-circuit which controls the PM transistor. In the example shown in Figure 11-5, the internal pin name is called “INP.”

Table 11-1 Keyword Explanation for the PG Spec File (Continued)

keyword	Description
internal_delay_thresholds	The thresholds in the ratio of power supply VDD where you want to characterize the delay from control pins to internal pins. This line is optional. If you do not specify the thresholds, a default list of thresholds (0.05 0.1 0.5 0.9 0.95) will be used instead. If you want to provide the thresholds, the list has to be in an ascending order.
control_pin_input_slew	A list of slews used on control pins to characterize the internal delay. This line is also optional. If you do not specify the slews, a default list of slews (1.0e-12 1.0e-11 1.0e-10 1.0e-9 2.0e-9 4.0e-9 8.0e-9 1.6e-8) will be used. Note that the values have to be in an ascending order.

Note that it is unnecessary to specify init_condition or final_condition for the internal pin, because the internal pin is not included in the sub-circuit pin name list. PrimeRail will issue an error message if a pin specified in init_condition, or final_condition is not included in sub-circuit pin name list.

A new IV curve will be created such that the internal pin is swapped from 0 to VDD (or VDD to 0 depending how the init_condition or final_condition is specified in the pg.spec file) instead of the original control pin.

Creating a PM Cell Control File

A PM cell control file (*.config) is a user-defined ASCII file which contains different constraints for the multithreshold-CMOS cells and also the names of the output files being created. (These output files will later be used during analysis to see the effect of the turn-on sequence.) This control file is used during the full-chip rail analysis by setting the `pgPMConfigFile` switch (see “Calculating Rush Currents and Wake-Up Time” on page 11-17).

The following is an example of the power management control file:

```
# Power Management Control File
#    First word is KEY word
#    lines which starts with # are treated as comments
# Threshold 0.01
# MaxSteps 100000
# StepSize 1.0e-12
# MaxCurrentFile maxcurrent.out
# SaveStdCellVoltage true
```

```
# Dump pm_dump.out
```

Table 11-2 Keyword Explanation for the PM Control File

Keyword	Description
Threshold	The percentage of supply voltages considered as the stable controlled voltage level after power up. The default is 0.01.
Transition	The global transition time for the power management cell when power database does not have that information.
MaxSteps	The maximum number of steps to run. The default is 100,000.
StepSize	The time step for power-up analysis. The default is 1.0e-12 (second).
MaxCurrentFile	An ASCII output file containing power management cell's peak current value and time for power-up analysis.
SaveStdCellVoltage	An option to store voltage waveforms on standard cell output ports in the controlled voltage domain. The option is disabled by default.
Dump	An ASCII output file containing the information of power management cell power-up sequence. This file is similar to the sequence file that is generated during `poCalculatePower`.

Running Library Characterization

Run the `pgLibCharacterize` command to characterize the power management cell when pg.spec and control files are present. You do not need to run library characterization on macro cells.

1. Enter `pgLibCharacterize` to open the Library Characterization dialog box (see Figure 6-3 on page 6-7).
2. Select PM Cell as the Pre-Characterization Type. Select other options as necessary.

   Note:
   Do not select Intrinsic Parasitic and Current Waveform options for switch cells.

   For a detailed description about the options available in this dialog box, see “Characterizing Cell Data” on page 6-6.
3. Click OK or Apply.

   The tool generates a power management model file (*.pm) called libChar_<ref_lib_name>.pm.

The tool will link the characterization results in the library. During the linking process, the tool reports the basic properties of the cell as follows:

```
Power Management Cell Model Report:
        Cell: PM_HEADER1, Main PG: TVDD, Virtual PG: VDD,
Control Signal: CTL1 [r 1.08(V)]
        Linear Resistance: 63.232, Leakage Current: -210e-3
(nA), Max Rush Current: -5.864 (mA) at -1.08 (V)
```

During the characterization process, PrimeRail generates SPICE-related intermediate files that are usually deleted after a successful characterization run. If you want to keep the files without deleting them, define the `rfKeepSpiceFiles` switch before invoking the `pgLibCharacterize` command by typing the following in the command window:

```
define rfKeepSpiceFiles 1
```

PrimeRail saves these intermediate files in the directory called *.libChar_<ref_lib>.

To verify characterization results, run the `pgListCharResult` or `pgDumpCharacterize` command and check the parameters reported in the output files.

For more information about running library characterization, see Chapter 6, “Library Characterization for Cell-Level Dynamic Analysis,” in this user guide.

Gate-Level Power Analysis and Current Waveform Generation

When power management cells are modeled, perform either vector-free or vector-based power analysis and generate current waveforms using the `poCalculatePower` command.

During the power analysis, the tool generates the event file or the sequence file for the power management cell using PrimeTime timing information, depending on the type of the dynamic analysis you are performing.

You need to run different steps, based on the mode of the power analysis:

- Vector-Free Power Analysis

  To perform a vector-free gate-level power analysis for power management cells.

  1. Run the `poCalculatePower` command.
  2. In the Power & Current Calculation dialog box that opens, choose Vector-free as the power analysis mode. (See Figure 7-7 on page 7-17.)
  3. In the Rush Current Analysis Configuration section, choose Yes to enable the “Proceed Rush Current Analysis” option. Choose Auto as the rush current mode.

4. In the PM Cell Event File text field, enter the name you want to assign to the event file to be generated. By default, the file is named poCP_pm_event.cfg.
5. Specify other options as necessary and then click Apply. The tool generates an event file that contains the timing information.
6. Now rerun the `poCalculatePower` command with the “What-if” option. Specify the name of the generated event file in the PM Cell Event File text field.
7. Click OK.

- VCD-Based Power Analysis

  When you are performing VCD-based power analysis for power management cells, the steps are slightly different from the vector-free analysis flow. No event file is needed in the VCD flow. You simply need to run the `poCalculatePower` command to generate dynamic current waveforms for the power management cells.

For details about the `poCalculatePower` command, see “Calculating Power and Current Waveforms” on page 7-13 or the man page.

Extracting Power and Ground Network Parasitics

Run the `poExtractPGParasitics` command to extract resistance and capacitance for main power and ground and virtual power and ground net.

For more information about how to run power and network parasitics extraction, see “Extracting Power and Ground Parasitics” on page 7-5.

Calculating Rush Currents and Wake-Up Time

When power and ground net extraction is done, run the `poRailAnalysis` command to calculate rush current and wake-up time with the power management cell control file that you create (see “Creating a PM Cell Control File” on page 11-14).

Note that you must set the `pgPMConfigFile` switch before executing the `poRailAnalysis` command. To set this switch, type the following in the command window:

```
define pgPMConfigFile "pm_config.cfg"
```

where pm_config is the PM cell control file that you created during the data preparation stage.

Calculating Rush Currents

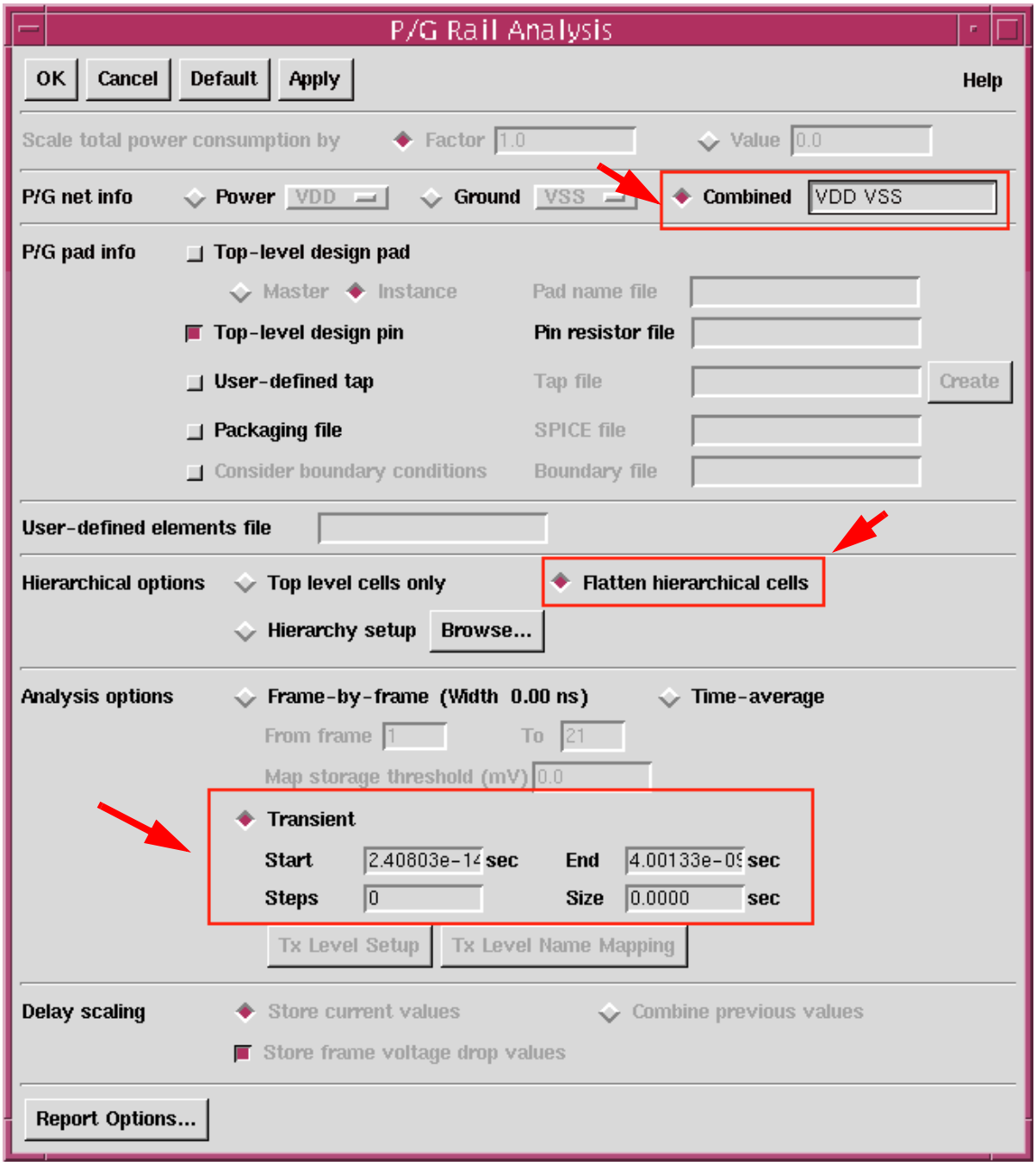
When PM cell control file is ready, run the `poRailAnalysis` command to analyze rush currents and wake-up time.

To perform rail analysis,

1. Enter `poRailAnalysis` or choose Cell-level Analysis > Rail Analysis – Rail Analysis.

   The P/G Rail Analysis dialog box opens, as shown in Figure 11-6.

Figure 11-6 Performing Combined Rail Analysis



2. In the “P/G net info” section, select Combined and enter the names of the nets to be analyzed. Selecting this option will also attach the virtual net to the main net.

3. In the “Hierarchical options” section, select “Flatten hierarchical cells” to flatten the hierarchical cells.
4. In the “Analysis options” section, select Transient and specify starting time, ending time or time step size to be used during simulation.
5. Specify other options as necessary.

   For more information about running the `poRailAnalysis` command see the Help for the command.
6. Click OK or Apply.

When the rail analysis is complete, the tool reports the following information in the log file:

- Controlled voltage domains and associated virtual power and ground nets
- Total leakage currents from power management cells and also from the controlled voltage domain
- Estimated effective capacitance being discharged
- Number of power management cells being used
- Wake-up time to reach the stable power and ground net voltage (threshold)
- Peak current and its time

With the generated information on rush currents, you can then perform full-chip rail analysis to investigate the di/dt noise caused by the power-up sequence specified in the power management cell control file.

Result Display and Report

When rail analysis is done, you can view different data points through the `pgMap` command. Or you can open the output transient current waveform file (in FSDB format) in a standard waveform viewer for viewing, like nWave. All the files related to the power management cell flow are named with the prefix PM_*.

For more information about viewing rail analysis results, see the Help for the `pgMap` command or “Displaying Voltage Drop and Electromigration Maps” on page 8-16.

12

Advanced Analysis

This chapter describes how you perform full-chip analysis by using various advanced analysis capabilities available in PrimeRail.

When analyzing designs that contain both gate-level and transistor-level blocks at the full-chip level, you have to first analyze the blocks separately and then apply the analysis results in the full-chip analysis.

The transistor-level analysis serves to provide models for the transistor-level blocks to the full-chip analysis, including both cell-level and transistor-level blocks. You can also use it as a standalone analysis capability when analyzing a memory design.

The chapter consists of the following sections:

Importing Macro Models

After you run the `poTxGenDWM` command to characterize a hard macro cell as a macro model, run the `poTxImportDWM` command to import the macro models that are generated by `poTxGenDWM` to the cell-level reference library. You must import the generated macro models before you perform a full-chip analysis.

The macro model is used to set the proper current source locations of the devices in power and ground network parasitic extraction. PrimeRail also calculates the operating modes of the macro model for all the hard macro cell instances in the design when PrimeRail generates dynamic current waveforms for cell instances. Performing dynamic rail analysis at the top level calculates the voltage variations of the full-chip power and ground network by using operation modes of the current waveforms inside hard macro cell instances.

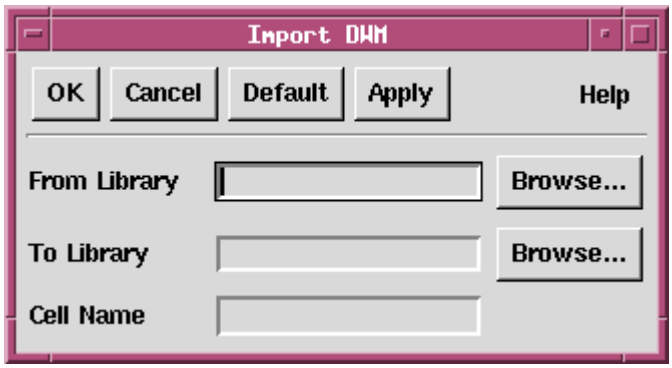
For a detailed description about how to generate hard macro models, see Chapter 10, “Using Macro Models,” in this user guide.

To import the generated macro models,

1. Enter `poTxImportDWM` or choose Macro Modeling > DWM Generation – Import DWM to Design/Ref. Lib.

   The Import DWM dialog box appears, as shown in Figure 12-1.

Figure 12-1 The Import DWM Dialog Box



2. In the From Library text field, enter the name of the transistor-level run library.
3. In the To Library text field, enter the name of the cell-level reference library where the CONN view of the macro cell is located.
4. Enter the name of the macro cell.
5. Click OK or Apply.

Note:

When the macro model is imported, you need to run the `poCalculatePower` command in order to activate the DWM models in the power database.

If this is the first time you import the macro model for this design, you need to perform power and ground net extraction with the `poExtractPGParasitics` command after the macro model is imported. Otherwise the newly imported macro model will have different current locations from those of the old one.

Including Package Parasitics

Effects of the package RLC (resistance-inductance-capacitance) model in the variation of voltage drop can be huge and can result in spikes in the power grid. Therefore, it is important for dynamic tools to analyze these effects. In PrimeRail, the effects are manifested as increased voltage drops (or bounces for the ground network) and can be reported and diagnosed using the voltage drop or electromigration maps.

PrimeRail supports the following ways of including package parasitics:

- Lumped RLC Models
- Distributed RLC Models
- General Linear Packaging Models

Lumped RLC Models

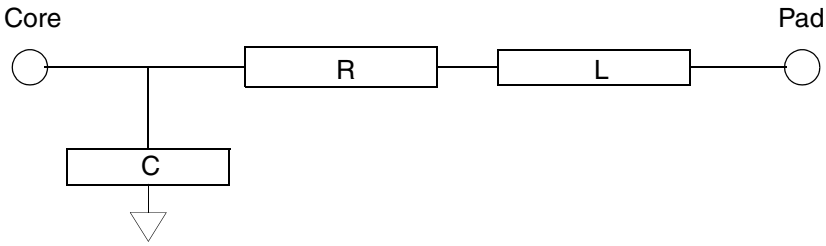
PrimeRail takes the package parasitic information in the lumped format through the top-level pad file or the user-defined tap file during full-chip rail analysis, as specified in the “P/G pad info” section of the P/G Rail Analysis dialog box (see Figure 12-15 on page 12-38).

Note:

The dynamic analysis supports only pad cells and user-defined tap files for external package RLC analysis. PrimeRail does not support top-level design pins due to the accessibility of the pin information to be used. Pin information can be read from both the CEL or FRAM view.

PrimeRail supports the following three types of RLC network. Each type requires a different setting for the `poPKGRLCType` switch.

Type A

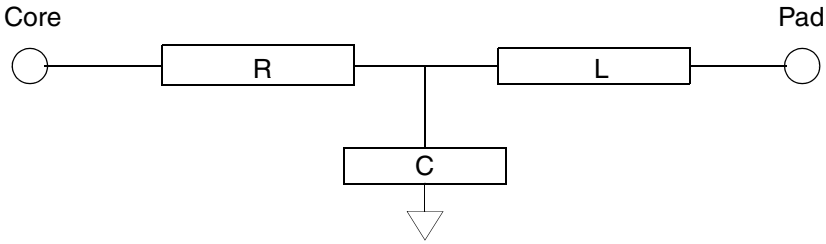


To specify Type A to be used, enter the following in the command window before invoking the `poRailAnalysis` command:

```
define poPKGRLCType 0
```

This is the default setting.

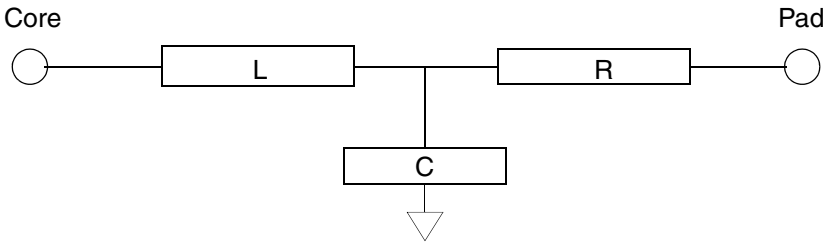
Type B



To specify Type B to be used, enter the following in the command window before invoking the `poRailAnalysis` command:

```
define poPKGRLCType 1
```

Type C



To specify Type C to be used, enter the following in the command window before invoking the `poRailAnalysis` command:

```
define poPKGRLCType 2
```

The following is an example of the package RLC information in the top-level design pad cell file:

```
T1VDDCOREA R(97e-3) L(3.4e-9) C(40e-12)
T1VDDIOA R(97e-3) L(3.4e-9) C(40e-12)
T1VDDECA R(97e-3) L(3.4e-9) C(40e-12)
```

Distributed RLC Models

If any of the built-in lumped RLC models does not appropriately reflect the parasitics of your package, you can import a SPICE SUBCKT file of RLCV (resistance-inductance-capacitance-voltage) elements that model the package parasitics correctly. The ports described in the SUBCKT file define the interface points between package elements and the IC core.

To import RLCV elements through the SUBCKT file (*.spi), use the `pgPKGFile` switch or specify the name of the SUBCKT file in the Rail Analysis dialog box, by selecting the “Package file” option and enter the name of the SUBCKT file in the SPICE file text box.

To import the SUBCKT file by defining the `pgPKGFile` switch, enter the following in the command window before invoking the `poRailAnalysis` command:

```
define pgPKGFile "fileName.spi"
```

The following is the syntax used in the SUBCKT file:

```
.SUBCKT net_name
+ tapN $ LayerN XN YN
+ tapN1 $ LayerN1 XN1 YN1

...

R node_name node_name value
L node_name node_name value
C node_name node_name value
V node_name node_name value
...
.ENDS
```

Element	Description
net_name	The name of the net. This is to select proper package boundary conditions for rail analysis, which is a net-based analysis.
tapN	The name of the tap port that is located at LayerN XN YN.

Element	Description
LayerN	The number of the layer in the Milkyway database.
XN, YN	The coordinates of the tap location in the Milkyway DB unit.
R	The element that defines the location and value of the resistor. Set the node name to 0 if it is an ideal voltage node. Value: Any constant value
L	The element that defines the location and value of the inductor. Set the node name to 0 if it is an ideal voltage node. Value: Any constant value
C	The element that defines the location and value of the capacitor. Set the node name to 0 if it is an ideal voltage node. Value: Any constant value
V	The element that defines the location and value of the voltage. Set the node name to 0 if it is an ideal voltage node. The voltage value is considered the relative value to the ideal voltage supply. Value: 0.0

In order to set proper boundary conditions, make sure that at least one node in the R, L, or V element is connected to the ideal voltage node (that is, the node name is 0). Otherwise the results will be undefined.

The power and ground coupled package model has to be broken into multiple models, one for each power or ground net. When preparing the package model for a power or ground net, you should treat all the voltage values of other power or ground nets as constant. The package RLC can be used with other boundary conditions.

Example

The following is a sample subcircuit of a design and its corresponding SPICE SUBCKT file:

A SUBCKT File Sample:

```
.SUBCKT VDD
+ tap1 $ 20 100 200
+ tap2 $ 20 200 100

R1 tap1 n1 1.0
L1 n1 n3 1.0-12
R2 tap2 n2 1.0
L2 n2 n3 1.0-12
C1 n3 0 1.0-9
L3 n3  n4 1.0-12
V1 n4 0 0.0

.ENDS
```

General Linear Packaging Models

In addition to simple package models consisting of R, L, C, and V elements, the PrimeRail matrix solver also supports package circuits with mutual inductors (K), and dependent sources, such as, voltage-controlled voltage sources (E), current-controlled current sources (F), voltage-controlled current sources (G), current-controlled voltage sources (H), and sub-circuit instances (X) for packaging in the rail analysis.

You can utilize the general linear circuit elements and subcircuit instances for packaging in net-based rail analysis. Using user-defined elements (such as user-defined taps, pads, or top design pins) together with packaging models is also supported.

Net-Based Rail Analysis

Using general linear packaging models in net-based rail analysis is the same as using distributed RLC packaging models. You first need to configure the information for packaging models in a SUBCKT file, and then you import the SUBCKT file during rail analysis (`poRailAnalysis`) by selecting the “Package file” option and entering the SUBCKT file name in the “SPICE file” text box.

The syntax of the SUBCKT file is shown as follows:

```
.SUBCKT net_name
+ tapN $ LayerN XN YN
      + tapN1 $ LayerN1 XN1 YN1
        ...

      R node_name node_name value
      L node_name node_name value
      C node_name node_name value
      V node_name node_name value
      K inductor_name inductor_name value
      E node_name node_name node_name node_name value
      F node_name node_name volt_src_name value
      G node_name node_name node_name node_name value
      H node_name node_name volt_src_name value
      X node_name node_name ... node_name subckt_name
      ...
      ...
      .ENDS
```

Element	Description
net_name	The name of the net. This is to select a proper package boundary condition for the net-based rail analysis. tapN: The name of the tap port that is located at LayerN XN YN. LayerN: The number of the layer in the Milkyway database. XN YN: The coordinates of the tap location in the Milkyway database unit.
R	Defines the location and value of the resistor. Set the node name to 0 if it is an ideal voltage node. Valid value: Any constant value.
L	Defines the location and value of the inductor. Set the node name to 0 if it is an ideal voltage node. Valid value: Any constant value.

Element	Description
C	Defines the location and value of the capacitor. Set the node name to 0 if it is an ideal voltage node. Valid value: Any constant value.
V	Defines the location and value of the voltage source. Set the node name to 0 if it is an ideal voltage node. The voltage value is considered the relative value to the ideal voltage supply. Valid value: 0.0
K	Defines the location and value of the mutual inductor. inductor_name: The name of the coupled inductor. Valid value: Any value greater than or equal to -1 and less than or equal to 1.
E	Defines the location and value of the voltage-controlled voltage source. Set the node name to 0 if it is an ideal voltage node. Valid value: Any constant value.
F	Defines the location and value of the current-controlled current source. Set the node name to 0 if it is an ideal voltage node. volt_src_name: The name of the voltage source through which the controlling current flows. Valid value: Any constant value.
G	Defines the location and value of the voltage-controlled current source. Set the node name to 0 if it is an ideal voltage node. Valid value: Any constant value.
H	Defines the location and value of the current-controlled voltage source. Set the node name to 0 if it is an ideal voltage node. volt_src_name: The name of the voltage source through which the controlling current flows. Valid value: Any constant value.
X	Defines the location of a subcircuit instance. Set the node name to 0 if it is an ideal voltage node. subckt_name: the name of the .SUBCKT definition.

When the SPICE file is imported, the tool prints the following message to the log file:

```
Circuit summary:
      # of subckt: 1
```

```
# of subckt instances: 0
# of device models: 6
# of device instances: 12
# of signal nets: 14
```

The message summarizes the number of devices and nets in the package. Note that the tool supports X elements, which means the package circuit can be hierarchical. The ports of the .SUBCKT section with power and ground net name as its reference name define the interface points between package elements and IC core.

Using the Package File With User-Defined Elements

The package file can be used together with user-defined elements, such as, user-defined taps, pads, or top-design pin to define the boundary conditions of the IC core.

For example, if you want to use user-defined taps along with packaging models, define two taps in the tap file as follows:

```
VDD  16  21 104
VDD  18  14 330
```

In the package file (*.spi), define a resistor to connect these two taps:

```
.SUBCKT VDD
+ tap1 $ 16 21 104
+ tap2 $ 18 14 330

R1 tap1 tap2 1.0
.ENDS
```

Note that if you use a package together with user-defined elements, no voltage source with nonzero DC value can exist in the package. In other words, when nonideal voltage supplies exist in the package, user-defined taps are not allowed to be defined in the core.

Note:

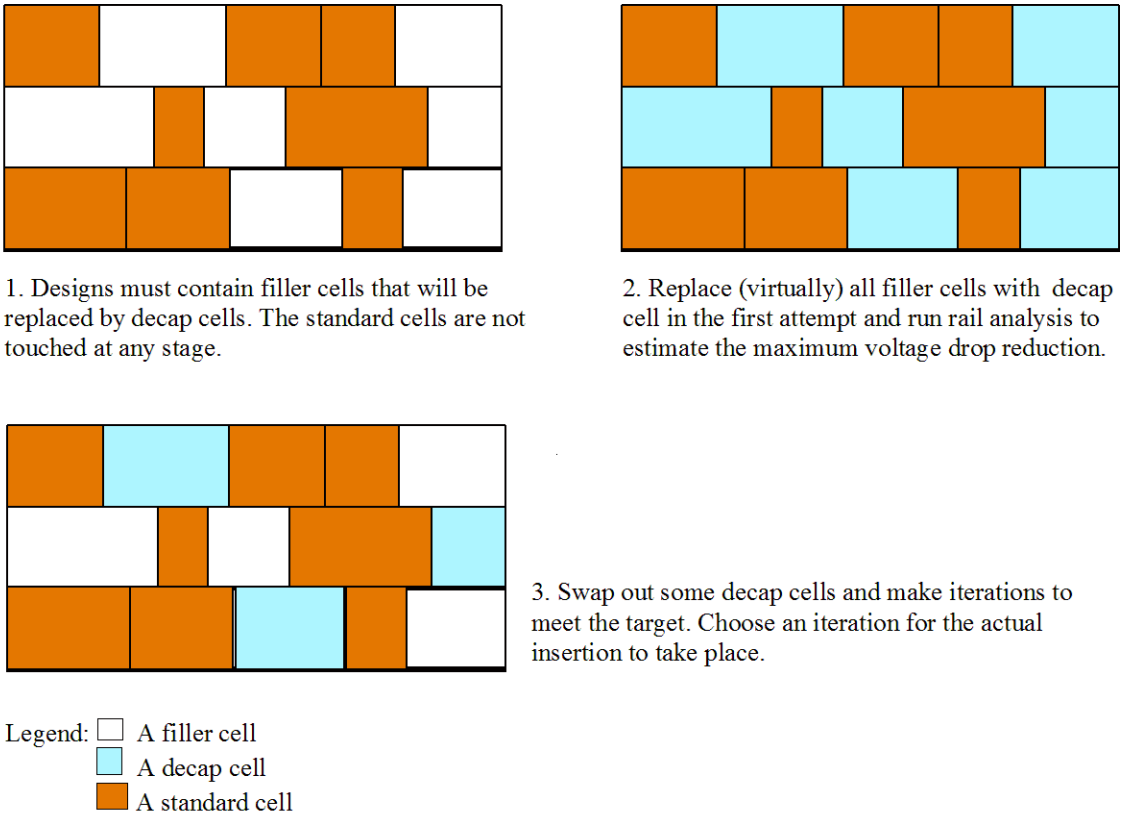
The number of interface points between package and core must be less than 1,000. Otherwise the tool terminates with an error.

Decoupling Capacitor Insertion

One of the most widely used techniques to solve power supply noise is to add decoupling capacitance (decap) cells in the affected areas. It is very helpful if the analysis tool can make some suggestions where decoupling capacitors can be used more efficiently to minimize power supply noise. It will be even better if the tool can do something further than just the analysis, such as placing the decap cells to achieve a certain user-predefined percentage of voltage drop reduction.

As shown in Figure 12-2, PrimeRail is capable of maximizing voltage drop reduction with minimum cost in terms of available decap resources. The tool first looks at the available filler cells in the design and then virtually replaces them with decap cells. The tool considers the cell's capacitance value versus leakage as a cost function to meet the user-provided target reduction. It can take several iterations during the analysis to find the optimal solution. When the analysis is complete, PrimeRail provides suggestions for decoupling capacitor insertion, and you can choose which suggestion is the best fit and perform the actual decoupling capacitor insertion.

Figure 12-2 Examples for Inserting Decoupling Capacitors



When the insertion is complete, rerun rail analysis to verify the insertion results. You can also examine the inserted decap cells by displaying the decap density map.

Note that in PrimeRail, inserting a decap cell is not effective in the following cases:

- The power and ground network is not properly routed; that is, some instances are “isolated” from the rest of the network. The effective resistance from the ideal voltage source to such instances is so high that the dynamic voltage drop is dominated by large resistance. In this case solving the voltage drop problem requires the unrealistically large capacitance in order to reduce the voltage drop.

- The power and ground network is close to ideal; that is, there is a very small effective parasitic resistance. It usually happens on a very conservative power and ground net design or flip-chip packaging. The dynamic voltage drop is dominated by a large peak inductance. To solve this problem, reduce the peak current instead of adding more capacitors.
- The inductive noise is a dominant factor in the dynamic voltage drop flow. In this case the number of affordable decoupling capacitors inserted is much less than what is needed to reduce the noise effectively.

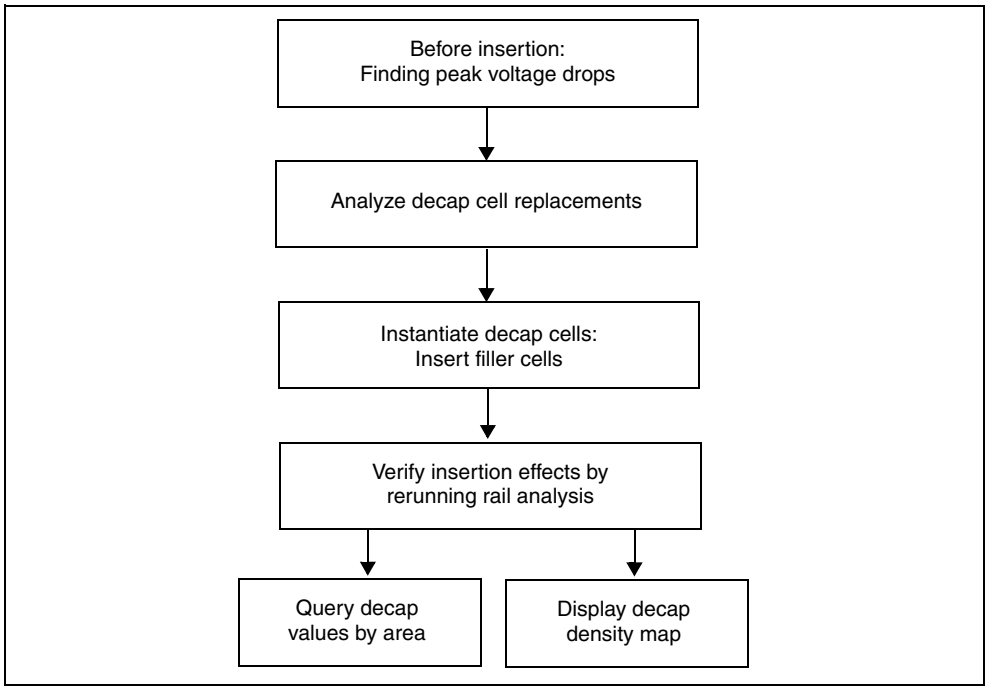
It is also important to know the following aspects of this feature so that their expectation is set correctly:

- The decoupling capacitor insertion flow in PrimeRail works at the block level. This means that the tool will not dive into a soft macro to perform decap insertion.
- Because the Prime-Rail tool is used for sign-off, the decap insertion flow is expected to be used as part of the ECO flow.
- PrimeRail does not generate any ECO file but can insert decoupling capacitors within its environment.

Decap Insertion Flow

Figure 12-3 illustrates the flow of inserting decoupling capacitors in PrimeRail:

Figure 12-3 Decoupling Capacitor Insertion Flow



The following are the known assumptions in the PrimeRail decap insertion flow.

- During the decap insertion flow, it is recommended that you do not include any packaging in the package file or the tap file. Because package parasitics can introduce oscillation in voltage drop waveforms, inserting decoupling capacitance cells might not be efficient to reduce the magnitude of the oscillation.
- The Decap Insertion dialog box accepts a list of cell master names, separated by a comma (,) or a white space. Wildcard usage is supported. Decap insertion can be performed only on a net basis, which means that if you want to insert decap cells for multiple power and ground nets, you have to run the decap insertion flow for each power and ground network separately.
- After running instantiation, you must save the Milkyway database before exiting from PrimeRail.

Design Requirement

In addition to the requirement of a normal dynamic analysis, you need to prepare the following information specifically for performing decoupling capacitor insertion in PrimeRail:

- A FRAM view for filler and decap cells.

- The SPICE .subckt file and model information for decap cells for the library characterization purpose.
- A design that has filler cells added by IC Compiler. If it is a multiple-VDD design, you may consider decap insertion by area.

If you do not have the .subckt information for the decap cells, you must provide the capacitance value. This is the minimum requirement because a dummy .subckt file can be created using that value. If you want to define the DC leakage in addition to the decap value, you can add one resistor connected from vdd to vss, where R = Vdd / leakage_current.

The following is an example of such a dummy .subckt file:

```
.subckt decap1 vdd vss
C_dcap1 vdd vss 10f
R_leak vdd vss 1.38e9
.ends
```

Before Insertion—Finding Peak Voltage Drops

Before you insert a decap cell in a design, you need to complete the following steps:

1. Add fillers to the design by using any place and route tools, such as IC Compiler.

   If you are doing placement and routing in IC Compiler, the command used to insert filler cells is `insert_stdcell_filler`. For more information about this command, see the command man pages or the IC Compiler documentation.

2. Run `pgLibCharacterize` or `pgPreCharacterize` to characterize intrinsic capacitance and leakage current for filler and decap cells in PrimeRail.

   Capacitance values for decap cells are needed and can be characterized in PrimeRail. Besides intrinsic capacitances, the leakage current (which is optional) can also be characterized for leakage consideration as a cost function during the decap analysis. Run `pgLibCharacterize` with the Gate Intrinsic Parasitic and Filler Cell Leakage options. Note that for these filler and decap cells, the .subckt ports usually contain only power and ground ports; it does not make sense to select any other options, like Current Waveform during library characterization in the Library Characterization dialog box.

   Also, make sure you specify the same PVT information for decap characterization as you will be using it for analysis. Characterizing decap cells with a different temperature than the one used during analysis will result in incorrect analysis results.

   When characterization is complete, write out the intrinsic parasitic and filler cell leakage information to an output file and verify if it has the correct PVT information and the characterized data.

Note:
The SPICE netlist for filler and decap cells is required for characterizing leakage currents.

If you run the `pgLibCharacterize` command, the characterization result is automatically saved to the file called *filler.leakage*. When characterization is done, the tool automatically links the characterization to the library.

If you run the `pgPreCharacterize` command, the characterization result file is saved to a user-specified file, which you need to manually link to the library by using the `pgLinkCharacterize` command.

For more information about library characterization, see Chapter 6, "Library Characterization for Cell-Level Dynamic Analysis" in this user guide.

3. When filler cells are added to the design, regenerate the rail database and extract the parasitic information. This includes the following steps:

   a. Run `poPurgeRail` to remove the existing rail database.

   b. Run `poCalculatePower` to regenerate the power database and current waveforms at power ground ports of the standard cells.

      For more information, see "Calculating Power and Current Waveforms" on page 7-13.

   c. Rerun power and ground net extraction with filler cell port instances with the `poExtractPGParasitics` command.

      Performing power and ground net extraction in the decoupling capacitor insertion flow is the same as in any normal cell-level dynamic analysis flow. The only difference is you need to define the `poIncludeFiller` switch before running `poExtractPGParasitics`. By default, filler cells are ignored for any kind of analysis. You need to define the `poIncludeFiller` switch to ensure the power and ground ports of the filler cells are extracted as part of the parasitic network.

      ```
      define poIncludeFiller 1
      ```

      For more information about running extraction, see "Extracting Power and Ground Parasitics" on page 7-5 in this user guide.

   d. When all the previous steps discussed in this section are finished, run `poRailAnalysis` to perform rail analysis. Or, you can perform rail analysis with the `pgDecapInsertion` command when inserting decoupling capacitors.

Analyzing Decap Cell Replacements

PrimeRail is capable of analyzing which filler cells are to be replaced by decap cells before the replacement is actually done. The tool iterates through different decap replacements until the user-defined target is met and runs rail analysis to verify the result. During the

iteration process, the tool prints multiple DECAP messages that give detailed information about the virtual replacements. Based on this information you can choose which iteration to instantiate.

Considering Leakage Current

Before invoking the `pgDecapInsertion` command, use the LEAKAGE_FACTOR switch to specify a cost function which determines the balance between decoupling capacitances versus leakage currents, like

```
define LEAKAGE_FACTOR 0.5
```

The following formula expresses the cost function calculation in PrimeRail:

```
(1.0-leakage_factor) x total_decap_cap + leakage_factor x total_decap_leakage
```

Set LEAKAGE_FACTOR within [0.0, 1.0]. Otherwise the tool resets the value to 0.0. The objective of decap insertion is to minimize the voltage drop value using the minimum cost.

By default, LEAKAGE_FACTOR is “0.0” and this means the cost function is to consider only the capacitance values associated with the decap cells. The larger the value of LEAKAGE_FACTOR is, the greater the effect of leakage current is in cost function calculation. When LEAKAGE_FACTOR is set to 1.0, the cost function is the total leakage current of decap cells.

For a value between 0.0 and 1.0, the cost function is a weighted sum of capacitance and leakage current. You can adjust the value between total capacitance and total leakage as necessary.

Note:

When using the LEAKAGE_FACTOR switch, be sure not to define PWL current waveforms in the user-defined elements file for decap cells during decap insertion flow. The tool is unable to distinguish between PWL tables inserted by the user-defined elements file and PWL leakages inserted during decap insertion flow. As a result, the overlapping PWL tables for the same decap cell master (or instance) will be ignored.

Tips

The following tips are useful in the decap insertion flow:

- Use `axgListSummary` to write out the design’s master cell information.
- Make a note of peak voltage information for some top cell instances (like 4 to 5). If the peak happens in different areas, the cell instances need to be chosen from both locations.
- Run `pgMap` to display voltage drop map and also save the voltage waveform information.
- Record the reported core and effective capacitance values during rail analysis.

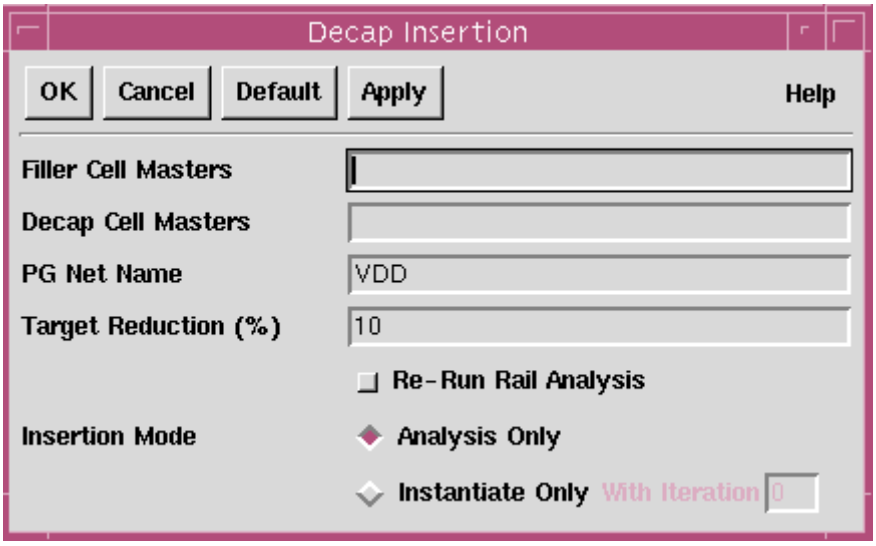
Analyzing Filler Cell Replacements

To analyze filler cell replacements before actual inserting the decap cells,

1. Enter `pgDecapInsertion` or choose Cell-level Analysis > Optimization – Decap Insertion.

   The Decap Insertion dialog box opens, as shown in Figure 12-4.

Figure 12-4 Decap Insertion Dialog Box



2. Enter the names of the filler cell masters to be replaced with decap cell masters. Use a comma (,) or a space to separate master names. Wildcard usage is also supported.

   Note that the filler cell masters must be placed or routed in IC Compiler. For more information, see step 1 in "Before Insertion—Finding Peak Voltage Drops" on page 12-14.

3. Enter the names of the decap cell masters used to replace filler cells. Use a comma (,) or a space to separate master names. Wildcard usage is also supported.

4. Enter the name of the power or ground net to be analyzed. You can analyze only one net at a time.

   The name must be consistent with the power or ground net name specified in the P/G Rail Analysis dialog box.

5. Specify the percentage of reduction you want the decap insertion flow to meet.

6. Select Re-Run Rail Analysis if you want to rerun the rail analysis to generate a reference voltage drop value. By default, this option is off.

7. Select the insertion mode.

- Analysis only – When selected, PrimeRail uses iterations to determine which filler cells can be replaced with decap cells to satisfy the preset target, and it performs rail analysis to verify the result. The decap cell to be replaced will have the same or a smaller footprint than the filler cell. PrimeRail automatically determines which decap cells can be used for replacing certain filler cells. However, the cell instantiation does not actually happen in this mode. That is, the filler cells are not actually replaced with decap cells; it is more like a virtual decap cell replacement.

  During analysis PrimeRail iterates among different decap replacements until the target is met. The tool prints the messages to the log file to reflect the iterative process, with the prefix DECAP.

  In this mode PrimeRail does not automatically replace decap cells using the last iteration. Instead it gives you the option of deciding which iteration to instantiate.

  Iteration Example:

```
DECAP: After inserting all filler cells to design, the voltage drop
DECAP: reduction is 13.796% (328.634 mV -> 283.295 mV),
which is less than user target 30%
DECAP: Decap insertion is not effective for reducing voltage drop
DECAP: decap insertion analysis succeeded
```

  Explanation:

  As discussed earlier, at first attempt the tool will replace all the filler cells to see the maximum voltage drop reduction. In this example, the target reduction is too high or the decap resource is not enough; even all the filler cells are replaced. The method of simply inserting decap cells cannot solve the voltage drop problem. PrimeRail will not run further to make the iterations to meet this target.

  Iteration Example:

```
DECAP: After insert all filler cell to design, the voltage drop
DECAP: reduction is 40%, which is greater than user target 10%
DECAP: Will reduce the amount of cap to meet target.
```

  Explanation:

  The decap resource is enough to solve the voltage drop problem and the preset target reduction can be met. PrimeRail will use the decap resource as less as possible to meet the target.

- Instantiate Only: When selected, PrimeRail instantiates the decap cell replacement. For more information, see “Instantiating Decoupling Capacitors” on page 12-19.

  Note:
  You need to run the “Analysis Only” mode first so that PrimeRail can use the iterations to determine which filler cells are to be replaced in order to meet the target for what-if rail analysis.

8. Click OK or Apply.

Instantiating Decoupling Capacitors

When virtual decap cell replacement is done in the Analysis Only insertion mode (`pgDecapInsertion`), you can choose which iteration to instantiate based on the DECAP message printed during decap cell replacement.

After instantiation is done, be sure to save the cell before closing PrimeRail. Otherwise PrimeRail will not save the instantiation result (or decap replacement) in the Milkyway database.

To instantiate decoupling capacitors,

1. Enter `pgDecapInsertion` to open the Decap Insertion dialog box (see Figure 12-4 on page 12-17).
2. Select Instantiate Only as the insertion mode and specify the number of iterations to be used. This option lets you decide which decap insertion pattern is instantiated (according to the iteration numbers shown as messages in the log file during the Analysis Only mode).
3. Specify other options as necessary.
4. Click OK or Apply.

After instantiation is done, a message is printed to the log file:

```
DECAP: replace cell xofiller!FILL1!1 HNId 2330 from master FILL1 to
master FILLDCAP1
DECAP: replace cell xofiller!FILL1!2 HNId 2331 from master FILL1 to
master FILLDCAP1
DECAP: replace cell xofiller!FILL1!3 HNId 2332 from master FILL1 to
master FILLDCAP1
-----more messages in PR_LOG_DETAIL/PrimeRail.log.03_14_14_13.decap-----
DECAP: total 817 filler cell instances are replaced during decap
instantiation
DECAP: decap insertion instantiation succeeded
```

A detailed message is saved to the PR_LOG_DETAIL directory. In the message, the tool prints out the filler cell instance name, the filler cell master name, and the replacement decap cell master name.

Verifying Insertion Results

You can quickly check the status (master cells count) of decap and filler cells with the `axgListPRSummary` command. To display the decap map and the voltage drop map, run `pgMap` to see the improvement on voltage drop effects.

If you want to verify the rail analysis result, run `aprPGConnect` on the newly added cells and `poPurgeRail` to purge the RAIL view. Then rerun dynamic analysis, including `poExtractPGParasitics`, `poCalculatePower`, and `poRailAnalysis`.

When decap insertion is done, run DRC check in your place and route tool to make sure no violation exists in the design. PrimeRail does not perform any DRC check.

Displaying Decap Density Maps

You can display a decap density map to check for the decoupling capacitors used in the design.

When displaying a decoupling capacitance map, the tool draws the following types of decoupling capacitors on the map:

- Node capacitors on the power and ground nets
- Intrinsic capacitors for the cell instances
- Capacitors inserted through the user-defined elements file

To display a decoupling capacitance density map,

1. Enter `pgMap` in the command window or choose Cell-level Analysis > Display – Display Cell-level Maps.
2. In the Display Map dialog box that appears (see Figure 12-6 on page 12-25), choose which analysis results you want to display from the Results list.

   When done, click the Load button to load the data to memory. You must load the data before displaying a map. Click Remove to remove the data.
3. In the Map Options section, choose Decap from the pull-down menu to display a decoupling capacitance density map.
   - Map Configuration – Specify additional options for map display in the Map Configuration dialog box that appears.

     Color Configuration – Define the color of each step to be shown on the map in the Display Color Setup dialog box that appears.
   - Density Mode – Enter the dimension of the window to display the map. You can display the decoupling capacitance map only on a window basis.
4. Display Options – Select the options related to the decoupling capacitance map.
   - Show Text – Show decoupling capacitances on the map.
   - Pads – Highlight the pads used as ideal voltage sources in rail analysis.

- Legend – Display the upper and lower bound values of the metal and via layers in the map. Click Select Layer to choose the layer whose upper and lower bound values are to be shown in the map.
- Violations In Red – Indicate violations in red on the map.
- Peak Position In Blink – Show a blinking rectangle where the maximum value is located.
- With Solid Fill Pattern as Default – Draw the map with the shapes filled with a solid color.
- Floating Metals In Grey – Show floating metals on the map.

5. Click OK or Apply in the Display Map dialog box. The tool displays the results in the cell editing window.

Querying Decap Values by Area

When the decoupling capacitance density map is displayed, you can check for the decoupling capacitance values by area for the design. To do this, click Query at the top of the Display Map dialog box and then click a spot in the cell editing window. PrimeRail writes the values to the log and the command window.

The following is an example of querying decoupling capacitance values:

```
[ Decap map] - area  query

Map = Decap
--------------------------------
Node 3508: layer = metal1, bBox = (120.400 249.740) (120.760
250.800)
    Decap Value = 0.000792
Node 3509: layer = metal1, bBox = (120.400 247.330) (120.760
249.740)
    Decap Value = 0.000426
Node 6559: layer = metal1, bBox = (121.400 249.740) (121.800
250.800)
    Decap Value = 0.0504
------------------------------------------------
Total cap under selected area = 0.0516 pF
```

What-If Analysis

In PrimeRail, you can choose to analyze time-average voltage and current violations of the design by adding non-ideal voltage sources or external resistors on pins and pads or through a user-defined tap file.

In the dynamic analysis flow, you can use the what-if analysis capability in PrimeRail to optimize voltage drop and electromigration effects by inserting virtual capacitors or resistors to the power and ground network through the voltage drop or resistivity map. This lets you easily diagnose design weaknesses of the power and ground network without repeating the rail analysis setup process. You can even switch between the results before and after virtual capacitors or resistors are applied.

This section includes the following topics:

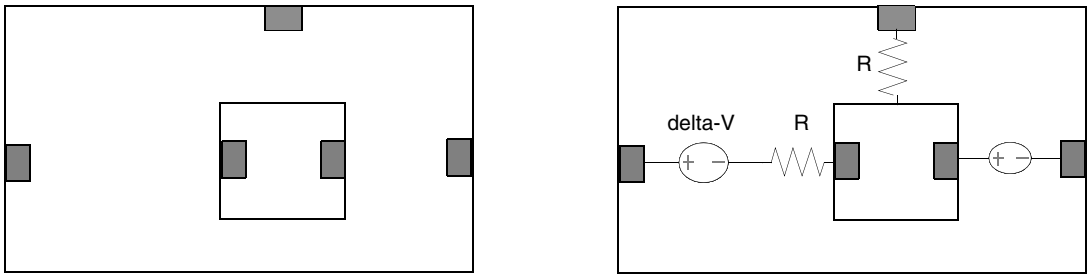
Why Use What-If Analysis

If the power and ground nets of your design do not yet connect to pads or pins or you are conducting a block-level analysis, PrimeRail allows you to assign non-ideal voltages (voltages that are lower or higher than the actual supply voltages) and external resistors to pins or pads in your design, predicting what the actual voltages will be for calculating the current consumption of the design. The non-ideal voltage sources can be controlled by a resistance value or a voltage drop or rise or by both.

If your design does not yet have pins defined, or the locations of the pad cells are not finalized, you can insert non-ideal voltage sources and external resistors through a user-defined tap file where the voltage is specified by coordinates and layer numbers.

Figure 12-5 is a sample block at the top level before and after inserting non-ideal voltages and external resistors for the what-if scenario.

Figure 12-5 An Example of Inserting Non-Ideal Voltages and External Resistors to Pins or Pads



Note:

For more information about specifying ideal voltage sources for rail analysis, see “Ideal Voltage Source Locations for Rail Analysis” on page 7-23.

Adding Non-Ideal Voltages or External Resistors on Pins or Pads

To conduct the what-if analysis, add non-ideal voltages or external resistors to pins or pads in a design. Adding voltages or external resistors through a user-defined tap file is also supported.

To add a non-ideal voltage drop (or rise) or an external resistor to pads or pins in your design, create an ASCII file and specify the information with the following format:

```
<name> [<resValue>] [<delta-V>]
```

where each variable requires a specific format, as described below:

```
<name> : = pinName | padMasterName | padInstanceName
<resValue> := <valueSet> | R(<valueSet>)
<delta-V>  := V(<valueSet>)
<valueSet> = nomValue | (nomValue minValue maxValue)
```

The `pinName`, `padMasterName`, and `padInstanceName` variables represent the simple names of a pin, a pad master, and a pad instance, respectively. They relate to the “P/G pad info” option you select in the P/G Rail Analysis dialog box either during time-average or dynamic analysis flow. For example, set the `pinName` variable when the “Top-level design pin” option is selected. The `padMasterName` and `padInstanceName` variables are for the “Top-level design pad” option.

The `delta-V` variable represents an external voltage drop or rise.

When non-ideal voltages or external resistors are specified in the ASCII file, run the `poRailAnalysis` command to perform what-if analysis on the design, by specifying the name of the ASCII file that contains non-ideal voltage sources or external resistors in the appropriate P/G Pad Info option.

Example:

```
"VDD" R(1.0 0.99 1.01)
"VDD" R(1.0) V(-0.1)
pvdi 1.0 V(-0.1)
```

Adding Voltages or External Resistors Through User-Defined Tap File

The user-defined tap file is used to specify where the voltages are supplied by coordinates and layer numbers in the ASCII format with the following format:

```
<netName> <layerNumber> <xCoord> <yCoord> [<resValue>] [<delta-V>]
```

where each variable requires a specific format, as described below:

```
<netName>     := the name of power or ground net
<layerNumber> := layer number
<xCoord>      := X-Coordinate in user unit
<yCoord>      := Y-Coordinate in user unit
<resValue>    := <valueSet> | R(<valueSet>)
<delta-V>     := V(<valueSet>)
<valueSet> = nomValue | (nomValue minValue maxValue)
```

Alternatively, you can manually create a tap file or have PrimeRail automatically generate one by using the `poGenUserDefineTap` command. The command enables you to insert taps to a power or ground net by directly selecting a point in the design layout. The tool automatically writes the location of the taps you insert to a tap file.

For a detailed description of running the `poGenUserDefineTap` command, see "User-Defined Taps" on page 7-25.

When the tap file is available, run `poRailAnalysis` to perform what-if analysis on the design, using the information specified in the tap file.

Example:

```
VDD 16  36 68.5 R(1.0) V(-0.1)
VSS 18  10.4 7.4 R(1.0)
VDD 16  36 68.6
```

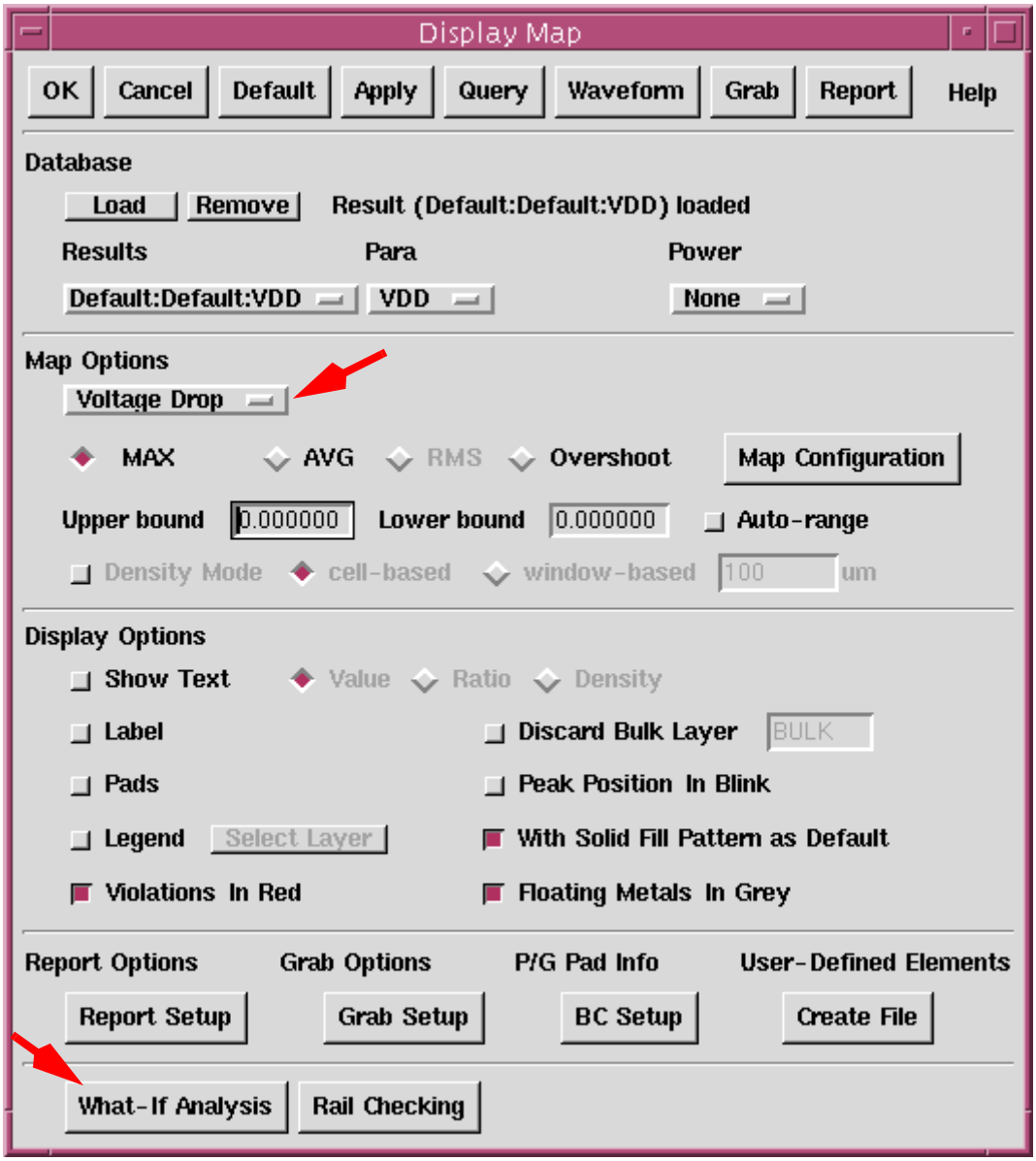
What-If Capacitance Analysis

When you complete the dynamic rail analysis at the cell or transistor level, you can insert a virtual capacitor and check whether the change you make improves the analysis results.

To perform what-if capacitance analysis,

1. Open the library and the cell to be analyzed.
2. Run `pgMap` to open the Display Map dialog box.

Figure 12-6 Display Map Dialog Box

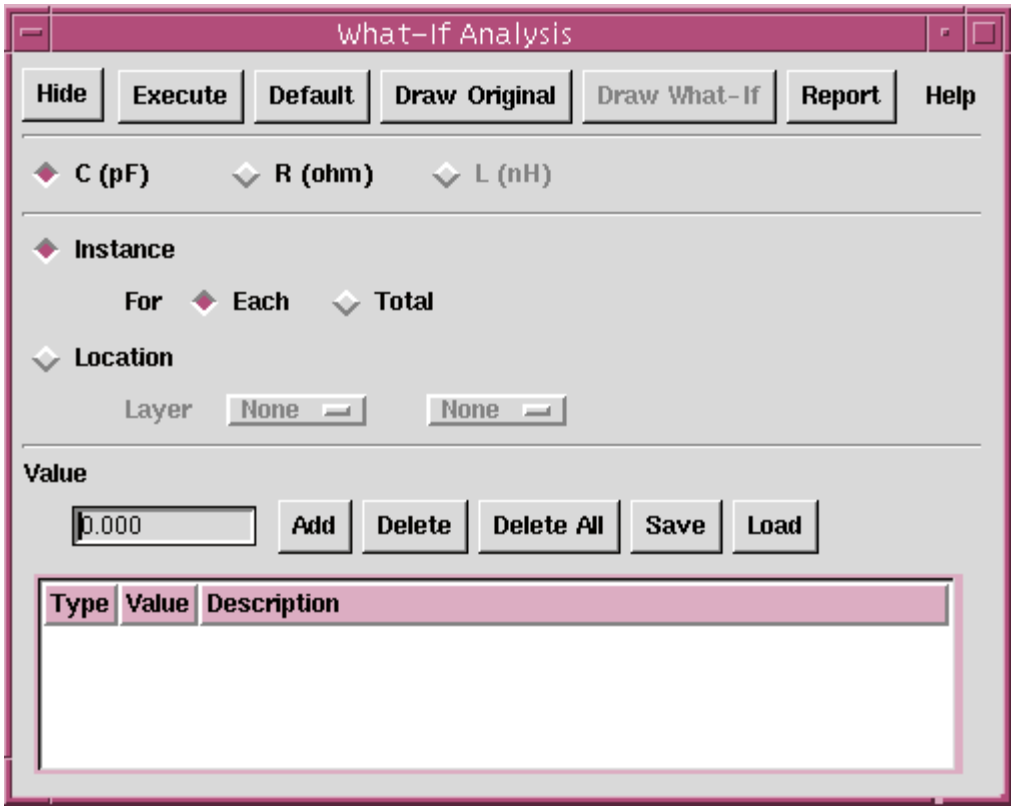


From the Results menu, choose the database result to be loaded and then click Load.

In the Map Options section, select VD Map or Resistivity Map. Specify other options as necessary. Click Apply to display the map in the cell editing window.

3. Next, click the What-If Analysis button at the bottom of the Display Map dialog box to open the What-If Analysis dialog box.

Figure 12-7 What-If Analysis Dialog Box



4. Select C (pF) to perform what-if capacitance analysis.
5. Specify whether to add the capacitors by instance or location.
   - Instance – Assign the capacitance value defined in step 6 by instance.

     Each – Assign the defined capacitance value to each of the instances in the area you select in step 7.

     Total – Assign the defined capacitance value equally to all the instances in the area you select in step 7.

     For example, if you select Each and the value assigned in the Value text box is 1,000, the tool adds a 1,000 picofarad (pF) capacitor to each of the instances. If you select Total and the value assigned is 1,000, and the amount of instances in your selected area is 10, each instance is added with a 100 pF capacitor.
   - Location – Assign the capacitance value defined in step 6 by location. Select the layers where an instance with the capacitance value defined in step 6 is to be connected.

     Note that if the voltage drop map is displayed with transistor-level dynamic analysis results, only metal layers can be selected.

6. Enter the value to be assigned. Click Add to add the element you just create to the list box in the lower section of the What-If Analysis dialog box.

   To remove a virtual capacitor, click to select a capacitor and then click Delete. To remove all the virtual resistors in the list, click Delete All.

7. Now select an area in the cell editing window. If no capacitor is present in the selected area, the tool issues an error message.

8. Click Execute at the top of the dialog box. The tool automatically performs rail analysis in the background with the virtual capacitors you created and redraws the voltage drop map in the cell editing window.

   Note:
   The Execute button is dimmed if the map being displayed is resistivity map.

9. Click Draw Original or Draw What-If to switch between the results before and after the virtual capacitors are applied.

10. The tool clears the element information listed in the list box when you close the What-If Analysis dialog box. If you want to keep the element information for next run, click Save. Note that the output element file is in binary format and cannot be modified.

    Click Load to restore the element list.

11. If you want to save the element information to an ASCII file, click the Report button at the top of the What-If Analysis dialog box. The tool saves the output ASCII file, named WhatIfListResult.txt, in the working directory.

What-If Resistance Analysis

PrimeRail supports adding a branch with a pre-specified resistance value and layer information between two existing nodes to the power network. To see the what-if effect on the rail analysis result, you can easily add a virtual resistor through the resistivity map. Note that no physical wires or vias are added in the layout during what-if analysis flow.

To perform what-if resistance analysis on the resistivity map,

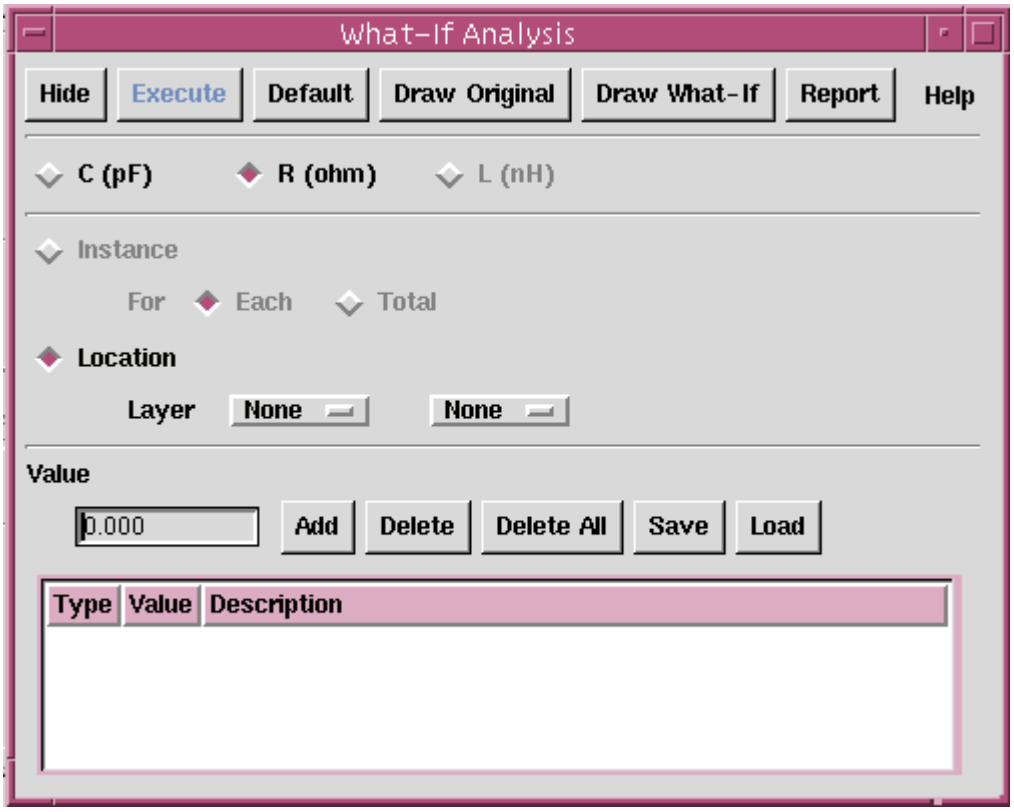
1. Open the library and the cell to be analyzed.

2. Run `pgMap` to open the Display Map dialog box (see Figure 12-6 on page 12-25).

   Load the results (or Para), and select Resistivity from the Map Options menu. The tool displays the resistivity map in the cell editing window.

3. Click the What-If Analysis button at the bottom of the Display Map dialog box to open the What-If Analysis dialog box.

Figure 12-8 What-If Analysis Dialog Box



4. Select “R (ohm)” to perform what-if resistance analysis.
5. Select the layers where the resistance value defined in step 6 is to be assigned.
6. Enter the value to be assigned. Then click Add to add the virtual resistor you just created to the list box in the lower section of the What-If Analysis dialog box.

   To remove a virtual resistor, click to select a resistor and then click Delete. To remove all the virtual resistors in the list, click Delete All.
7. Now in the cell editing window, select two points where the virtual resistor is to be added in between. When done, a blinking red line shows up in the cell editing window. If no resistor is present in the selected area, the tool issues an error message.

   Note that PrimeRail allows the value of a virtual resistor to be zero. If this is the case, PrimeRail adds a smaller resistor, 1.0e-6 ohm, by default.
8. Now click Draw What-If at the top of the dialog box to redraw the resistivity map with the inserted virtual resistors.

   Click Draw Original or Draw What-If to switch between the results before and after the elements are applied.

Multiple PVT Analysis

PrimeRail is capable of handling multiple process, voltage, and temperature (PVT) corners during library characterization, dynamic power analysis, and dynamic macro model generation. This allows you to perform power and rail analysis on several different PVT corners for a single design. During the library characterization, PrimeRail is capable of choosing the correct cell information when multiple .db files with different nominal voltages exist in the Logical Model (LM) view. Later when performing the dynamic power analysis, the tool then chooses the correct library characterization information based on the operating voltage when multiple PVT sets exist.

In PrimeRail, the operating voltage can be defined in the .db, pg.spec, and .tdf files. PrimeRail is able to check whether these values are matched with each other. If they are not, the tool prints error messages to the log file. To keep the number of warnings manageable, a maximum of five warnings of each type will be printed in the log file, with additional messages going to the PrimeRail.log.detail file under the PR_LOG_DETAIL directory.

The following is the recommended procedure for performing the multiple PVT analysis on the design:

1. During data preparation, link at least one .db file to the LM view of each reference library whose nom_voltage matches the operating voltage of the design.

   Run `poLoadRailSetup` to read in the necessary rail setup information from IC Compiler.

   For details, see Chapter 5, “Design Setup and Checking,” in this user guide.

2. Ensure that the pg.spec file to be used in library characterization defines the voltage of the power ports at a value that matches the operating voltage of the design.

   Using separate pg.spec files for each PVT corner, perform library characterization and link the result for each PVT corner at a time. Then run `poCallPTPX` to generate PrimeTime PX reports based on the desired PVT corner. The .db files should correspond to the nom_voltage that matches the operating voltage for the design.

   For more information about running library characterization, see Chapter 6, “Library Characterization for Cell-Level Dynamic Analysis,” in this user guide.

3. Run `poCalculatePower` and check the resulting warning and log messages. Then run `poExtractPGParasitics` and `poRailAnalysis` to analyze the dynamic reliability effects of the design.

   If you are running `poCalculatePower` and `poRailAnalysis` with DWMs that contain multiple PVT corners, the tool automatically picks the nearest corner of the DWM according to the PVT information stored in the PrimeTime PX report and power database, respectively.

The following syntax is a sample message about PVT corners when running the `poCalculatePower` and `poRailAnalysis` commands with multiple PVT corners:

```
PVT corner 1.00:1.50:27.00 is used for DWM sram1056x12
DWM: Open model for cell sram1056x12 in library memlib2.
```

Note that library characterization will fail if the voltages defined in the pg.spec file do not match any voltage in the .db file. If PrimeTime PX is run on a .db file whose nom_voltage does not match that in the synopsys_pr_setup.e file, the tool will ignore the switching power and short-circuit power of each cell instance and print the warning messages to the log file.

Using Reduced Core Models

PrimeRail supports the capability to generate a reduced core model that consists of Norton equivalent currents of the full-chip switching currents and a reduced-order model of the core power and ground network. The reduced core models are used in the context of system power integrity analysis and optimization. In PrimeRail, both static and dynamic reduced core models are supported.

Why Reduce Core Models Are Needed

As the technology advances, faster switching frequency and lower supply voltages together make the design of a robust core power network more challenging than before. A poorly designed core power network may result in a variety of problems, such as the degradation of the performance, functional failures, and wearing out of metals due to electromigration. Since the core power network is part of a system-level power delivery network (see Figure 12-9), the subject of on-chip power integrity cannot be properly addressed without considering the package and PCB power networks together. However, due to the extremely large size of the system-level power delivery network, analyzing all the three power domains together is not feasible. There are two possible solutions:

1. Analyzing and optimizing the core power network with the package and PCB parts being represented as lumped/distributed RLCK circuits and controlled sources.
2. Analyzing and optimizing the package and PCB power network with the core being represented as a compact model, which simplifies the full-chip switching current signatures and the parasitic of the core power network.

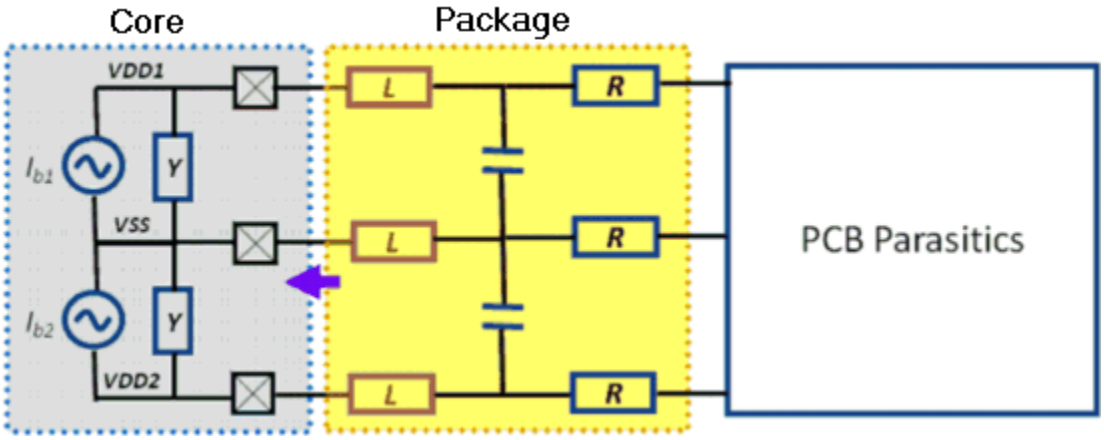
While chip designers usually apply the solution (1) to analyze on-chip power networks, package and PCB designers resort to the solution (2) to conduct off-chip power network analysis and optimization. PrimeRail supports both solutions.

In PrimeRail, the reduced core model can be connected with the package and PCB power networks to conduct off-chip power network analysis and optimization. Compared with using the full-blown core power network in the off-chip analysis, the error induced by using the

reduced core model is expected to be within 5 percent and 10 percent for static and dynamic analysis, respectively. Due to the compact size of reduced core model, the runtime and memory usage required by the off-chip power network analysis can be greatly reduced.

Figure 12-9 shows an example of the system-level power delivery network.

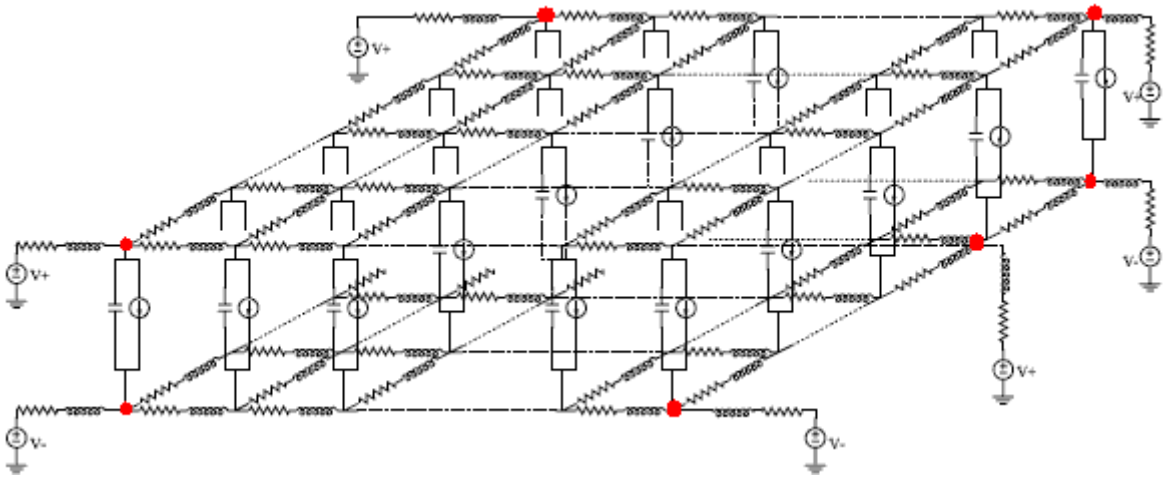
Figure 12-9 System-Level Power Delivery Network



How Reduced Core Models Are Generated

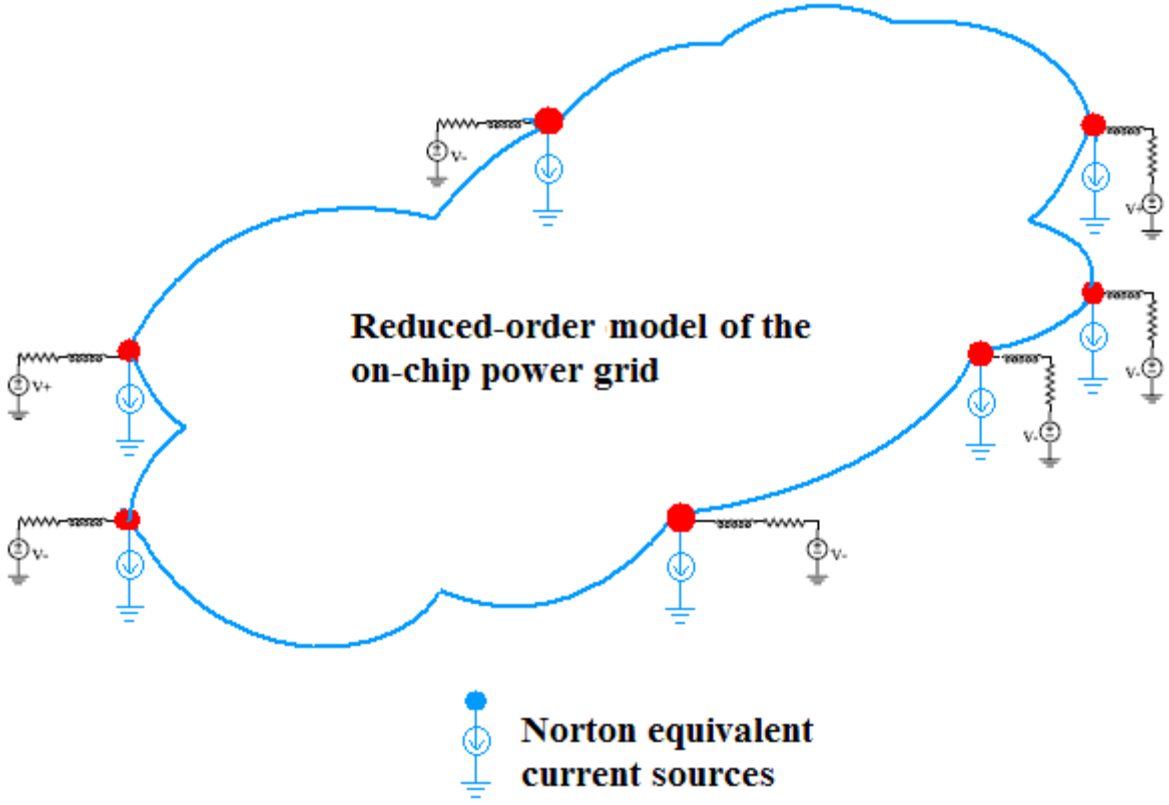
An on-chip power grid is modeled as a linear RLC network with independent time-varying current sources that model the switching currents of the functional blocks, as shown in Figure 12-10. Note the boundary nodes between the core and package power networks are denoted by the red dots.

Figure 12-10 A Power Grid Model



The extremely large circuit size of the power grid model makes it unsuitable for system-level power integrity analysis. A more compact model, called the reduced core model, should be generated to make feasible the computing resources required by the system-level analysis. The reduced core model of an on-chip power grid model consists of two parts: Norton equivalent currents of the internal switching currents and the reduced-order model of the on-chip power grid, as shown in Figure 12-11.

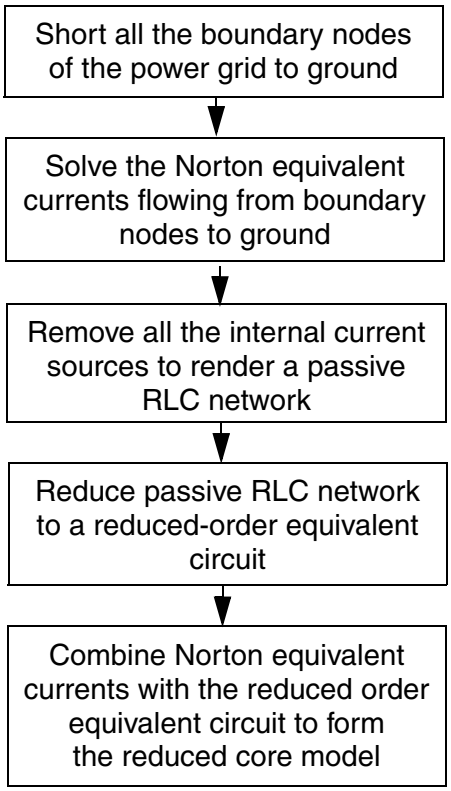
Figure 12-11 Reduced Core Model of the On-Chip Power Grid Model



By shorting all the boundary nodes to ground, the currents flowing from the boundary nodes to ground are the Norton equivalent currents. When all the internal switching currents are replaced by the Norton equivalent current sources, they can be removed from the power grid model so that a passive RLC network remains. The passive RLC network can then be reduced by model order reduction (MOR) algorithms to render a reduced-order model of the on-chip power grid.

Figure 12-12 shows the flow chart of reduced core model generation.

Figure 12-12 Reduced Core Model Generation Flow

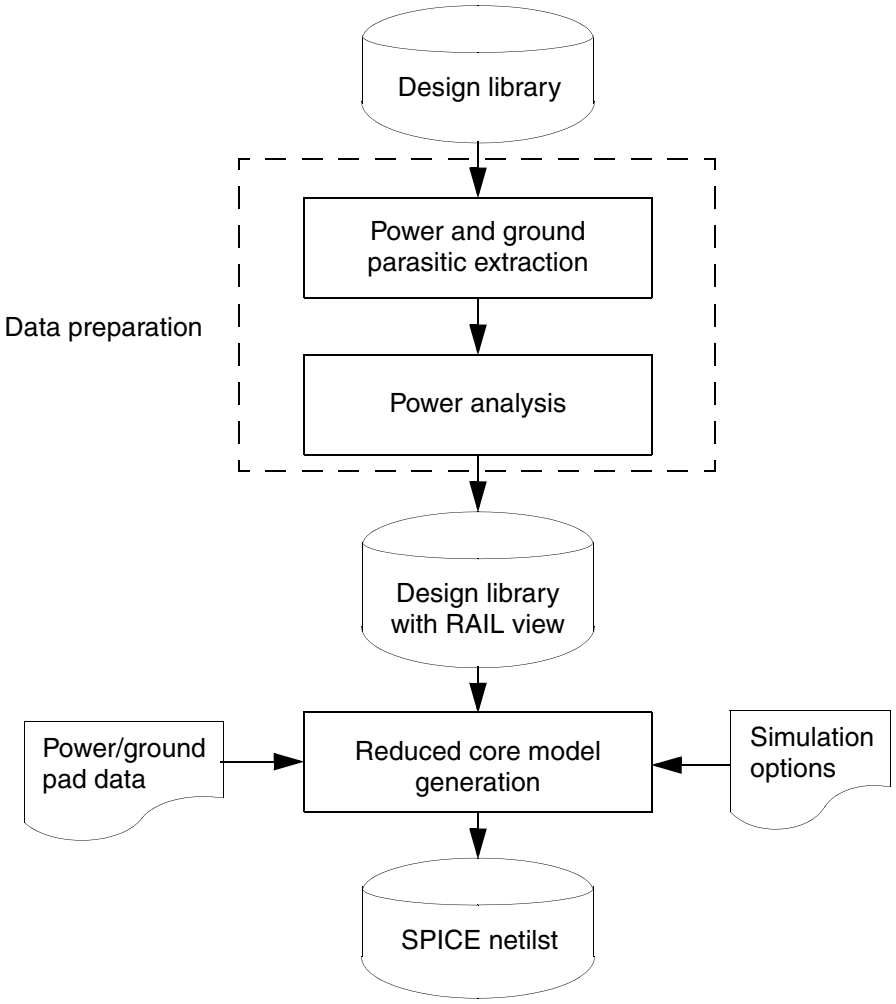


Generating Reduced Core Models

Before generating a reduced core model, you must complete the runs of power and ground net extraction and power analysis on the design so PrimeRail can obtain the necessary power and ground parasitics and the switching currents of all functional blocks from the Milkyway RAIL view. The power and ground pad information is also required to define the boundary nodes. If PrimeRail rail analysis has been performed, the tool can read the power and ground pad information from the RAIL view directly. With the input data, reduced core model generation can be started and the results are saved to a SPICE file. The tool also allows you to adjust the accuracy of the reduced core model by modifying some simulation options, like step size and end time.

Figure 12-13 illustrates the data flow of the reduced core model generation.

Figure 12-13 Reduced Core Model Generation Data Flow



The following list provides additional information to the steps depicted in Figure 12-13:

- Design library with RAIL view: For reduced core model generation, the power and ground parasitic and full-chip switching currents of all functional blocks must be ready in the RAIL view.
- Power and ground pads information: All the power and ground pads input to PrimeRail rail analysis are accepted by reduced core model generation. If you have run through rail analysis on these power and ground pads, you do not have to provide the power and ground pad information because the information is already available in the RAIL view.
- Simulation options: The tool automatically determines the suitable settings based on the power information saved in the RAIL view. You should change the default settings only when you want to adjust the accuracy of the reduced core model.

- Output SPICE file: The output SPICE file consists of Norton equivalent currents and a multiport admittance matrix in the frequency domain. The Norton equivalent currents are represented as current sources with piecewise linear delay model (PWL) waveforms for dynamic reduced core models and as DC current sources for static reduced core models. The multiport admittance matrix is represented as an HSPICE SP model in data frequency table format.

  For more details about HSPICE SP models, see the HSPICE documentation.

Note:

Before generating the reduced core model, PrimeRail first converts the power grid to a sparse matrix and saves it in memory. Since every node of the power grid consumes about 600 bytes of memory, PrimeRail can only handle a power grid network with less than 53 million nodes on a 32-GB memory machine.

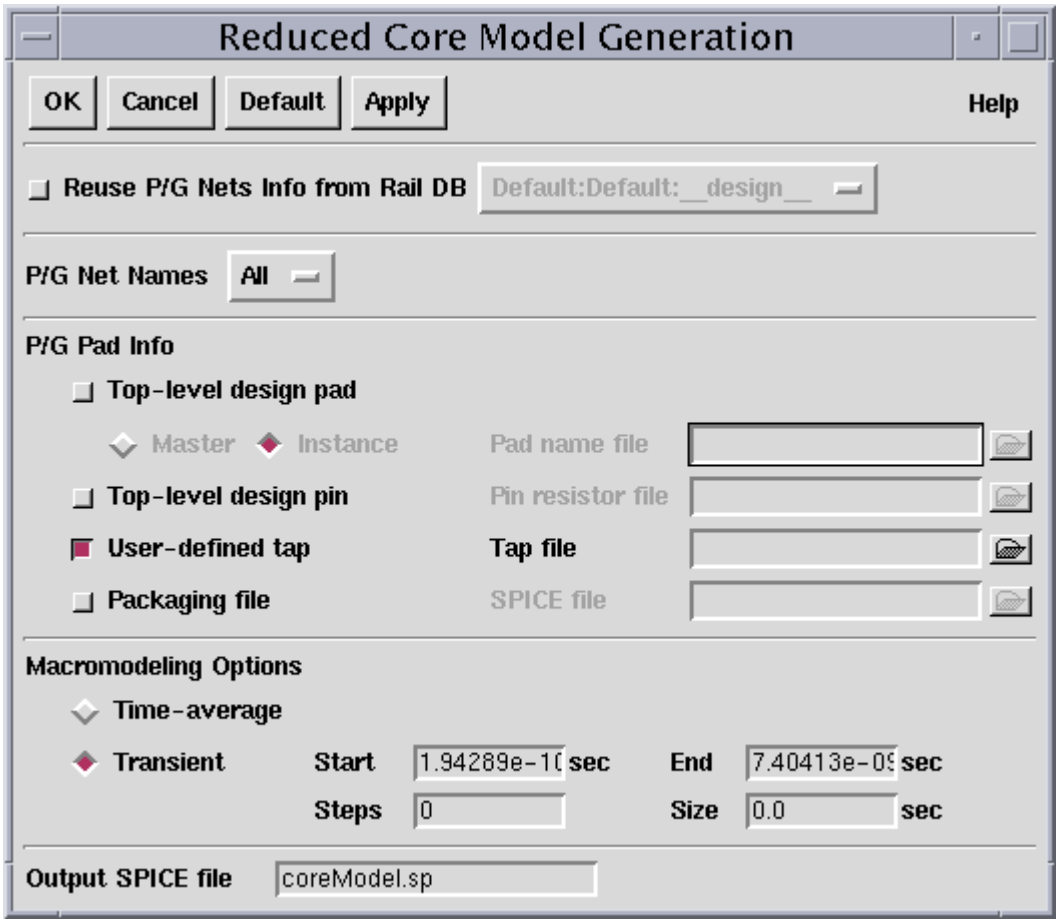
Since most of the temporary data related to reduced-order model generation is saved in disks, the memory usage is much smaller than that of sparse matrix construction. However, to reach a good compromise between runtime and memory usage, some memory is reserved to cache the disk I/O data. The maximum reserved memory is limited to 8 GB.

To generate a reduced core model,

1. Be sure the power and ground parasitic data and full-chip switching currents are available in the RAIL view before generating the reduced core model.
2. Enter `pgGenCoreModel` in the command window.

   The Reduced Core Model Generation dialog box appears, as shown in Figure 12-14.

Figure 12-14 Reduced Core Model Generation Dialog Box



3. Select “Reuse P/G nets info from RAIL DB” if you have run through rail analysis and choose one rail result from the pull-down menu. The tool will then read all the power and ground net information from the RAIL view.

   When selected, all the other input sections will be grayed out except the “Output SPICE file” option.

4. Choose one or multiple power and ground net names from the pull-down menu.

5. In the P/G Pad Info section, define the boundary nodes. This section is fully compatible with that of the Rail Analysis dialog box. (See Figure 12-15 on page 12-38.)

6. In the Macromodeling Options section, specify the following information:

   - Time-average: Choose to enable static reduced core model generation.
   - Transient: Choose to enable dynamic reduced core model generation. The simulation options are filled automatically by the tool, based on the power information in the RAIL view. Note that specifying a smaller time step size decreases the step size of Norton

current waveforms but increases the maximum frequency of the SP model. On the other hand, specifying a larger end time increases the duration of Norton current waveforms but decreases the frequency step size of the SP model.

7. Specify the name of the output SPICE file. The default name is coreModel.sp.
8. Click OK or Apply.

Limitation:

Since the reduced core model generation engine uses some matrix computation API's from the Intel mathematical kernel library, which is supported by Linux platforms only, the generating reduced core models feature is available on AMD64, SUSE32, and SUSE64 platforms only.

The maximum number of boundary nodes is limited to 1,000. This limitation can be removed after the runtime and memory usage of the reduced core model engine is further reduced.

Performing Full-Chip Rail Analysis

After you analyze the reliability effects of each block in the design, run the `poRailAnalysis` command to perform a full-chip rail analysis.

In the full-chip rail analysis, the tool reads in the power ground net extraction data generated for each block of the design by the `poExtractPGParasitics` command and assembles a global RC (resistance-capacitance) tree. You can further refine this global RC tree by adding the intrinsic parasitics determined during current waveform generation as well as the RLC package parasitics, inserted decoupling capacitors, hard macro models, and modeled power management cells.

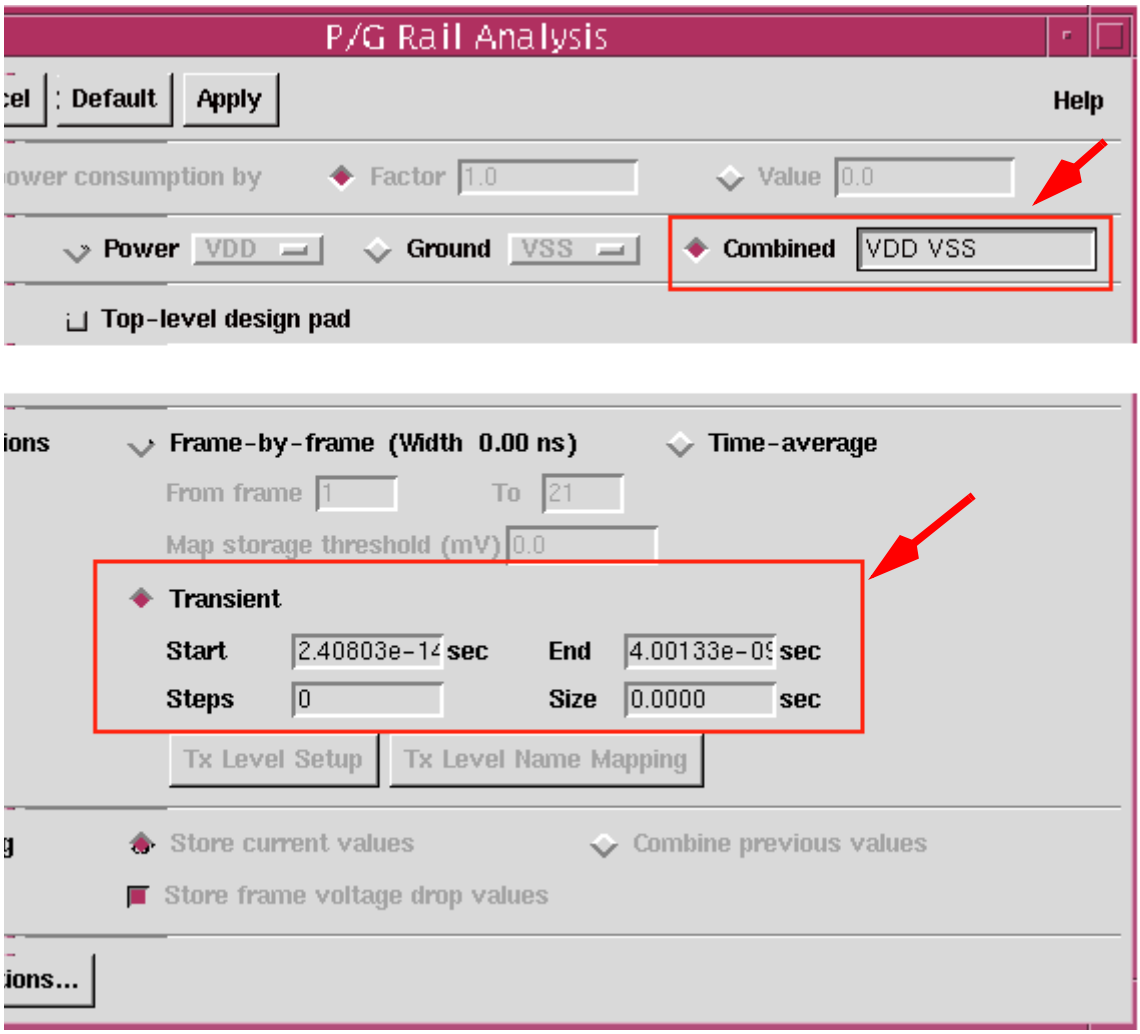
The resulting RLC tree is then loaded with the dynamic current waveforms that are determined by the `poCalculatePower` command (see "Calculating Power and Current Waveforms" on page 7-13). The `poRailAnalysis` command then simulates the entire system according to the time window and step you specify. The resulting voltage waveforms are captured in the FSDB format and can be analyzed in a standard waveform viewer, such as nWave.

To perform full-chip rail analysis, you need to first run the `poExtractPGParasitics` command from the top-level after you have imported the DWM models (or the generated DWM-lite model). Then, you need to run the `poCalculatePower` command to determine current waveforms for standard cells and hard macros together before you can run dynamic rail analysis using the `poRailAnalysis` command. The full-chip rail analysis steps are described below:

1. Enter `poRailAnalysis` or choose Cell-level Analysis > Rail Analysis – Rail Analysis.

The P/G Rail Analysis dialog box opens.

Figure 12-15 P/G Rail Analysis Dialog Box



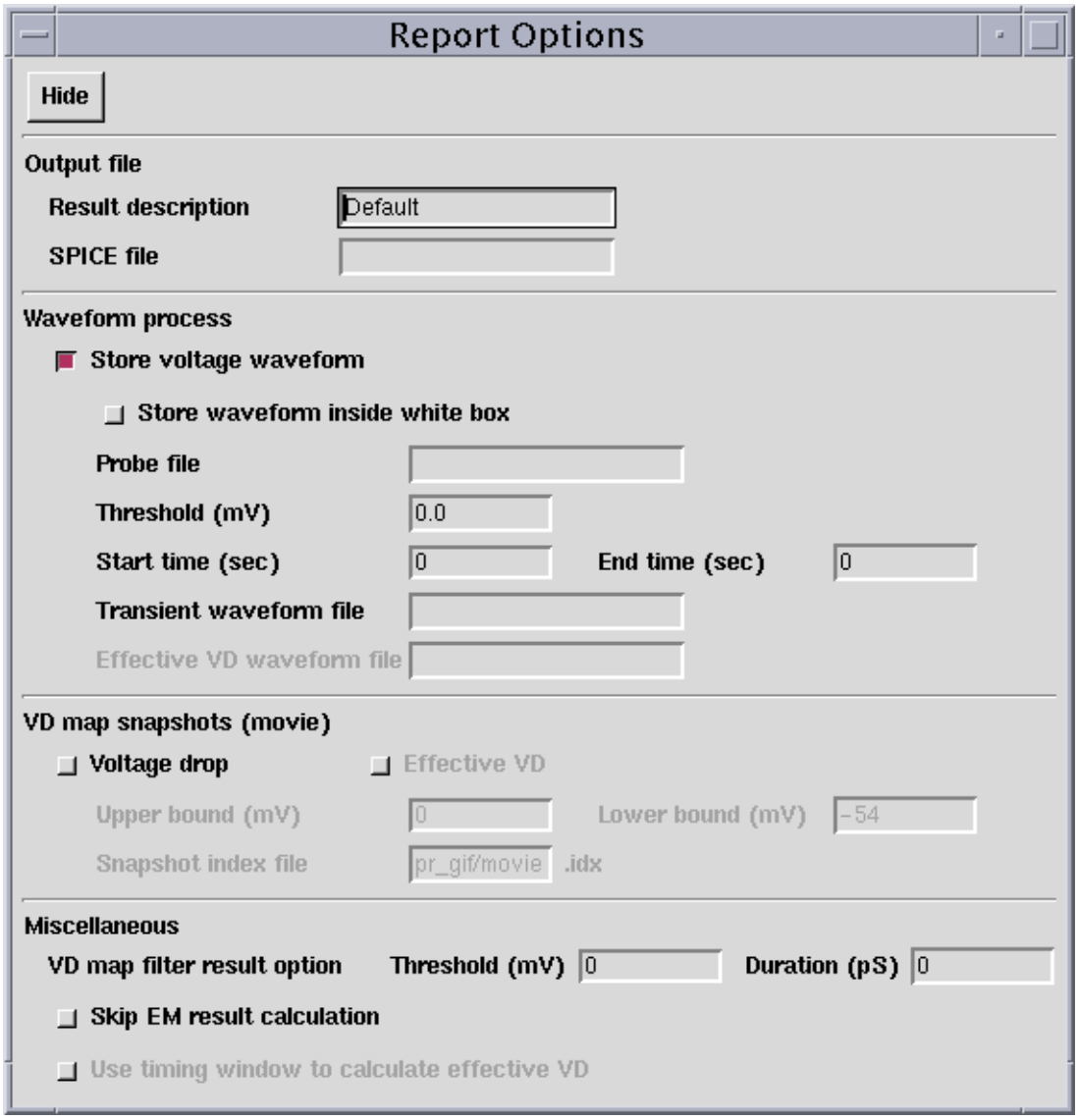
2. In the “P/G net info” section, select Power or Ground and specify the name of the power or ground net to be analyzed.

   If you want to analyze multiple power and ground networks, select Combined and enter the names of the nets in the text box.

3. In the “Analysis options” section, select Transient for full-chip rail analysis. Specify starting time, ending time or time step size to be used during simulation.

4. Click the Report Options button to open the Report Options dialog box, as shown in Figure 12-16.

Figure 12-16 Setting Report Options During Rail Analysis



5. Specify the options as necessary.

   For a detailed description of each option in the dialog box, see the online Help for the command.

6. Click OK or Apply.

13

Performing Transistor-Level Signal Net Electromigration Analysis

PrimeRail provides the signal net dynamic analysis functionality that enables you to check the current density violations for signal nets in the design. Similar to the transistor-level dynamic analysis flow, PrimeRail also heavily leverages the established Synopsys dynamic verification products, like NanoSim for SPICE simulation and StarRC for transistor-level extraction.

This chapter contains the following sections:

Dynamic Signal Electromigration Analysis Flow

Running signal net electromigration analysis in PrimeRail consists of several stages. The flow first starts by an initial screening stage in which PrimeRail drives StarRC to perform a transistor-level extraction and subsequently drives NanoSim to simulate the resulting back-annotated netlist. The StarRC extraction at this stage extracts only the capacitance data for every net in the design and the reduced capacitance and resistance data for the input pins.

With the extracted data from StarRC, the NanoSim simulation at the screening stage runs significantly faster than a simulation with full resistance and capacitance extraction data and provides a pessimistic for signal net currents. This means that the nets that do not violate a user-specified threshold in simulation are pruned from the detailed analysis involving full RC parasitics.

Note:

You can define the electromigration threshold information in an Advanced Library Format (ALF) file. The electromigration rule file contains the layer, via, site, and macro cell definitions.

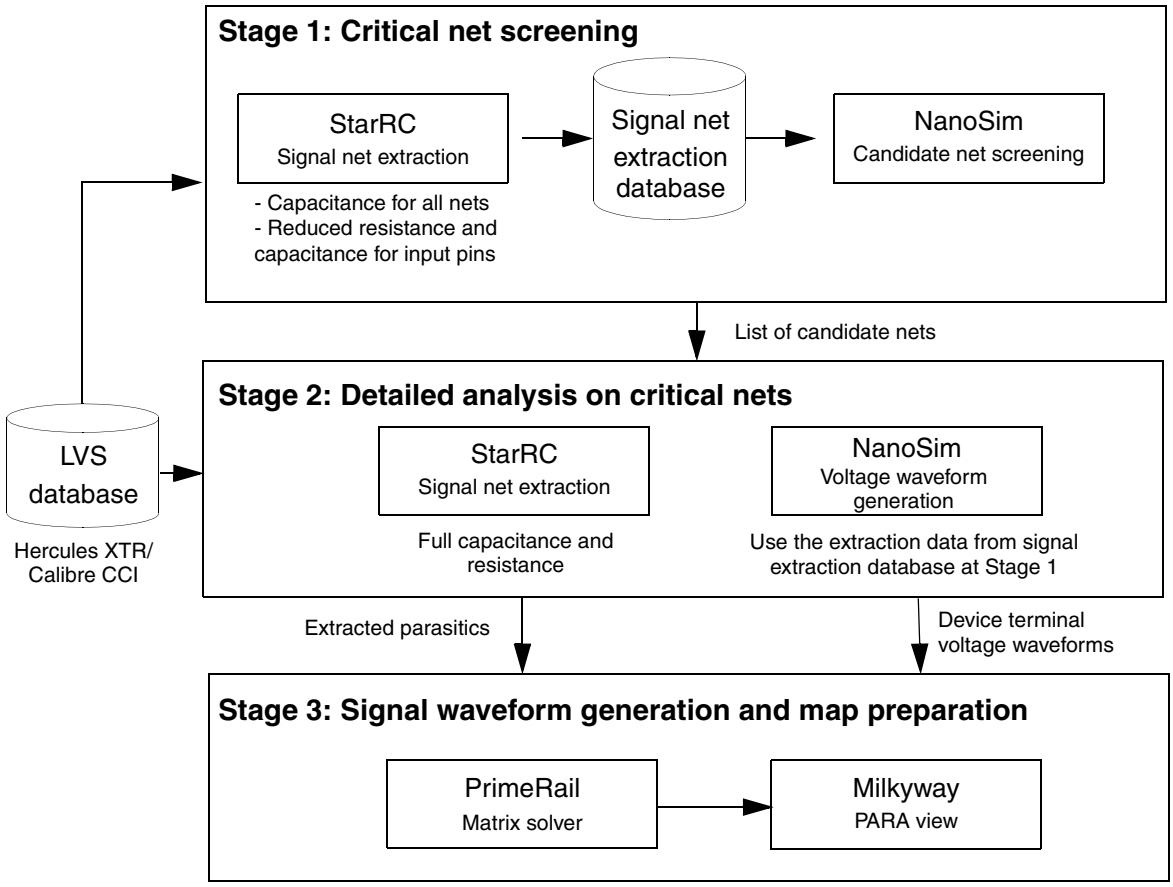
The second stage is to take the list of nets determined as potentially critical in the previous pruning step and to drive StarRC to perform a full resistance and capacitance extraction on the list of nets given. Based on reduced parasitic data from the previous stage, NanoSim simulates the critical nets and determines the signal waveforms of the device terminals connecting to the critical nets.

The final stage is to create a Milkyway PARA view that contains the detailed resistance and capacitance parasitics and device terminal waveforms for the nets simulated in the second stage. Based on the detailed parasitics, PrimeRail invokes the matrix solver to calculate the currents for each net segment of the critical nets.

The signal net electromigration analysis in PrimeRail supports the LVS data both from Hercules and Calibre.

Figure 13-1 illustrates the signal net electromigration dynamic analysis flow in PrimeRail.

Figure 13-1 The Transistor-Level Signal Net Electromigration Dynamic Analysis Flow



Input Files

To perform the StarRC extraction and NanoSim simulation successfully, you must provide the following files:

- StarRC NXTGRD and layer map files
- StarRC skeleton command file
- SPICE device models
- NanoSim configuration file
- NanoSim simulation vectors

Analyzing Current Violations for Signal Nets

PrimeRail provides the `poSigDynamicAnalysis` command for conducting the signal net electromigration analysis. With the command, you can run StarRC extraction, NanoSim circuit simulation, and current violation calculation in a single user interface. It is unnecessary to invoke each individual tool for completing the flow.

The signal net electromigration analysis in PrimeRail consists of several stages. To control which stage to run in the flow, use the `poSigEMFlowCtrl` switch before invoking the `poSigDynamicAnalysis` command.

The following is the syntax of the `poSigEMFlowCtrl` switch:

```
define poSigEMFlowCtrl "option1 option2 ..."
```

where the option keyword can be any of the following:

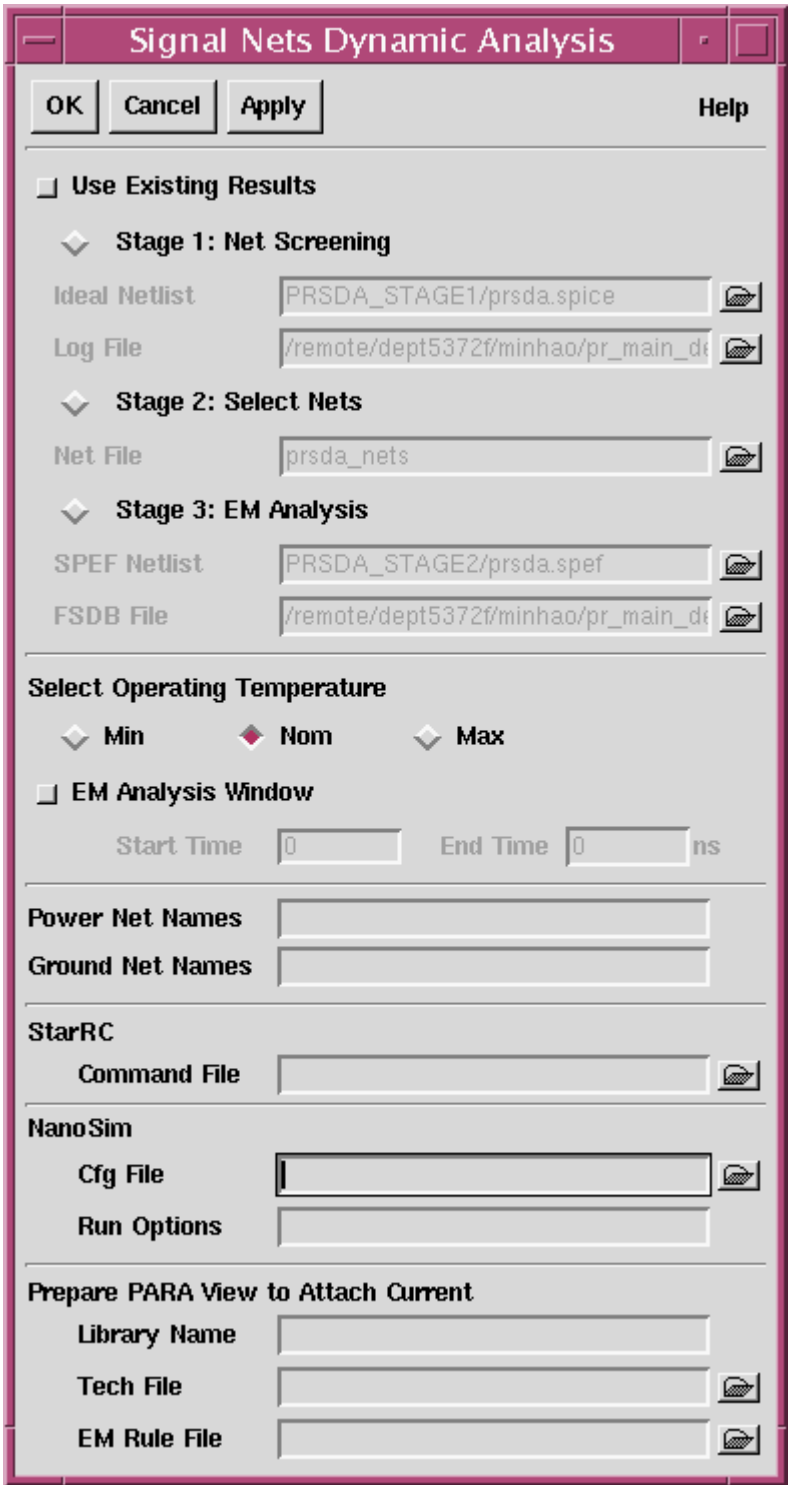
- skip_s1_rcxt: Skip the StarRC extraction at the first stage. You need to make sure the extracted data is present in the memory.
- skip_s2_rcxt: Skip the StarRC extraction at the second stage.
- stop_after_s1_rcxt: Stop the analysis process when the StarRC extraction is done at the first stage.
- stop_after_s2_rcxt: Stop the analysis process when the StarRC extraction is done at the second stage.

To conduct a signal net dynamic electromigration analysis,

1. Enter `poSigDynamicAnalysis` in the command window or choose Transistor-level Dynamic Analysis > Transistor-level – Signal EM Analysis.

   The Signal Nets Dynamic Analysis dialog box appears.

Figure 13-2 The Signal Nets Dynamic Analysis Dialog Box



2. Select Use Existing Results if you already have the extraction or simulation data and want PrimeRail to calculate possible current violations for signal nets based on the existing data.

   If you do not have the necessary extraction and simulation data, deselect this option and proceed to step 3.

   Because `poSigDynamicAnalysis` generates different output files in each process, you can choose from which process in the flow you want PrimeRail to capture dynamic currents for signal nets. This helps you debug the flow when you are analyzing a large design.

   Select one of the following options according to the stage of the flow or the types of files you have:

   - Stage 1: Net Screening. Select when you want to generate a net file based on the ideal netlist and the NanoSim log file and continue with the analysis process.

     Ideal Netlist – Specify the names of the ideal netlist file. The ideal netlist is generated during the net screening process.

     Log File – Specify the name of the log file that is generated during the net screening process.

   - Stage 2: Select Nets. Select when you want to skip the screening process and continue with the analysis process.

     Net File – Enter the name of the net file that is generated when the screening process is complete.

   - Stage 3: EM Analysis. Select when you want to generate the current file by invoking the matrix solver and continue with the rest of the process.

     SPEF Netlist – Enter the name of the StarRC SPEF file.

     FSDB File – Enter the name of the NanoSim FSDB file.

3. In the Select Operating Temperature section, choose the corner of the temperature to be used throughout the flow, including extraction, circuit simulation, and parasitic backannotation. The default is Nom.

   Note:
   The definition of these three operating temperature corners is the same as that in the Milkyway technology file.

4. Select EM Analysis Window and specify the start time and end time. Use this option when you want to use only a certain part of simulation results for EM analysis. However, the simulation time must start from time 0, based on the nature of circuit simulation. You must manually set the NanoSim -t option.

5. Enter the names of power and ground nets separated by spaces. The listed nets will be excluded during the analysis. Wildcard usage is supported.

6. In the StarRC section, enter the name of the StarRC command file to be used for StarRC extraction. The command also automatically appends additional options to the file during extraction to preserve the signal net geometries.
7. In the NanoSim section, specify the NanoSim configuration file and additional command line options to be passed to NanoSim.
8. In the “Prepare PARA View to Attach Current” section, enter the name of the Milkyway design library and technology file to be used for creating the PARA view.
9. Enter the name of the electromigration rule file that contains the user-defined thresholds of the design being analyzed. Currently, only the ALF format is supported.

   For a detailed description about how to load an electromigration rule file to Milkyway, see“Loading an ALF File” on page 4-9.

   If your electromigration rules contains length and via rules, see the detailed descriptions in “Applying Length-Dependent Electromigration Rules” on page 13-18.
10. Click OK or Apply.

    When analysis is complete, PrimeRail writes the log message to the log file and the command window.

Checking Signal Current Violations on Design Resistors

PrimeRail allows you to run signal electromigration analysis on a user-defined design resistor. In this way, you can treat the device just like a regular interconnect wire.

You must provide a mapping file that describes the relationship between resistor model card names and physical layer names, as defined in the StarRC ITF file.

The following is an example of the mapping file:

```
rppolywo              poly
ropolyli              poly
rm1s                  metal1
rm2s                  metal2
...
```

To check for current violations on design resistors, type the following before invoking the `poSigDynamicAnalysis` command:

```
define poXResLayerFile <layerFileName>
```

Viewing Transistor-Level Signal Net Electromigration Analysis Results

When the signal net electromigration analysis at the transistor level is complete, use the `poSigDisplayEMMap` command to view and check the possible current density violations for signal nets in the design.

Before you execute the `poSigDynamicAnalysis` command, be sure you open the library and the cell for which its signal net electromigration map is to be displayed.

The following are the switches available to work with the `poSigDisplayEMMap` command:

`poEMHealingFactor`

If you want to apply a healing factor to model bidirectional signal nets, set the `poEMHealingFactor` switch before invoking the `poSigDisplayEMMap` command:

```
define poEMHealingFactor healing_factor
```

The effective average current is defined by the following formula:

```
Max [(curr_avg_positive - healing_factor *
curr_avg_negative), (curr_avg_negative - healing_factor
* curr_avg_positive)]
```

where `curr_avg_positive` and `curr_avg_negative` are positive and negative part average of the current, respectively.

`poEMScaleFactor`

When you use electromigration rules and want to apply a scaling factor to the electromigration rules, set the `poEMScaleFactor` switch before invoking the `poSigDisplayEMMap` command:

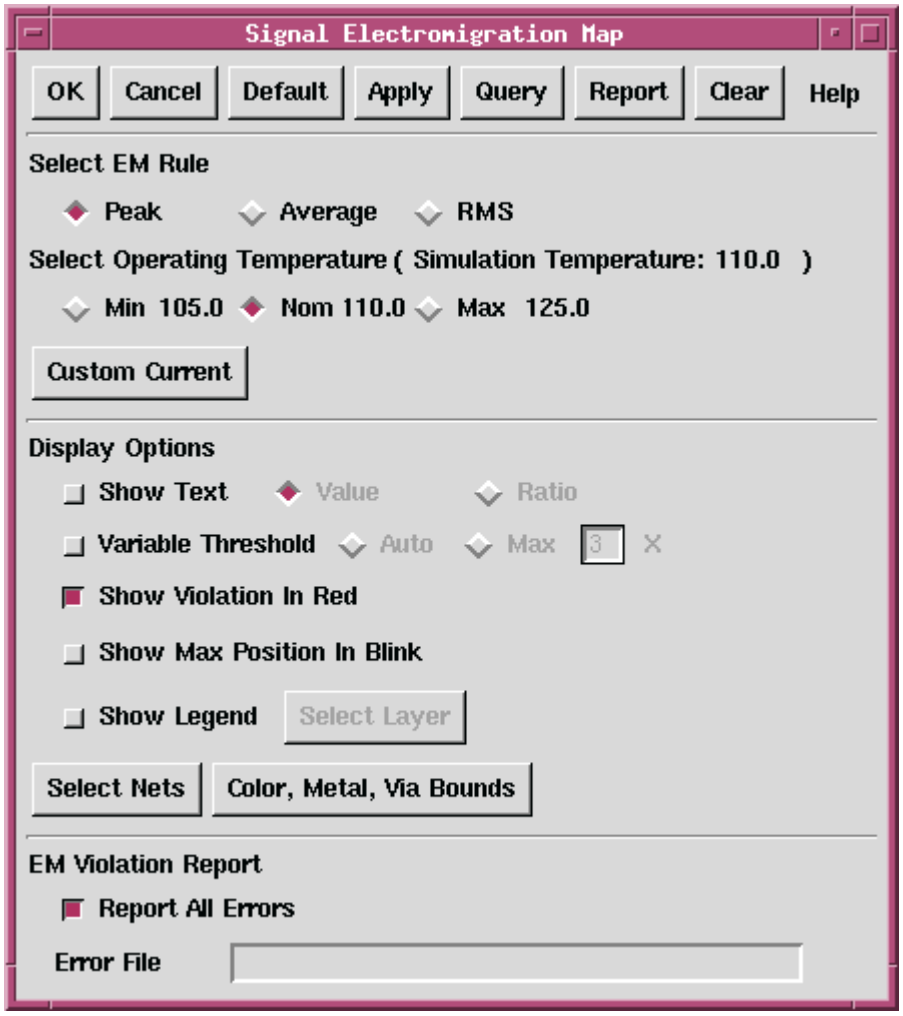
```
define poEMScaleFactor scale_value
```

To display a signal electromigration map,

1. Enter `poSigDisplayEMMap` in the command window or choose Transistor-level Dynamic Analysis > Display – Display Signal EM Map.

   The Signal Electromigration Map dialog box appears.

Figure 13-3 Signal Electromigration Map Dialog Box



2. In the Select EM Rule section, select to check the peak, average, or root mean square (RMS) current density set in the electromigration rule file. The default is Peak.

   The peak current in the electromigration rule is divided by a duty ratio (peak_current/duty_ratio). If you want to display the actual peak current in the map, enter the following in the command window before invoking the `poSigDisplayEMMap` command:

   ```
   define poEMActualPeak 1
   ```

   When the `poEMActualPeak` flag is set, PrimeRail ignores the duty ratio and uses actual peak currents for map display and electromigration rule checking.

3. In the Select Operating Temperatures section, select the corner temperature to be used with the time-average and RMS electromigration rule file that is width or temperature dependent.

The tool automatically aligns temperature corners used during circuit simulation and EM rule checking. The temperature corners are used through the entire flow, including extraction, circuit simulation and parasitic backannotation.

4. Click Custom Current and enter the formula you want to use for checking the electromigration rules in the dialog box that opens.

   Type the formula in the text box. For example,

   ```
   abs(curr_avg)+ln(curr_peak)+sin(curr_rms)
   ```

   When the above formula is applied, the current on each resistor is changed to the sum of the absolute value of average currents, the nature log of peak currents, and the sinusoid function of RMS currents. The current for average rule checking (that is, the average current of the resistor when no custom current is defined) will then be changed to the defined custom current.

   This allows you to use the formula other than peak, average and RMS for current checking. The custom formula overrides the default method of average (or RMS) electromigration checking.

   Table 13-1 and Table 13-2 list the supported variables, functions and constants that can be used in custom current.

Table 13-1 Built-In Variables Supported in Custom Current

Variable	Description
curr_avg	Average current
curr_custom	Currents stored with the custom formula during `poSigDynamicAnalysis`
curr_dr	Duty ratio
curr_peak	Peak current
curr_rms	RMS current

Table 13-2 Built-In Functions and Constants Supported in Custom Current

Function /Constant	Description
_e	exp=2.7182 ...
_pi	pi=3.14 ...
abs	Abs

Table 13-2 Built-In Functions and Constants Supported in Custom Current (Continued)

Function /Constant	Description
acos	ACos
acosh	ACosh
asin	ASin
asinh	ASinh
atan	ATan
atanh	ATanh
avg	Avg
cos	Cos
cosh	Cosh
exp	Exp
if	Ite
ln	Ln
log10	Log10
log2	Log2
max	Max
min	Min
rint	Rint
sign	Sign
sin	Sin
sinh	Sinh
sqrt	Sqrt
sum	Sum

Table 13-2 Built-In Functions and Constants Supported in Custom Current (Continued)

Function /Constant	Description
tan	Tan
tanh	Tanh

5. In the Display Options section,

   - Show Text – Select to make the current density's value or ratio visible in the map. The default is off.

     The current density value can appear as either an absolute value or as a ratio to a specified threshold. The default is Value.

   - Variable Threshold – Displays the areas of the design where current density values are greater than the limit specified in the electromigration rule file.

     Auto – Automatically set the default value to the maximum number of current density and threshold ratio.

     Max – Enter a threshold to be used in scaling the limit by multiplying the upper limit derived from the electromigration rule file.

     For example, if the limit specified in the electromigration rule file is 12 amperes per centimeter and the threshold is set to 3, the tool will show and scale the current densities that are up to 3 times higher than the limit—that is, current densities between 12 and 48—in 12 steps. The current densities over 48 will be colored in red as violations.

     When selected, the Show Violation In Red option will not be available.

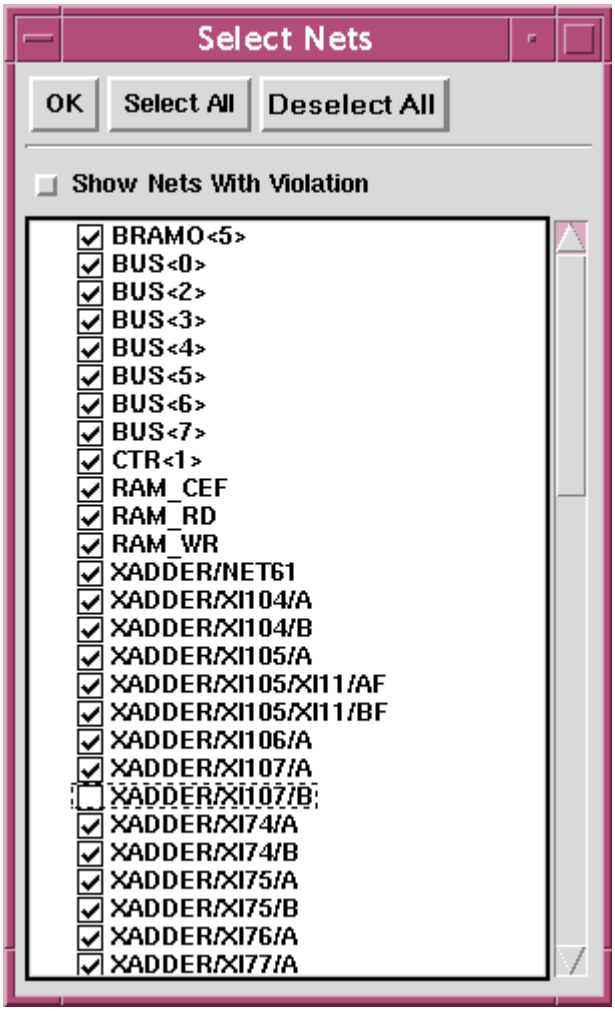
   - Show Violation In Red – Select to color the location in red where a current violation exists.

   - Show Max Position In Blink – Select to show a blinking rectangle where the maximum current density is located. Two rectangles can appear in the map for one current density because of different current density units; one is for the conducting layer, the other for the contact layer.

   - Show Legend – Select and click Select Layer to choose a layer (conducting layer or contact layer) for which its upper and lower bound limits are to be shown in the map. Specify the upper and lower bounds in the Color, Metal, Via Display dialog box (see Figure 13-6 on page 13-20). The default is off.

     When the variable threshold is used, the tool will show the corresponding scaling factor of the range limit (that is, between 1x to Nx) for each of 12 steps in the legend.

- Select Nets – Click to choose the nets to be displayed in the map through the Select Nets dialog box that appears.



  Check the nets that you want to show in the Signal Electromigration map. If you want to display only the nets with current violations, select Show Nets With Violation.

  Click Select All to select all the nets in the list and show them in the map.

  Click Deselect All to uncheck all the nets in the list.

  Click OK to apply the changes and go back to the Signal Electromigration Map dialog box.

6. Click Color, Metal, Via Bounds to configure necessary settings for the Signal Electromigration Map to be displayed. See the following section, “Setting Color, Metal, and Via Bounds,” for details.

7. In the EM Violation Report section, enter the name of the file you want to assign to the output report file that contains the current violation information for the design. See “Generating EM Violation Report” on page 13-16.” for details.
8. Click Apply.

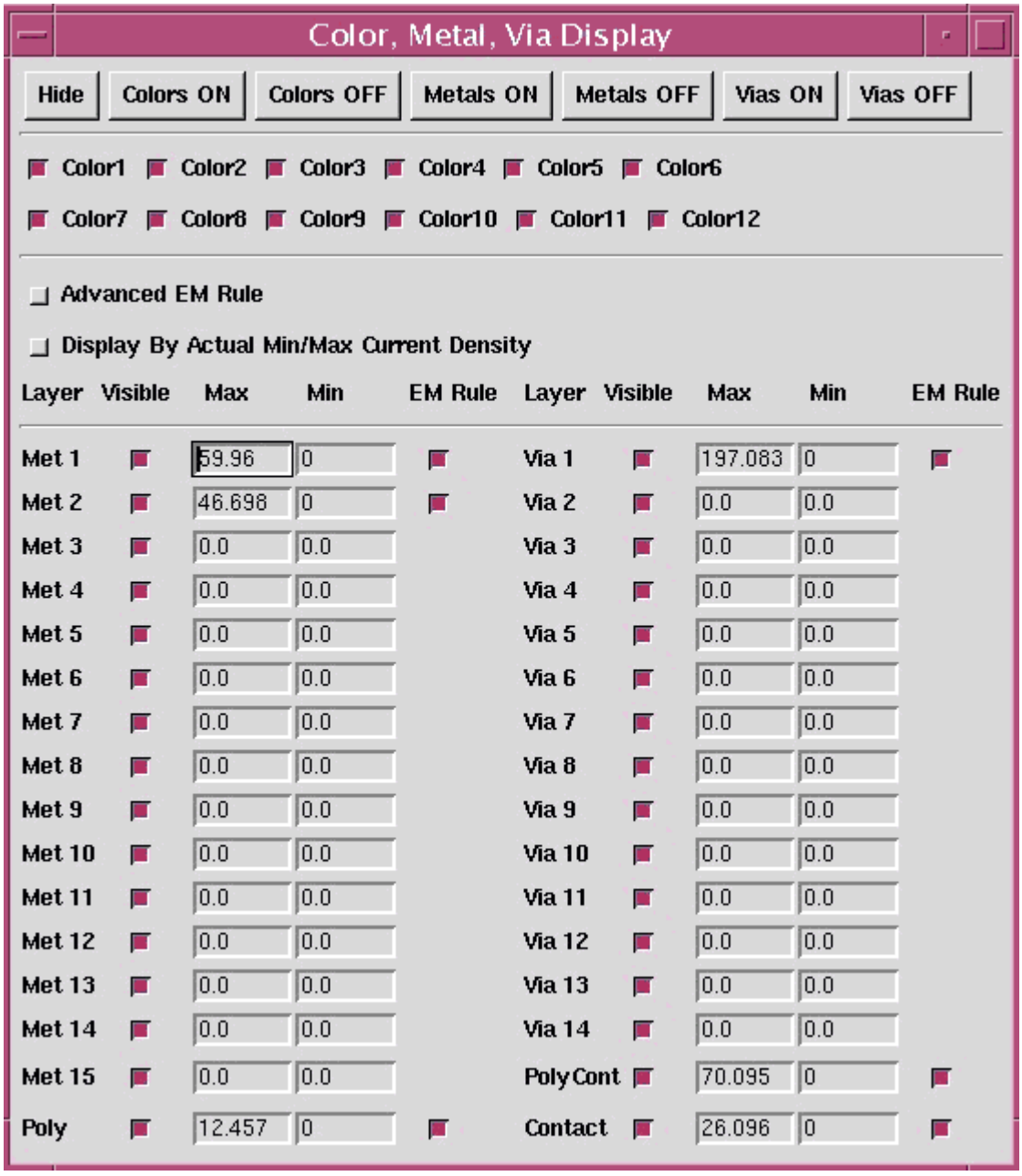
Setting Color, Metal, and Via Bounds

For each metal layer, you can specify whether to show the wire segments of the layer, as well as the lower and upper current density bounds. The current density unit for conducting layers is ampere per centimeter, whereas the unit for contact layers is ampere per centimeter square.

To set steps, metals, and vias for display,

1. In the Signal Electromigration Map dialog box, click the Color, Metal, Via Bounds button to open the Color, Metal, Via Display dialog box.

Figure 13-4 Color, Metal, Via Display Dialog Box



2. Specify whether to show colors, metals, or vias in the electromigration map by clicking the Colors ON, Colors OFF, Metals ON, Metals OFF, Vias ON, and Vias OFF buttons.
3. Select the colors you want to show on the electromigration map. PrimeRail allows you to define up to 12 different colors.

4. Select Advanced EM Rule if length and via rules are considered. For more information, see “Applying Length-Dependent Electromigration Rules” on page 13-18.
5. Select Display By Actual Min/Max Current Density if you want to use the actual minimum or maximum current density to display the electromigration map on all selected layers. Note that this option overrides all wire electromigration rule selections.
6. Select the metal you want to show by clicking the Visible button.

   Modify the upper and lower bound values, if necessary. If you loaded an electromigration rule file, the EM Rule block of the metal is selected to indicate the presence of the EM rule.
7. Select the via layer you want to show by clicking the Visible button.

   Modify the upper and lower bound values, if necessary. If you loaded an electromigration rule file, the EM Rule block of the via layer is selected to indicate the presence of the electromigration rule.
8. Click Hide to close the Color, Metal, Via Display dialog box and return to the Electromigration Map dialog box. Click Apply to redraw the map based on the changes you make.

Generating EM Violation Report

In addition to viewing current violation results from the map, you can choose to generate an EM violation report file that lists the signal nets or resistors whose current density values exceed the peak, average, or RMS current densities as defined in electromigration rule file. PrimeRail prints all the violations found to the EM violation report file, no matter which mode of the EM rule (peak, average, or RMS) you choose in the Signal Electromigration Map dialog box. The EM violation report is in the ASCII format.

To generate an EM violation report file, enter the name of the file to be created in the Error File text box at the bottom of the Signal Electromigration Map dialog box (see Figure 13-3 on page 13-9). Click Apply and then Report at the top of the Signal Electromigration Map dialog box to generate the EM violation report file.

The following is an example of the EM violation report file:

```
*********************************************************
* Title           : Signal Net Electromigration Violation Report
* Date            : Fri Feb  2 10:42:19 2007
*
* Vendor          : Synopsys
* Program         : PrimeRail
* Version         : Z-2007.03-Development for AMD.64 --
                    Fri Feb  2 10:23:14 CST 2007
*
```

```
* Design         : mpu
* Hier Separator : /
* Temperature    : 110.0 C
*
* Units:
*   Length Unit        : um
*   Current Unit       : mA
*   Conduct Layer Density Unit : mA/um
*   Contact Layer Density Unit   : mA/VIAs (mA/um^2 if VIA
                                    count is unavailable )
*   Conduct Layer Density Threshold Unit: mA/um
*   Contact Layer Density Threshold Unit : mA/VIAs (mA/um^2
                               if VIA count is unavailable )
*
* Total EM violations    : 478
* EM Rule Table:
* Scaling Factor         : 1.000000
*
*
*
*   Layer: metal1      Mode: average
*                | 1.000e+02 | 1.100e+02 | 1.250e+02 | Temperature
*
-----------+-----------+-----------+-----------+-----------------
-
*     1.000e-01  | 2.290e-01 | 1.600e-01 | 5.700e-02 |
*     1.000e+00  | 2.810e+00 | 1.960e+00 | 7.000e-01 |
*     1.000e+01  | 2.850e+01 | 1.990e+01 | 7.100e+00 |
*
-----------+-----------+-----------+-----------+-----------------
-
*     Width      |                                     | Max. Current
*
*
*   Layer: via1      Mode: average
*                | 1.000e+02 | 1.100e+02 | 1.250e+02 | Temperature
*
-----------+-----------+-----------+-----------+-----------------
-
*     1.690e-02  | 2.710e-01 | 1.890e-01 | 6.700e-02 |
*     3.380e-02  | 5.420e-01 | 3.780e-01 | 1.340e-01 |
*
-----------+-----------+-----------+-----------+------------------
-
*     Area       |                                     | Max. Current
*
*********************************************************************

CELL:mpu NET:BUS<0>

NODE   LAYER    No MODE CURRENT (>CURRENT    WIDTH     NEW_WIDTH
(LEFT, BOTTOM) (RIGHT, TOP)
RES    NAME             DENSITY  DENSITY     AREA      NEW_AREA
```

```
NAME                                THRESHOLD)  VIACNT    NEW_VIACNT
=================================================================
R1     MT2       7 rms     3.36 (>2.984e+00) 1.2 1.35
(184.700, 250.600) (185.900, 261.350)
R2     MT2       7 rms     3.11 (>2.984e+00) 1.2 1.25
(184.700, 210.600) (185.900, 250.600)
R4     MT2       7 rms     3.01 (>2.984e+00) 1.2 1.21
(184.450, 206.300) (185.650, 209.300)
```

Applying Length-Dependent Electromigration Rules

To apply advanced electromigration rules that include length and via rules when displaying a signal electromigration map, run `poSigDisplayEMMap` and click the Color, Metal, Via Bounds button to open the Color, Metal, Via Display dialog box (see Figure 13-3 on page 13-9). Select Advanced EM Rule and other necessary options. Click Hide to close the Color, Metal, Via Display dialog box and return to the Electromigration Map dialog box. Then click Apply to redraw the map based on the changes you make.

The advanced electromigration rules from the foundry will be multiplied by a scaling factor, called multiplication factor. This multiplication factor is calculated based on the length and via counts of vias and nets.

For example, when no vias are in the wire being analyzed, the multiplication factor is the length factor. However, when the wire has vias, the multiplication factor will be decided by the following equation:

$$\text{multiplicationFactor} = \max[\min(\text{lengthFactor}),\text{viaFactor}]$$

Using Length-Dependent Electromigration Rules

PrimeRail supports the use of length-dependent electromigration rules when analyzing signal nets at the transistor level. The tool multiplies a factor to the original maximum current density allowed on a wire. The magnitude of the factor depends on the length of the wire and is limited by a certain range (by default, the factor is set to be from 1 to 4). Consult with your process engineer or documentation to determine the appropriate values.

The average current rule is multiplied by a factor, called length factor (LF), that is dependent on the wire length or the adjacent wire length of a via.

Equation 13-1 Length Factor for Length-Dependent Electromigration Rules

$$\begin{bmatrix} 1 & \iota \geq 20 \\ \frac{20}{\iota} & 5 \geq \iota \geq 20 \\ 4 & \iota \leq 5 \end{bmatrix}$$

where ι is the length of the wire, in microns.

Length i is defined as the end-to-end longest current route on a connected metal layer, as shown in Figure 13-5. The current route is broken by vias.

Figure 13-5 Defining Length When Length-Dependent Electromigration Rules Are Applied

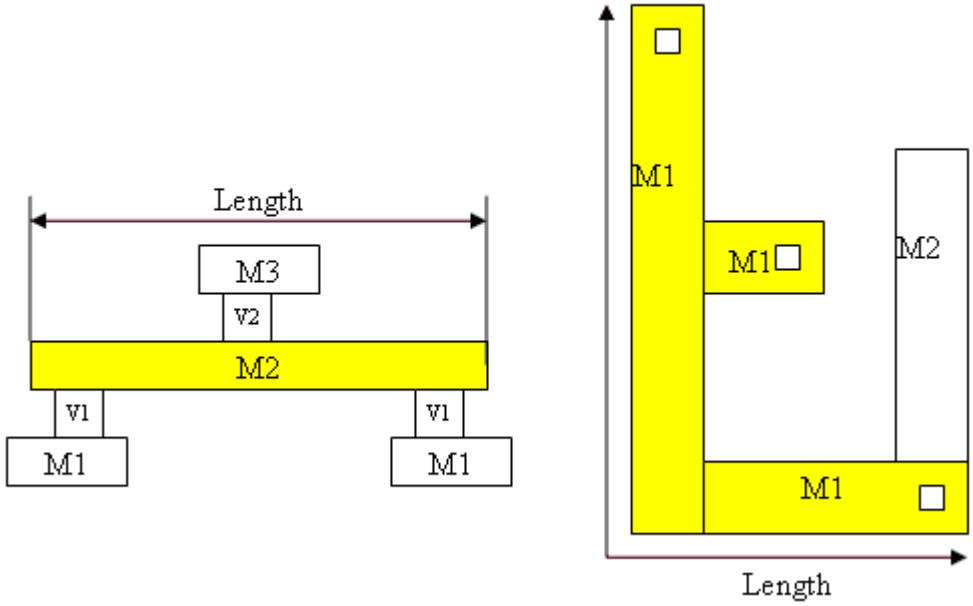
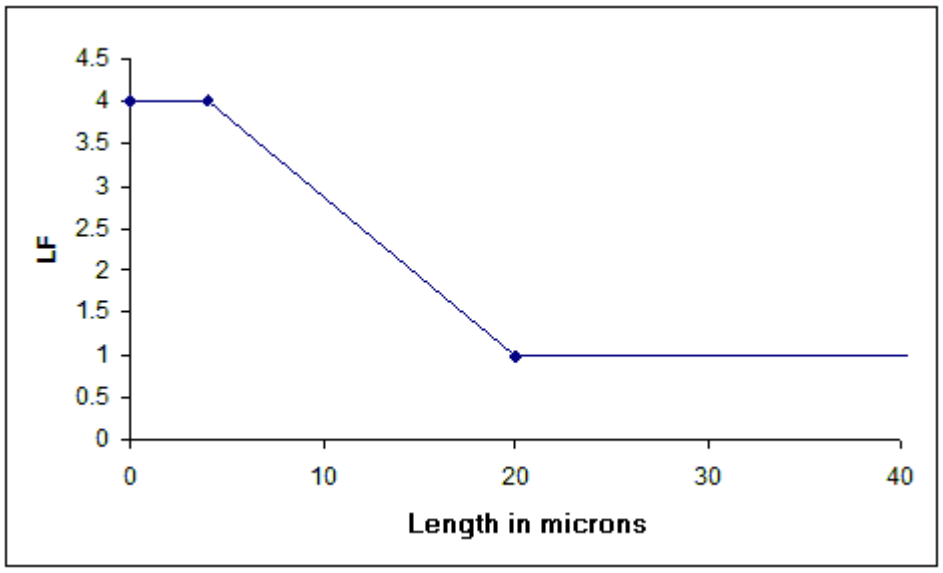


Figure 13-6 shows how the length factor varies with the length of a wire.

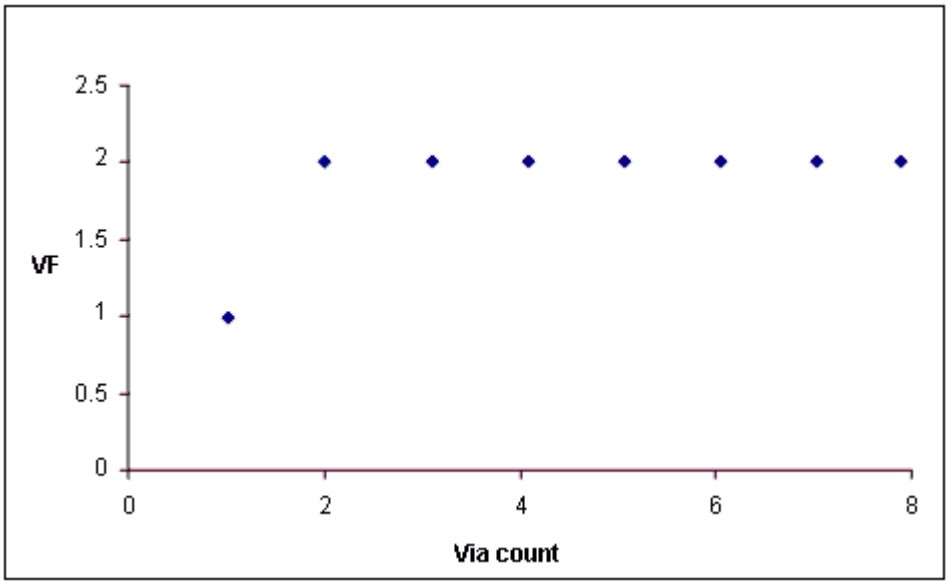
Figure 13-6 Length Factor Table



The length factor of a via is defined by the smallest length factor of wires to which it connects. For example, for the via V1 that connects to nets N1 (metal 1, length factor is 2) and N2 (metal 2, length factor is 3), the length factor is 2.

For via array rules, the maximum average current per via is doubled for a via array (more than two vias). Therefore the tool defines the via factor (VF) by adjusting the electromigration rule on vias (see Figure 13-7).

Figure 13-7 Via Factor Table



Note that the length and via array rule cannot be collateral. The tightest Imax is thus chosen for calculation.

A

Backward Compatibility

In version A-2008.06, some of the commands are either changed or removed. For backward compatibility reasons, this chapter provides the information for the commands that are either removed or changed. For a complete description on the analysis flow, refer to the related chapters in this user guide.

This chapter includes the following topics:

Astro-Rail Mode

Beginning with version A-2007.12, both time-average (static) and dynamic power analysis flows in PrimeRail have adopted the PrimeTime PX technology for calculating power consumption of the design.

If you want to continue using Astro-Rail power analysis commands (such as, `poPowerAnalysis`) for backward compatibility reasons, invoke PrimeRail in the Astro-Rail mode:

```
%PrimeRail -AstroRail
```

By invoking PrimeRail in the Astro-Rail mode, the GUI menu will be exactly the same as Astro-Rail; that is, only cell-level static commands will be available. Neither cell-level and transistor-level dynamic analysis commands nor other PrimeRail specific commands and options will be allowed in this mode. An error message will be issued if a PrimeRail specific command is entered.

For a detailed description about using commands in the Astro-Rail mode, please refer to *Astro-Rail User Guide*.

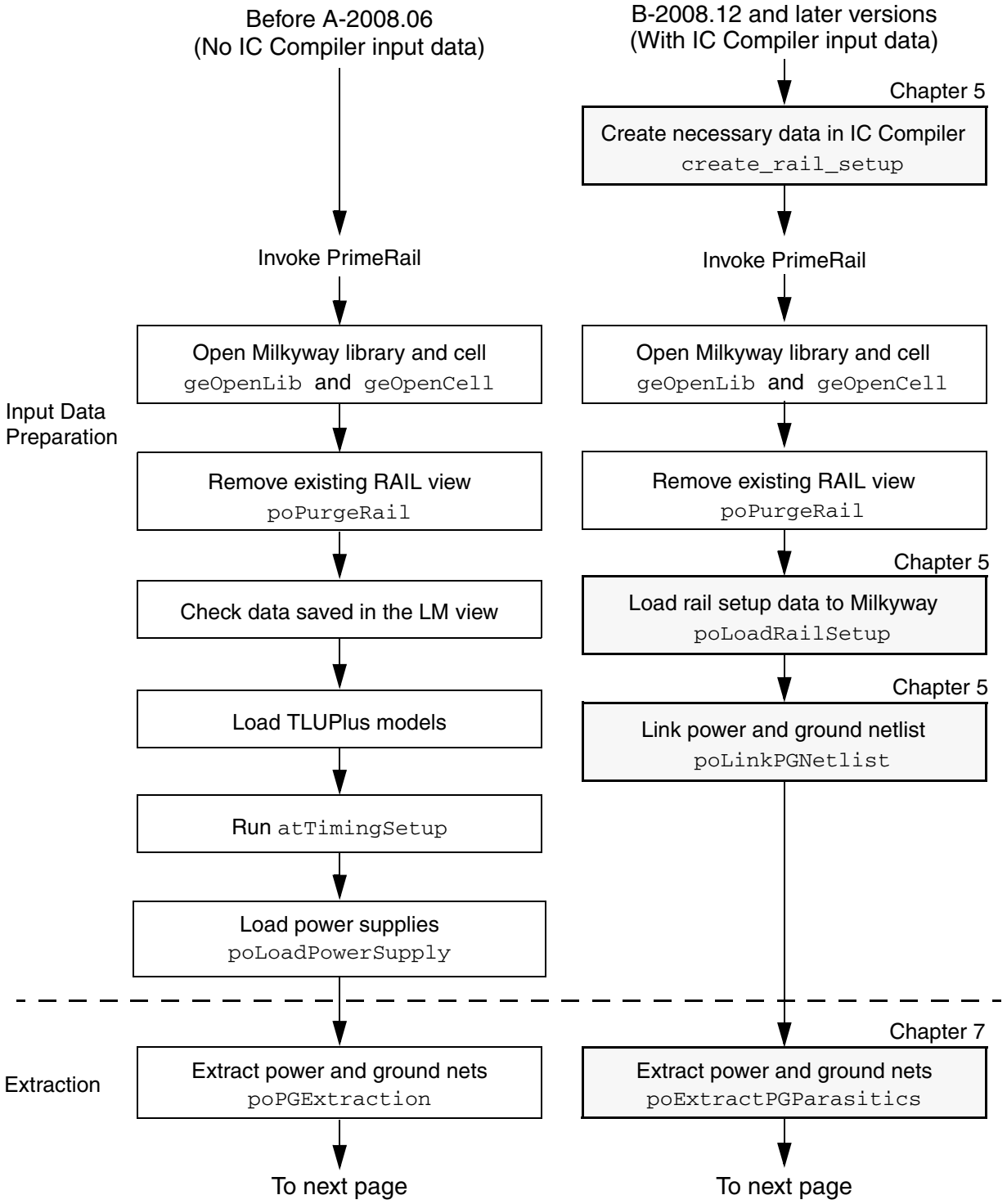
Differences When Using IC Compiler Input Data

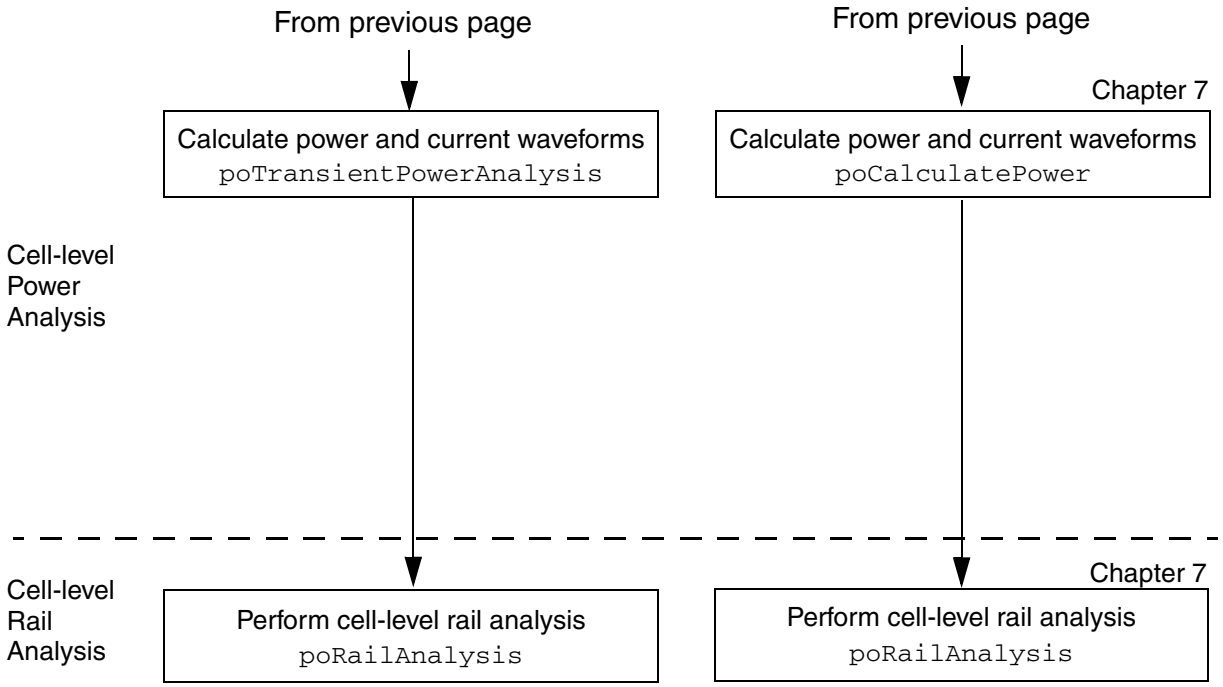
Beginning with version A-2008.06, PrimeRail is capable of reading in the necessary setup information directly from the IC Compiler output files (see “Preparing Rail Setup Data” on page 5-2). This capability has an impact on the steps taken in data preparation, gate-level power analysis, and power and ground net extraction. Some new commands are implemented while others are either removed or changed.

Figure A-1 compares the steps taken in completing a cell-level power and rail analysis with or without using IC Compiler input data.

Note that the information documented in this section is for backward compatibility reasons. Please refer to the related chapters in this user guide if you need a complete introduction of the flow.

Figure A-1 Comparing steps taken in PrimeRail cell-level analysis flow before and after A-2008.06





Data Preparation Without IC Compiler Output Data

This section provides the information on preparing input data for PrimeRail when you do not have input data from IC Compiler.

Note:

This section describes the information that is specific to the flow when you do not have IC Compiler input data. See Chapter 4, “Preparing Library and Other Input Data,” in this user guide if you need more information.

Standard cells and some hard macros are characterized in .lib syntax for timing and power information. The data must exist in the Milkyway LM (logical model) view.

Checking Power Information Stored in LM View

The LM view replaces the TIM and PWR views; it provides the timing and power information necessary for conducting power and rail analysis in PrimeRail. Although the LM view contains the data as defined in your .lib file, it is recommended that you double-check the power information before proceeding to power or rail analysis.

When you use the LM view, you do not need to convert a .lib file to a CLF file. However, you still need to load nontiming CLF commands, not included in the .lib file, into the library, using the `auLoadCLF` command.

If you have not created an LM view in the Milkyway database, use the `gePrepLibs` command to create an LM view.

PrimeRail is capable of accepting the power and ground pin syntax in the Liberty format during power analysis for correctly calculating the power values for multi-VDD and multi-VSS designs.

For more information, see the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide* and the *Library Compiler Technology and Symbol Libraries Reference Manual*.

Loading TLUPlus Models

Run `set_tlu_plus_files` to specify the TLUPlus files to be used during extraction. The PrimeRail extraction engine supports the TLUPlus models for resistance effects on power and ground nets and CONN views. The TLUPlus information stored in the Milkyway database can be converted to an ASCII file, so you can view and edit the file if necessary. A new TLUPlus ASCII file can be used to replace the TLUPlus information stored in the Milkyway database, which can be used by the extraction later.

If there are multiple TLUPlus files attached, the tool will determine the operating condition and the operating temperature by using the settings specified on the Parasitics page of the Timing Setup dialog box (by entering `atTimingSetup` or choosing Design Setup > Timing Setup > Configure Timer and click the Parasitics tab).

If there are multiple operating conditions selected, the tool will use the most pessimistic one (for example, if both nom and min are selected, the tool will choose nom). The operating condition chosen determines which TLUPlus file to be used. If the chosen TLUPlus file contains temperature degradation coefficients, the tool will use the temperature difference between the setting on the Parasitics page (of the `atTimingSetup` command) and the baseline temperature in the TLUPlus file to degrade the resistance results.

Note:

Only TLUPlus models version 2.0 or higher are supported.

When TLUPlus models are available in your design, you will get different extraction results depending on which option—Reuse Old Extraction or RC Extraction—you select in the P/G Extraction dialog box (see Figure A-4 on page A-11). Table describes how the tool performs extraction with or without these two options.

Option		Description
Reuse Old Extraction	**RC Extraction**	
Off	Off	Only parasitic resistance is extracted. The tool uses the Milkyway technology file model and ignores the TLUPlus model if it is present. The previous extraction data is discarded.
On	Off	Only parasitic resistance is extracted. The tool uses the Milkyway technology file model and ignores the TLUPlus model if it is present. The previous extraction result is used. Note that if only some blocks have been extracted in the design, the tool uses the available extraction results and runs extraction on the rest of the blocks. This is the default.
On	On	Both parasitic resistance and capacitance are extracted. However, the tool uses only the resistance data for rail analysis. During extraction, the tool uses the TLUPlus model. If no model is attached to the main library, an error will be issued. The previous extraction result is used. Note that if only some blocks have been extracted in the design, the tool uses the available extraction results and runs extraction on the rest of the blocks.
Off	On	Both parasitic resistance and capacitance are extracted. However, the tool uses only the resistance data for rail analysis. During extraction, the tool uses the TLUPlus model. If no model is attached to the main library, an error will be issued. The previous extraction result is discarded.

Setting Operating Parameters with atTimingSetup

The technology file contains three settings of resistivity coefficients, `unitMinResistance`, `unitNomResistance`, and `unitMaxResistance` to use the `atTimingSetup` command to specify the settings in extracting parasitic data of the power and ground nets on the Parasitics page of the Timing Setup dialog box.

On the Parasitics page you can also specify the temperature at which to perform the extraction. This is used with the `temperatureCoeff` setting in the technology file for each layer in the parasitic extraction.

Loading Power Supplies

When gate-level power analysis is complete, run `poLoadPowerSupply` to load the file where the voltage values of the power nets in the design are specified to the database. To specify these values, create an ASCII file with the `tdfSetPowerSupply` command for each power net.

The syntax of the `tdfSetPowerSupply` command is

```
tdfSetPowerSupply "netName" minVoltage nomVoltage maxVoltage
```

Note:

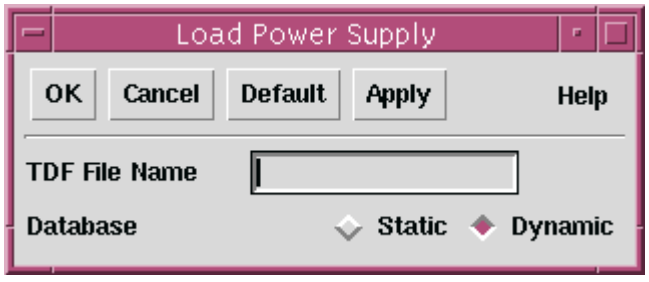
You can write as many TDF commands into one file as necessary to completely specify all the power nets in the design. The units of the voltage values are in volts, regardless of the setting of `unitVoltageName` in the library's technology file.

After this file is created, load it into the database, using the `poLoadPowerSupply` command.

To load power supply information,

1. Enter `poLoadPowerSupply` to open the Load Power Supply dialog box, as shown in Figure A-2.

Figure A-2 Load Power Supply Dialog Box



2. Enter the name of the ASCII file containing the TDF commands.
3. In the Database section, select Dynamic.
4. Click OK or Apply.

PrimeRail reports the values that are successfully loaded in the command window. If any net names specified in the TDF file do not exist or are not power nets, PrimeRail also reports them in the command window.

Cell-Level Analysis Flow without IC Compiler Output Data

Beginning with A-2008.06, PrimeRail uses the synopsys_pr_setup.e file that is generated by IC Compiler to obtain the configuration settings required by power and ground net extraction, power analysis, and rail analysis.

If you do not use a synopsys_pr_setup.e file during data preparation, you must follow the steps documented in this section to complete power and ground net extraction, power analysis, and rail analysis.

Note that this section will include only the steps that are different compared to those when an IC Compiler output synopsys_pr_setup.e file is used. For a detailed description on the flow, refer to “Preparing and Checking Input Data” on page 7-5, in this user guide.

Creating a Template for PrimeTime PX Script File

Performing PrimeTime PX gate-level power analysis requires a PrimeTime PX script file that contains the necessary configuration settings. This script file will be used with `poCallPTPX` to generate the report files that are necessary for running a power analysis.

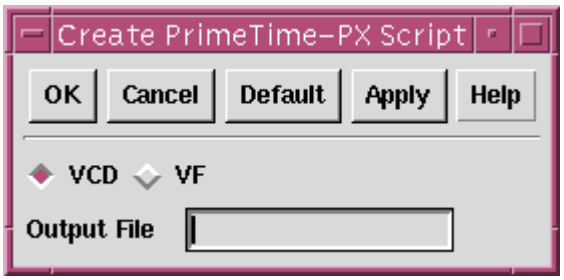
In the flow without the IC Compiler output data, you need to run `poCreatePTPXScriptTemplate` to create a template for the PrimeTime PX script file and use the output script template to create the actual script file. Note that the files listed in the script template are templates only and must be edited for proper file names. After creating and editing the script, you can run PrimeTime PX either externally or by using the `poCallPTPX` command.

To create a template for the PrimeTime PX script file,

1. Enter `poCreatePTPXScriptTemplate`.

   The Create PrimeTime-PX Script dialog box opens, as shown in Figure A-3.

Figure A-3 Create PrimeTime-PX Script Dialog Box



2. Choose to create a template script either for the VCD-based or vector-free (VF) flow.
   - VCD – Creates a PrimeTime PX script file that is used to perform a vector-based power analysis.
   - VF – Creates a PrimeTime PX script file that is used to perform a vector-free (time-average-based) power analysis.
3. Enter the name of the script template to be generated.
4. Click OK or Apply.

A Sample of PrimeTime PX Output Script File Template

The following is an example of the output script file template.

- In VCD mode:

```
set power_enable_analysis true
set link_path <lib1.db lib2.db ...>
read_verilog <netlist1.v netlist2.v ...>
current_design <top_module>
link
read_sdc <timing_constrants.sdc>
read_parasitics <signalnet_parasitics.dspf>
read_vcd -strip_path <path_to_top_module>
   <vcd_file>
set power_rail_output_file <vcd_report_file>
create_power_waveforms

exit
```

- In VF (vector-free) mode:

```
set power_enable_analysis true
set link_path <lib1.db lib2.db ...>
read_verilog <netlist1.v netlist2.v ...>
current_design <top_module>
link
read_sdc <timing_constrants.sdc>
read_parasitics <signalnet_parasitics.dspf>
read_saif <saif_file> -strip_path
```

```
   <path_to_top_module>
set power_rail_output_file <vf_report_file>
update_power
exit
```

If a Switching Activity Interchange Format (SAIF) file is not available for the design being analyzed, use the vcd2saif utility to translate a Value Change Dump (VCD) file to a SAIF file. Alternatively, you can specify the switching activity information interactively or by script by using the `set_switching_activity` command.

If you do not provide any switching activity information, PrimeTime PX uses the default switching activity for the primary inputs and the black box outputs. Starting from these known points, PrimeTime PX runs switching activity propagation by running zero-delay simulation to find out the rest of the unknown switching activities. If only partial switching activity information is given, the zero-delay simulation is used to derive the switching activity for the nets without given switching activity.

When the script file is ready, run `poCallPTPX` to calculate timing and power information of the design. Then run `poTransientPowerAnalysis` to calculate the current waveforms.

Extracting Power and Ground Network Parasitics

In the flow without IC Compiler output data, you need to run `poPGExtraction` for extracting both the resistance and capacitance information of the power supply network in the design.

Additionally, you can choose to use the TLUPlus capacitance model for extraction on the Parasitics page of the Timing Setup dialog box (or choose Design Setup > Timing Setup > Configure Timer and click the Parasitics tab). The command is also able to extract the resistance with resistivity variation and with width and spacing with neighbors and can account for the effect of geometry variation on parasitics.

Note:
Only TLUPlus models version 2.0 or higher are supported.

The `poPGExtraction` command operates on a net basis. This means you have to run extraction on each power and ground net to be analyzed. The extraction result is saved in the Milkyway PARA view where it can be referenced during rail analysis and map display.

poExtractPGParasitics Versus poPGExtraction

Both `poExtractPGParasitics` and `poPGExtraction` extract power and ground net parasitics in PrimeRail. Since version A-2008, the `poExtractPGParasitics` command has been implemented for the IC Compiler–PrimeRail recommended flow. The behavior of `poPGExtraction` remains unchanged.

The following lists the behaviors of `poExtractPGParasitics` that are different from `poPGExtraction`:

- The command always runs on a full-chip design; no hierarchical option is supported. It does not reuse any existing block-level RAIL view, but re-extracts all the blocks with the same given conditions (such as, TLUPlus files, parasitic corners and .db files) for the full-chip design and saves the full-chip extraction result directly into the RAIL view.
- Power and ground net extraction has to be performed after `poLinkPGNetlist`. Otherwise, the tool will return with an error.
- The options, Reuse Old Extraction, Std Cell P/G Modeling, and Include Std Cell Pins, are not supported.
- The command automatically links the power and ground net parasitics and saves the result into RAIL view directly.

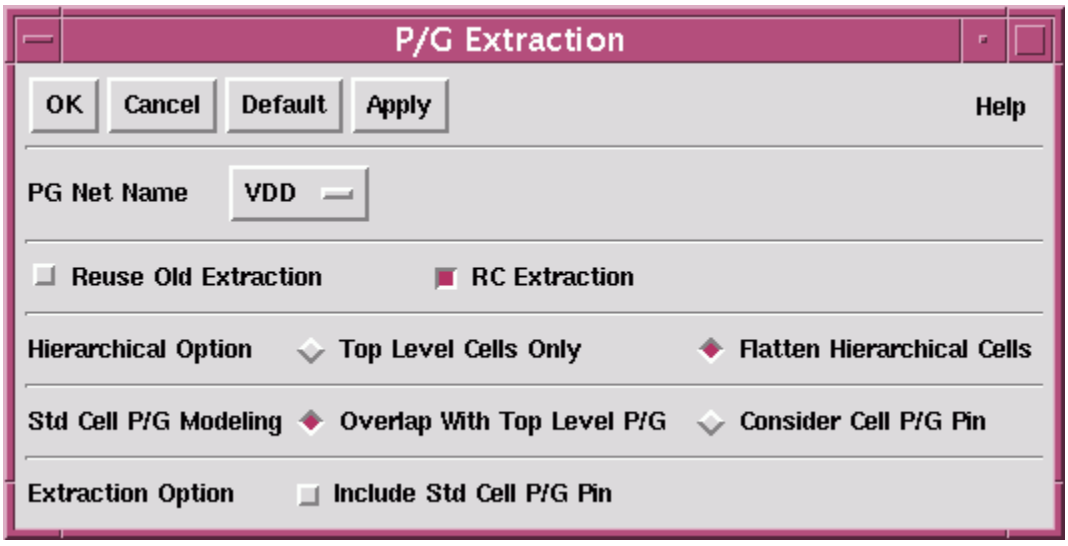
  Therefore, running the `poRailAnalysis` command will not need to read PARA views block-by-block and link them into the full-chip power and ground parasitics for every single rail analysis run, which is time consuming and unnecessary.

To extract power and ground network parasitics,

1. Enter `poPGExtraction`.

   The P/G Extraction dialog box appears.

Figure A-4 The P/G Extraction Dialog Box



2. Select the name of the net to be extracted.
3. Select Reuse Old Extraction if you want to use the previously extracted data from this process.
4. Select RC Extraction to extract both the resistance and capacitance information of the design.

5. In the Hierarchical Option section, select Flatten Hierarchical Cells to extract both the top-level cell as well as the child cells used. Select Top Level Cells Only if child cells should not be extracted.
6. In the Std Cell P/G Modeling section, make a selection.
   - When Consider Cell P/G Pin is selected, the detailed power and ground net geometry of standard cells will be extracted including small fingers.

     Selecting Consider Cell P/G Pin is recommended for extraction including power and ground net capacitance, because the fingers can have a noticeable impact on the total wire capacitance of the power and ground network.
   - When Overlap With Top Level P/G is selected, PrimeRail places the node of current consumption of each standard cell in the middle of the overlap between the pin in the standard cell and the top-level routing. Finger geometry will be ignored.
7. In the Extraction Option section, select Include Std Cell P/G Pin to consider the detailed location of transistors inside standard cells when determining the location to place the current waveform for a given cell. When deselected, the current waveform is placed at the center of the power and ground rail of the cell.

   You selected the Std Cell P/G Modeling option in step 6, when all the small finger geometry become part of the extraction network, but the current source location remained the same.

   The Include Cell P/G Pin option is not available when the Consider Cell PG Pin option is selected.

   Selecting this option is not recommended, because it does not provide a significant benefit in accuracy, and it consumes a higher runtime.
8. Click OK or Apply.

B

The Nature of Power Consumption Data

On a conceptual level, the power consumed by a cell instance in a design can be divided into two types:

1. Static power dissipation
2. Dynamic power dissipation

The first occurs constantly when the cell is powered—the chip is turned on, and the power net the cell is tied to is active. The second occurs when a signal that leads into the cell and comes out of the cell changes its voltage state.

From the computational standpoint in PrimeRail, the power consumption can be broken down into different components, and PrimeRail calculates these components with different methods, as described in the following sections:

- Switching Power
- Short-Circuit Power
- Internal Power
- Leakage Power

Switching Power

For switching power, PrimeRail calculates the instance-based power consumption using the power supply voltage and net switching information you specify, combined with the signal net loading capacitance to which the cell is driving. The power supply voltage and net switching values are defined with Scheme commands, as described in Appendix C, "Commands for Specifying Hard Macro Power."

The following is the formula PrimeRail uses to calculate the switching power consumption.

$$P = \frac{1}{2} C \times V^2 \times f$$

The factor 0.5 is assigned because the switching information is specified in PrimeRail as the number of toggles per unit of time, rather than the number of cycles per unit of time.

For the other three power types (short-circuit power, internal power, and leakage power), PrimeRail determines the power contributions from the PWR views in the database. These PWR views describe the power consumption of the cell as a function of the input net transition time and the output net loading capacitance. Sometimes, certain contributions can be constants.

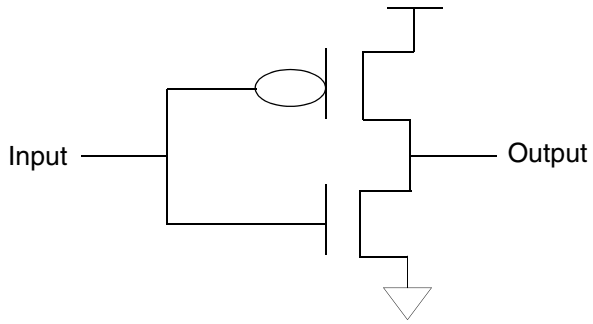
Depending on the environment of the cell instance, the data is represented in a table lookup methodology. You can define the input net transition times by using `definePortInstanceTransition` or those calculated with the Astro-based timing engine. PrimeRail reads the output net loading capacitance values from a StarRC-generated PARA view of the cell or obtains the values that are generated by use of the layout parasitic engine (LPE). See the *StarRC User Guide* for a detailed description of how to generate the output net loading capacitance values.

Short-Circuit Power

When a cell is changing its state from 0 to 1 (or vice versa), both n- and p-transistors are on for a short period of time. This allows a direct connection through the devices between the power and ground rails of the cell.

Figure B-1 is a simple example showing the transistors that are involved in an inverter cell.

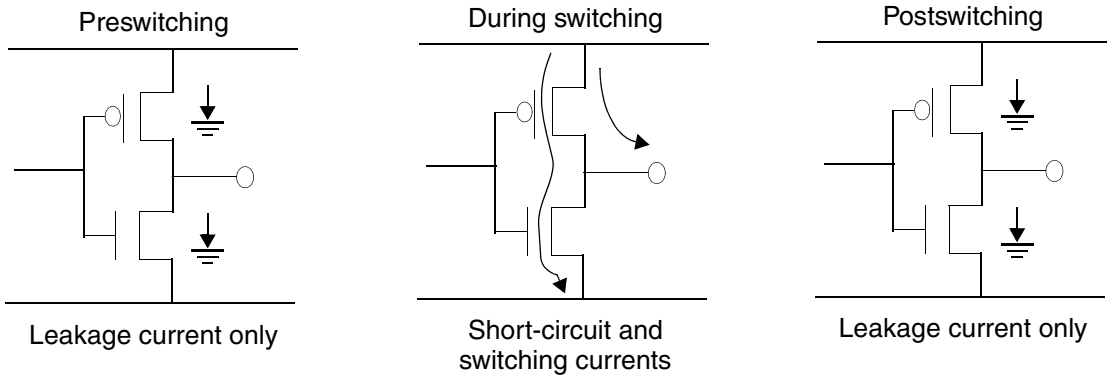
Figure B-1 Transistors in an Inverter Cell



In this example, the amount of current that flows directly from VDD to GND depends on the speed at which the input signal changes. This means that the faster an input signal changes, the less current can flow and the smaller the short-circuit power consumption.

Figure B-2 demonstrates the types of power involved in an inverter cell, showing the power flow before, during, and after a switching event. For more-complicated gates, the internal power contribution can also be involved.

Figure B-2 Power Distributions Involved in an Inverter Cell

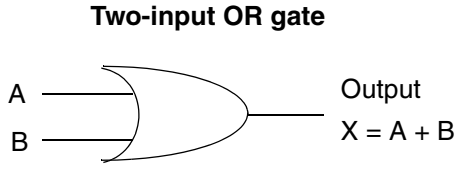


Internal Power

Not every change in the state of an input pin of a cell necessarily leads to a change in state of output pins. The power consumed by a cell when an input pin state change does not affect any of the output pin states classified as internal power consumption. Internal power can be substantial in complex gates because many individual transistors might be controlled by a single input signal.

For example, consider the case of the OR logic cell shown in Figure B-3.

Figure B-3 A Sample OR Logic Cell



If both A and B are at 1 and B changes to 0, the output stays at 1. However, the state of the transistors that B controls changes and thus, they consume power.

Leakage Power

The power consumed through the subthreshold currents and by reverse-biased diodes are considered leakage power in PrimeRail. The subthreshold currents constantly flow from source to drain in a transistor and occur even when the gate-to-source voltage of the device is below the threshold voltage for the transistor to be fully on. The power is consumed by the reverse-biased diodes that are formed between the diffusion regions of the transistors and the substrate.

The leakage power for a cell typically remains constant. It does not change when an input transition signal or the amount of the output load capacitance varies from instance to instance. However, because leakage power is a constant drain of current, it can be an important factor in calculating the overall power consumption, particularly for smaller process technologies.

C

Commands for Specifying Hard Macro Power

You might want to specify additional power values to carry out meaningful rail analysis in the following situations:

- You have already performed power analysis with other tools and only want to load the results into PrimeRail for rail analysis.
- Power models might not be available for some macro cells, such as memory cells. If this is the case, the power values of those cells need to be specified to account for their power consumption.

You can define your power consumption in several ways, as described in these sections:

- Cell-Instance-Based
- Port-Based

Cell-Instance-Based

Use the `defineCellInstancePower` command to specify a power consumption value for specific cell instances by instance name.

Syntax

`defineCellInstancePower` *`cellId instName powerValue`*

Argument	Description
`cellId`	ID of the top cell for which you are setting the power value. Valid values: ID of any open cell in the library. If the cell is currently being edited, you can substitute cellId with `geGetEditCell`.
`instName`	Name of a cell instance in which you are setting a power value. Valid values: Name of any cell instance in the library, or use pattern matching (.*) to set the power to cell instances that match the pattern.
`powerValue`	Power, in the unit you specify, is assigned to the cell instances. The unit should be defined in the header section of the technology file. For example, if unitPowerName = mW, the powerValue should be expressed in milliwatts (mW). Valid values: any real, nonnegative, floating point number.

Note:

The power value defined by `defineCellInstancePower` represents the total power consumption of a cell instance.

Port-Based

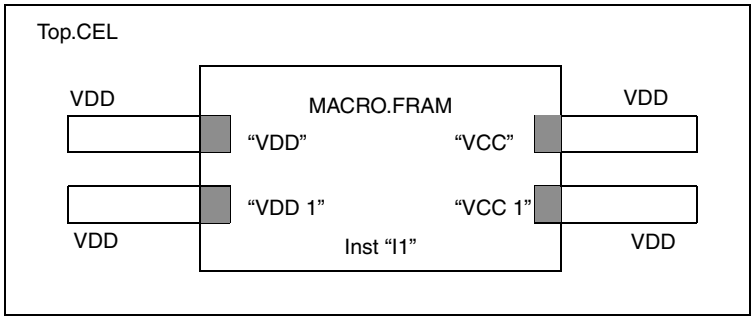
Use the `defineCellInstancePowerByPort` command to assign different power values to two or more child cell ports that are attached to the same parent power net. This is in contrast to the results obtained when you use the `defineCellInstancePower` command described previously, in which the two different power ports in the macro in question are given exactly the same power consumption value.

Syntax

`defineCellInstancePowerByPort` *`cellId cellInstName portName powerValue`*

Argument	Description
`cell ID`	ID of the top cell for which you want to set the port instance transition time. Valid values: ID of any opened cell in the library. If the cell is currently being edited, you can replace cellID with `geGetEditCell`.
`cellInstName`	Name of the cell instance for which you are setting the power values. Valid values: Name of any cell instances in the library, or use pattern matching (for example, ".*") to represent all the cell instances that match the pattern.
`portName`	Name of the port to which the cell instance connects to the power and ground bus. Valid values: Name of any power and ground port in the library, or use pattern matching (".*") to set the power value to the cell instances. The cell instance can have different power values at different power and ground ports.
`powerValue`	Power, in the unit you define, assigned to the cell instances. The unit should be defined in the header section of the technology file. For example, if unitPowerName = mW, then powerValue should be expressed in milliwatts (mW). Valid values: any real, nonnegative number.

The following figure shows a situation in which this command would prove useful.



This might apply to either a black box situation or a situation where the CONN view and CSF data are present for the macro in question. Assume that you assign a unique value of power consumption to the two ports, VDD and VCC, in this MACRO.FRAM block. The commands might be (using arbitrary numbers):

```
defineCellInstancePowerByPort (geGetEditCell) "I1" "VDD"\
 5.0
defineCellInstancePowerByPort (geGetEditCell) "I1" "VCC"\
 2.0
```

Assume there are five pins on the VDD port and two pins on the VCC port, as in the previous example.

Black Box Macros

In a black box situation, there are only pins for each port in the design. There are no internal details about the block.

The power value assigned to each logical port is now divided evenly among the pins of the appropriate port. This does not apply different values to pins that have different sizes. If there is a large difference in pin size, which can conceivably result in different numbers of top-level routes attaching to the pin, the power consumption difference on a pin-by-pin basis could result in incorrect rail analysis results. PrimeRail programmers are currently investigating scaling the power value assigned to a pin based on its size.

The final results, based on the example assumptions, are

- The VDD port's 5 milliwatts of power consumption is divided evenly among its five pins. Each VDD pin consumes 1 milliwatt of the top-level net.
- The VCC port's 2 milliwatts of power consumption is divided evenly among its 2 pins. Each VCC pin consumes 1 milliwatt of the top-level net.

Hard Macros With CONN View and CSF Data

If the hard macro has a CONN view with CSF data, the power value assigned to each port is used to scale the CSF entries associated with the same port.

For example, assume that the previous situation holds, except now the VDD port with its five pins, is connected to a current source file with 1,000 entries. The VCC port, with its two pins, is connected to a current source file with 100 entries. In this event, the following results occur:

- The VDD CSF of 1,000 entries has each value scaled so they add up to 5 milliwatts of power.
  - If they are unit value entries (due to the use of the `poGenCurrSource` command), each entry has a value of 0.005 milliwatt.
  - If they are representative of the transistor-level saturation current flow based on the XTR view, each entry might have a different value. They are all scaled relative to each other so they add up to the 5-milliwatt value.
- The VCC CSF of 100 entries has each value scaled so they add up to 2 milliwatt of power.

- If they are unit value entries (due to the use of the `poGenCurrSource` command), each entry has a value of 0.02 milliwatt.
- If they are representative of the transistor-level saturation current flow based on the XTR view, each entry might have a different value. They are all scaled relative to each other so they add up to the 2-milliwatt value.

Glossary

CONN View
: Represents the physical information of power and ground network inside the GDSII or hard macros. For memory cells and complex I/O cells, CONN views need to be created if the FRAM view does not contain detailed PG shapes. If the power consumed by I/O cells is negligible, it is recommended not to create CONN views for I/O cells. Use `poConnViewPreparation` to generate CONN views and current source files. Also, skipping bit cells is recommended if the cell being analyzed is a memory cell.

DWM
: Dynamic white box model. A Milkyway data structure stored in the RAIL view. In PrimeRail, you generate a DWM for a hard macro and use the model during full-chip analysis with reusability.

Electromigration
: The movement of metal molecules due to high current density, which leads to metal opens and shorts in the chip. High electromigration greatly reduces reliability and chip life span.

Hard macro
: A block that is generated in a methodology other than place and route and is brought into the Milkyway database as a GDSII file by use of the `auStreamIn` command. In PrimeRail, a hard macro has to be from a GDSII source.

Library characterization
: Characterizing intrinsic parasitics and current waveforms of the gate-level cells in addition to the established library characterization data. Characterizing parasitics of power management cells and leakage currents of filler cells is also supported.

Macro model
: Created to improve performance or conceal detailed intellectual property (IP) vendor data.

Map

The graphical view of analysis results, including current violation, IR drop violation, power distribution, and parasitic data. Different colors can be assigned to help visualize problem areas, like indicating various step levels of voltage rise and drop or current density limits, for example. Visibility control over layout layers and design hierarchy levels is also supported.

Multithreshold-CMOS circuit

MT-CMOS circuit. It contains high and low threshold (Vt) transistors which can be used to reduce leakage power in low power and high performance applications. High Vt CMOS transistors are used to disconnect the power grid to the core at the stand-by state, resulting in a very low leakage current as set by the high threshold voltage of the series transistor.

Pin

The input and output of cells within a design (such as gates and flip-flops). The ports of a subdesign are pins within the parent design.

Port

The inputs and outputs of a design. Port direction is designated as input, output. or bidirectional.

A logical connection from one block (the child) to the parent block in which this child is placed. For example, there might be one power port named VDD and one named VCC.

Power and ground specification file (pg.spec)

An ASCII file that contains the power and ground port names and how the ground to power ports are paired. The pg.spec file is needed during library characterization. The supply value must be consistent with the actual value in the .lib file. During characterization, the tool first crosschecks the supply information in the pg.spec file with the “nom_voltage” value in the linked LM view or the .db file for the specified corner.

Follow these recommendations when using a pg.spec file:

– During characterization, create a pg.spec file for the reference library.

– For multiple PVT corners, separate pg.spec files you need to represent the particular PVT corner. Then run the `pgLibCharacterize` command for each of the PVT corners using the corresponding pg.spec files.

Power management cell

PM cell. Power management cells act as switch cells. When turned on, they allows currents to flow through; when turned off, they are open circuits. The power management cells are used to reduce leakage currents in a design. A multithreshold-CMOS cell is an example of a power management cell.

PrimeRail Rail Setup File (synopsys_pr_setup.e)
A binary file that is created by the IC Compiler `create_rail_data` command. The file contains all the necessary settings required by PrimeRail, including verilog, TLUPlus file, operation corner, and so on.

Rush current
A momentary input current surge, measured during the initial turn-on of the power supply. Measures need to be taken for rush current; otherwise the uncontrolled rush current can often damage other components of the circuit, lower the available supply voltage to other components, and finally cause system failures.

SDC file
A design constraints file that is used for PrimeTime sign-off and for the PrimeTime PX run.

Signal net parasitic
The SPEF or DSPF signal net parasitic used for PrimeTime PX run. Make sure this is consistent with Milkyway database, which means the signal parasitic is generated from the same Milkyway design that can be analyzed in PrimeRail.

Soft macro
In PrimeRail, any cell that has been placed and routed in a placement and routing tool such as Astro. It is editable and can contain standard cells, hard macros, or other soft macros.

Subblock
A block that is under the top cell.

SPICE netlist
Contains .SUBCKT information for each of the leaf cells. The file is used for library characterization purpose. The .SUBCKT description should also have power and ground ports defined (sometimes, they are defined globally).

SPICE model
Contains the .model information for devices or transistors that are used in the SPICE netlist for each of the leaf cells.

TLUPlus model
A Table Look Up Plus (TLUPlus) based model for resistance and capacitance calculation. You need a StarRC Interconnect Technology Format (ITF) file and a layer mapping file for creating the TLUPlus model. Then run `set_tlu_plus_files` to load the TLUPlus files into the Milkyway library.

User-defined elements file
The file that contains information of user-defined resistance, capacitance, and inductance elements and controlled current sources to be used in the full-chip simulation.

User-defined tap file
The file used in rail analysis for defining nonideal voltage sources.

VCD file
A Value Change Dump (VCD) file that is used for vector-based analysis. Make sure the delay information described in the file is not zero.

Verilog netlist
A post-layout Verilog netlist used for the PrimeTime PX run. Make sure this netlist is consistent with the Milkyway database (or you can choose to dump it out from the Milkyway database).

Voltage drop
Voltage drop or IR drop is the reduction of voltage due to the currents passing through the resistance network. The impact occurs at points farther from the pad. Lower voltage supply to cells can lead to longer ramp times and changes to the switching threshold. Timing can be incorrect, causing poor performance.

Waveform
The shape and form of a signal, such as a wave moving across the surface of water or the vibration of a plucked string.

Wake-up time
Time the virtual power or ground net takes to reach the value of the actual power or ground net.

What-if analysis
A way to optimize voltage drop and electromigration effects by inserting virtual capacitors or resistors in the power and ground network over the voltage drop map or the resistivity map.

Index

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X