PrimeTime® PX
User Guide

Version D-2010.06, June 2010

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, HAPS, 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, 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

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 *PrimeTime PX Release Notes* in SolvNet.

To see the *PrimeTime PX Release Notes*,

1. Go to the release notes page on SolvNet located at the following address:

   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. Click PrimeTime Suite, then click the release you want in the list that appears at the bottom.

About This Guide

This manual describes the PrimeTime PX tool, its methodology, and its use. PrimeTime *PX* is a static and dynamic full-chip power analysis tool for complex multimillion-gate designs, intended for use within the PrimeTime environment. Its high-capacity power analysis includes gate-level average and peak power verification. PrimeTime PX supports industry-standard synthesis libraries and contains a powerful and flexible methodology that is fully integrated with existing design flows. It provides a high degree of accuracy, performance, ease of use, and comprehensive power diagnostics.

Audience

PrimeTime PX is intended for designers of high-performance ASIC and structured custom ICs who need accurate power dissipation data for cell-based designs.

Related Publications

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

https://solvnet.synopsys.com/DocsOnWeb

You might also want to refer to the documentation for the following related Synopsys products:

- PrimeTime
- Power Compiler

Conventions

The following conventions are 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, go to the SolvNet Web page at the following address:

https://solvnet.synopsys.com

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), 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

Overview

This chapter describes the PrimeTime PX tool and its features. It contains the following sections:

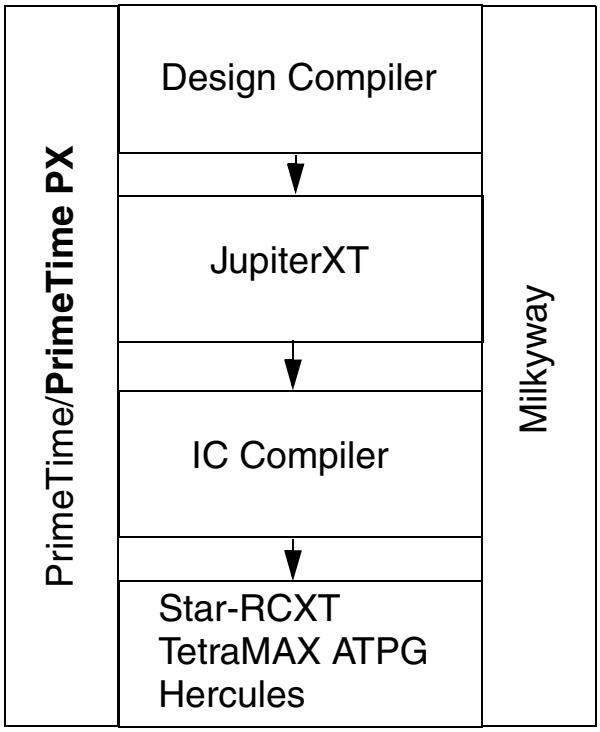
- Introduction to PrimeTime PX
- Power Modeling

Introduction to PrimeTime PX

PrimeTime PX is an add-on feature to PrimeTime that accurately analyzes power dissipation of cell-based designs. It is intended as an advanced solution for ASIC and structured custom circuit designers who are developing products for power-critical applications such as portable computing and telecommunications.

Figure 1-1 shows how PrimeTime PX fits into the Synopsys low-power design methodology.

Figure 1-1 PrimeTime PX Full-Chip Power Analysis



PrimeTime PX builds a detailed power profile of the design based on the circuit connectivity, the switching activity, the net capacitance, and the cell-level power behavior data in the Synopsys database format (.db) library, which can be either a nonlinear power model (NLPM) or a Composite Current Source (CCS) library. It then calculates the power behavior for a circuit at the cell level and reports the power consumption at the chip, block, and cell levels.

You can use the following power analysis techniques with PrimeTime PX:

- Averaged power analysis

For purely averaged power analysis, PrimeTime PX supports propagation of switching activity based on defaults, user-defined switching, or switching derived from an HDL simulation (either RTL or gate level).

- Time-based power analysis

  For extremely accurate analysis of power with respect to time, PrimeTime PX supports analysis based on the RTL or gate-level simulation activity over time.

  PrimeTime PX uses an event-driven algorithm to calculate the power consumption for each event. Detailed time-based power waveforms are generated to provide both average and peak power results. The tool can produce an average and peak power report.

Power Modeling

The power dissipated by a circuit falls into two broad categories, as described in this section.

- Leakage power
- Dynamic power, which includes internal power and switching power

For power analysis, your Synopsys library must contain power models for all of the cells. These power models (NLPM or CCS) contain tables that PrimeTime PX uses to calculate leakage power and internal power.

Switching power results from calculations based on the voltage, netlist capacitance, and switching of the nets.

PrimeTime PX uses leakage and dynamic power calculations to return results for peak power analysis and average power analysis.

Leakage Power

Leakage power is the power dissipated by a cell when it is not switching—that is, when it is inactive or static.

Intrinsic Leakage Power

Leakage power is dissipated in several ways. Most of the leakage power dissipation results from source-to-drain sub-threshold leakage, which is caused by reduced threshold voltages that prevent the gate from completely turning off. Leakage power is also dissipated when current leaks between the diffusion layers and the substrate. The leakage power is state and voltage dependent. These types of leakage power are referred to as intrinsic leakage.

Gate Leakage Power

Gate leakage power is the leakage power from the source to the gate or the gate to the drain. Gate leakage power contributes significantly to the overall leakage power of the design, as the dimension of the process technology diminishes. Gate leakage power depends strongly on the gate oxide thickness and the supply voltage and shows very little sensitivity to temperature. PrimeTime PX supports the analysis of the gate leakage power. The tool accesses the tables in the technology library to determine the leakage power for the cells in the design. When the technology library is characterized for sub-threshold leakage and gate leakage, PrimeTime PX can report these leakage components for cells. For more information, see Chapter 3, “Power Analysis Modes in PrimeTime PX”.

Dynamic Power

Dynamic power is the power dissipated when the circuit is active. A circuit is active anytime the voltage on a net changes due to some stimulus applied to the circuit. Because voltage on an input net can change without necessarily resulting in a logic transition on the output, dynamic power can be dissipated even when an output net does not change its logic state.

The dynamic power of a circuit is composed of two kinds of power:

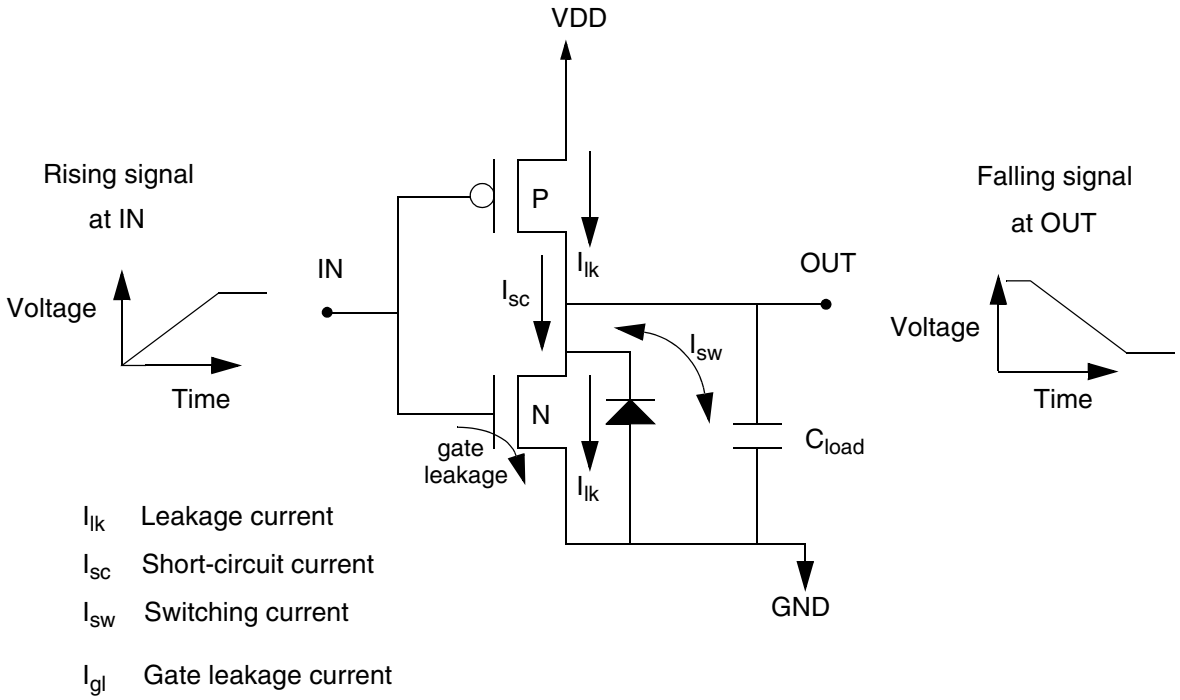
- Internal power
- Switching power

Internal Power

Internal power is any power dissipated within the boundary of a cell. During switching, a circuit dissipates internal power by charging or discharging any existing capacitances internal to the cell. Internal power includes power dissipated by a momentary short circuit between the P and N transistors of a gate, called short-circuit power.

To illustrate the cause of short-circuit power, consider the simple gate shown in Figure 1-2. A rising signal is applied at IN. As the signal transitions from low to high, the N-type transistor turns on and the P-type transistor turns off. However, for a short time during signal transition, both the P- and N-type transistors can be on simultaneously. During this time, current I_{sc} flows from VDD to GND, causing the dissipation of short-circuit power (P_{sc}).

Figure 1-2 Components of Power Dissipation



I_{lk} Leakage current

I_{sc} Short-circuit current

I_{sw} Switching current

I_{gl} Gate leakage current

For circuits with fast transition times, short-circuit power can be low. However, for circuits with slow transition times, short-circuit power can account for more than 50 percent of the total power dissipated by the gate. Short-circuit power is affected by the dimensions of the transistors and the load capacitance at the gate's output.

In most simple library cells, the input transition time and the load internal power are due mostly to short-circuit power. For more complex cells, the charging and discharging of internal capacitance can be the dominant source of internal power.

Library developers can model internal power by using the internal power library group. For more information on modeling internal power, see the Library Compiler reference manuals.

Switching Power

The switching power of a driving cell is the power dissipated by the charging and discharging of the load capacitance at the output of the cell. The total load capacitance at the output of a driving cell is the sum of the net and gate capacitances on the driving output.

Because such charging and discharging are the result of the logic transitions at the output of the cell, switching power increases as logic transitions increase. Therefore the switching power of a cell is a function of both the total load capacitance at the cell output and the rate of logic transitions.

Generating Power Models

PrimeTime PX provides an automated mechanism to store power data in a model. The power model generated by the tool can be instantiated for a faster chip-level power analysis and is based on the Extracted Timing Model (ETM) feature of PrimeTime. The power model is generated by incorporating power information into the ETM. So, you have to perform timing analysis and power analysis on your design prior to generating the power model. Based on the result of the analysis, you can use the `extract_model -power` command to generate the power model. The `extract_model -power` command can generate models that contain both timing and power data `abcd`.

When generating the power model from the gate-level design, PrimeTime PX associates the dynamic power of the gate-level design with the clock pins of the generated power model. In other words, the dynamic power of the gate-level design translates into the internal power of the generated power model. The internal power of the model can change based on the clock frequency at which the power model is instantiated. To get similar results in terms of power from the generated power model and the gate-level design, the power model must be instantiated at the same clock frequency as the clock that powers the power model.

The following are the limitations of PrimeTime PX in generating the power model:

- Multiple modes of operation are not supported in the generated power model.
- Block scope checking for power is not supported.
- When the `extract_model -power` command is used, .lib is the only format that is supported to write the power model. To generate the .db equivalent of the model, use the Library Compiler or Design Compiler, and compile the .lib file. For more information about this command see the *PrimeTime Modeling User Guide*, and the `extract_model` command man page.

2

PrimeTime PX Tutorials

PrimeTime PX supports two modes of power analysis, the averaged and the time-based power analysis mode. The tool installation directory contains tutorials for both modes of power analysis. To use the tutorial, copy the tutorial file to your own directory and follow the instructions in this chapter. The design used in the tutorials consists of a multiplier, an adder, and logic to connect them. The example design, the activity data files, the PrimeTime PX scripts, and the steps you complete to run the examples are described in the following sections:

- Averaged Power Analysis Mode Tutorial
- Time-Based Power Analysis Mode Tutorial

Averaged Power Analysis Mode Tutorial

The *install_dir*/doc/pt/tutpx/averaged contains the examples for averaged power analysis. The averaged power analysis mode is selected when you set the `power_analysis_mode` variable to `averaged`. This variable must be set before specifying any power analysis command.

The following sections describe the various files used in the tutorial, the commands used in the PrimeTime PX scripts, how you run the examples in the tool and how to view the generated report.

Related Files

In the tutorial directory, the files listed in Table 2-1 are related to averaged power analysis.

Table 2-1 Files for the Averaged Power Analysis Tutorial

File name	Description
`./sim/vcd.dump.gz`	VCD file
`./src/hdl/gate/mac.vg`	Gate-level netlist of the design
`./src/lib/snps/core_typ.db`	Technology library file
`./src/hdl/gate/mac.sdc`	Synopsys Design Constraints (SDC) file
`./src/annotate/mac.spef.gz`	Parasitic file
`./src/annotate/mac.spef.gz`	VCD file
`./sim/mac.saif`	SAIF file with switching activity
`./averaged/ave_saif.tcl`	Averaged power analysis using switching activity from the SAIF file
`./averaged/ave_vcd.tcl`	Averaged power analysis using switching activity from the VCD file
`./averaged/ave_vf.tcl`	Averaged power analysis using the default switching activity

PrimeTime PX Script File

The tutorial contains three sample Tcl scripts for the averaged power analysis mode. The commands in the script are grouped into different sections, which represent the basic steps of power analysis. These steps which are common for all types of power analysis, are as follows:

1. Set the power analysis mode
2. Read and link the design
3. Set input transition and annotate parasitics
4. Read the switching activity file
5. Perform power analysis

Example 2-1 PrimeTime PX Script in the Tutorial For Averaged Power Analysis

```
#######################################################
#     Set the Power Analysis Mode
#######################################################
set power_enable_analysis TRUE
set power_analysis_mode averaged
#######################################################
#     Read and link the Gate Level Netlist
#######################################################
set search_path  "../src/hdl/gate ../src/lib/snps . "
set link_library " * core_typ.db"
read_verilog   mac.vg
current_design mac
link
#######################################################
# Read SDC and set transition time or annotate parasitics
#######################################################
read_sdc   ../src/hdl/gate/mac.sdc
set_disable_timing [get_lib_pins ssc_core_typ/*/G]
read_parasitics ../src/annotate/mac.spef.gz
#######################################################
#       check, update, or report the timing
#######################################################
check_timing
update_timing
report_timing
#######################################################
#       read switching activity file
#######################################################
read_vcd -strip_path tb/macinst ../sim/vcd.dump.gz
report_switching_activity -list_not_annotated
#######################################################
#       check or update or report power
#######################################################
check_power
update_power
report_power -hierarchy
quit
```

For more information about each command, review the remaining sections of this chapter or check the man pages.

Gate-Level Netlist

PrimeTime PX supports a gate-level netlist only. The file mag.vg is a gate-level verilog netlist. This netlist contains leaf-level cells that are the instantiation of the library cells. The valid formats are Verilog, VHDL, EDIF, db, .ddc, or Milkyway. Verilog is used for this tutorial. The netlist can be either flat or hierarchical.

Technology Library

The technology library file contains library cells. Each cell has timing, power, and characterization information. Internal power and leakage power are in the library.

SDC File

The SDC file contains the design constraints. The driver cell information is used to calculate the transition time on the primary inputs.

Parasitic File

The parasitic file contains the capacitance of the nets. Capacitance is one of the factors in determining the dynamic power. You can unzip and view the file.

Switching Activity

In the averaged power analysis, you use either SAIF or VCD file formats to read the switching activity.

A SAIF file is generated either from gate-level or RTL simulation. RTL SAIF captures switching activity for only part of the design. PrimeTime PX propagates the partial switching activity throughout the whole design.

You can also use the VCD file to specify the switching activity information. If you do not specify switching activity information, the tool assumes certain default values for the switching activity.

Steps for Analyzing Power

For this tutorial, the working directory is **./tutpx/averaged**. The search path is set based on that directory. It should be your current directory. Before running the tutorial, you need to verify that PrimeTime PX has been installed.

Changing Your Working Directory

Change your current directory to ./tutpx/averaged. The search path setting is based on this directory.

Running PrimeTime PX

You can run any of the sample PrimeTime PX scripts, ave_saif.tcl, ave_vcd.tcl or ave_vf.tcl for the averaged power analysis flow. For instance, to run the ave_saif.tcl script, enter the following command:

```
%> pt_shell -f ave_saif.tcl
```

PrimeTime PX runs in batch mode. The tool stops when all the commands in the Tcl script have been executed.

Viewing the Power Report

In the averaged power analysis mode, using the SAIF file format for activity information, the power report generated by the `report_power` command is as shown in the following example:

```
****************************************
Report : Averaged Power
         -hierarchy
Design : mac
Version: B-2008.12
Date   : Tue Nov 18 01:28:23 2008
****************************************


                                    Switch   Int      Leak      Total
Hierarchy                           Power    Power    Power     Power    %
------------------------------------------------------------------------------
mac                                 1.55e-03 2.10e-03 2.59e-07  3.65e-03 100.0
 mult_21 (mac_DW02_mult_16_16_0)    7.28e-04 5.54e-04 1.49e-07  1.28e-03  35.2
   U1/U9720 (mac_DW01_add_25_0)     2.12e-04 1.27e-04 1.60e-08  3.39e-04   9.3
 add_23 (mac_DW01_add_33_0)         3.31e-04 2.40e-04 2.36e-08  5.72e-04  15.7
1
```

Vector-Free Power Analysis

When you select the averaged power analysis mode and do not specify any activity information, the tool performs vector-free power analysis. In this mode, PrimeTime PX applies the default toggle rate on the primary inputs and black box outputs and then propagates them. If required, you can change the default toggle rate. This usage model is useful for quick power estimation.

Related Files

In the tutorial directory, the files listed in Table 2-2 are related to the vector-free analysis tutorial.

Table 2-2 Files for the Vector-Free Tutorial

File name	Description
`./averaged/ave_vf.tcl`	PrimeTime PX script for vector-free analysis
`./src/hdl/gate/mac.vg`	Gate-level netlist of the design
`./src/lib/snps/core_typ.db`	Technology library file
`./src/hdl/gate/mac.sdc`	Synopsys Design Constraints (SDC) file
`./src/annotate/mac.spef.gz`	Parasitic file

PrimeTime PX Script

This is a Tcl script. You specify the search path, the link and the target library and the PrimeTime PX variables and commands for power analysis, in this script.

Example 2-2 PrimeTime PX Script in the Tutorial for Vector Free Power Analysis

```
     ########################################################
Link the Design
########################################################
     set search_path         "../src/hdl/gate ../src/lib/snps . "
set link_library    " * core_typ.db"

set power_enable_analysis true
set power_analysis_mode averaged
set read_verilog mac.vg
current_design      mac
link
########################################################
#       set transition time / annotate parasitics
########################################################
read_sdc             ../src/hdl/gate/mac.sdc
read_parasitics      ../src/annotate/mac.spef.gz
########################################################
#       power analysis
########################################################
check_timing
update_timing
report_timing
update_power
report_power -hierarchy
```

Steps for Analyzing Power

The current working directory should be **./tutpx/averaged**.

Running PrimeTime PX

To run the ave_vf.tcl script in PrimeTime PX, enter the following command:

```
%>pt_shell -f ave_vf.tcl
```

PrimeTime PX runs in batch mode and stops when all the commands in the script have been executed.

Viewing the Power Report

The power report from vector-free power analysis has the same format as the SAIF-based report. The report shows only averaged power but not peak power.

Time-Based Power Analysis Mode Tutorial

The *install_dir/doc/pt/tutpx/time_based* contains the examples for time-based power analysis. The time-based power analysis mode is selected when you set the `power_analysis_mode` variable to `time_based`. This variable must be set before specifying any power analysis command.

The following sections describe the various files used in the tutorial, the commands used in the PrimeTime PX scripts, how you run the examples in the tool and how to view the generated report.

Related Files

In the tutorial directory, the files listed in Table 2-3 are related to the time-based power analysis tutorial.

Table 2-3 Files for the Time-Based Power Analysis Tutorial

File name	Description
`./time_based/tim_gatevcd.tcl`	Script for gate-level time-based power analysis
`./time_based/tim_rtlvcd.tcl`	Script for RTL VCD time-based power analysis
`./src/hdl/gate/mac.vg`	Gate-level netlist of the design
`./src/lib/snps/core_typ.db`	Technology library file
`./src/hdl/gate/mac.sdc`	Synopsys Design Constraints (SDC) file
`./src/annotate/mac.spef.gz`	Parasitic file
`./sim/rtlvcd.dump`	RTL VCD file
`./sim/vcd.dump.gz`	Gate-level VCD file

PrimeTime PX Script File

The tutorial contains two sample Tcl scripts for the time-based power analysis mode. The commands in the script are grouped into different sections, which represent the basic steps of power analysis. These steps which are common for all types of power analysis, are as follows:

1. Set the power analysis mode
2. Read and link the design
3. Set input transition and annotate parasitics
4. Read the switching activity file
5. Perform power analysis

The following example script shows the various steps in time-based power analysis:

Example 2-3 Example Script for Time-Based Power Analysis

```
#######################################################
#     Set the Power Analysis Mode
#######################################################
set power_enable_analysis TRUE
set power_analysis_mode time_based
#######################################################
#       link design
#######################################################
set search_path   "../src/hdl/gate ../src/lib/snps . "
set link_library    " * core_typ.db"
read_verilog   mac.vg
current_design mac
link
#######################################################
#       set transition time / annotate parasitics
#######################################################
read_sdc                  ../src//hdl/gate/mac.sdc
set_disable_timing [get_lib_pins ssc_core_typ/*/G]
read_parasitics           ../src/annotate/mac.spef.gz
#######################################################
check_timing
update_timing
report_timing
#######################################################
#       read switching activity file
#######################################################
read_vcd ../sim/vcd.dump.gz -strip_path tb/macinst
#######################################################
#       analyze power
#######################################################
check_power
set_power_analysis_options -waveform_format out -waveform_output vcd
update_power
report_power
quit
```

Steps for Analyzing Power

Set the current working directory to **./tutpx/time_based**.

Running PrimeTime PX

The tutorial contains two sample scripts for time-based power analysis using PrimeTime PX. You use the time_gatevcd.tcl to run the gate-level time-based analysis and tim_rtlvcd.tcl to run the RTL VCD time-based analysis. To run the timegatevcd.tcl script, use the following command:

```
%> pt_shell -f tim_gatevcd.tcl
```

PrimeTime PX runs in batch mode. The tool halts when all the commands in the script have been executed.

Reviewing the Power Report and Waveforms

At the end of the run, PrimeTime PX prints out the power report. It itemizes the power consumption on each hierarchical instance.

```
                                  Switch   Int      Leak      Total
Hierarchy                         Power    Power    Power     Power     %
------------------------------------------------------------------------------
-----
mac                                1.53e-03 2.13e-03 2.59e-07  3.65e-03
                                                                      100.0
  mult_21 (mac_DW02_mult_16_16_0) 7.16e-04 5.57e-04 1.49e-07  1.27e-03
                                                                       34.8
    U1/U9720 (mac_DW01_add_25_0)  2.02e-04 1.28e-04 1.60e-08  3.30e-04
                                                                        9.0
  add_23 (mac_DW01_add_33_0)      3.24e-04 2.49e-04 2.36e-08  5.74e-04
                                                                       15.7

                                 Peak     Peak           Glitch   X-tran
Hierarchy                        Power    Time           Power    Power
------------------------------------------------------------------------------
mac                              0.197 180.000-180.010 5.43e-05   4.68e-07
  mult_21 (mac_DW02_mult_16_16_0) 4.74e-02 546.650-546.660 3.36e-05
                                                                   0.000
    U1/U9720 (mac_DW01_add_25_0)      1.59e-02 3116.120-3116.130
                                                   2.28e-05 0.000
  add_23 (mac_DW01_add_33_0)          2.01e-02 6744.580-6744.590
                                                   1.63e-05 0.000
```

After the power analysis, a power .fsdb waveform file is saved in the current directory. To view it, use the waveform viewer called nWave. In the UNIX shell, enter `nWave &` to start the viewer. Specify `nWave -h` to view the usage options and their descriptions. This viewer requires a snps_fs_nwave license.

Alternatively, in the GUI, choose Power > View Waveforms to open nWave. Figure 2-1 and Figure 2-2 show the waveforms for peak power analysis and cycle accurate peak power analysis respectively.

Figure 2-1 Peak Power Analysis Waveform

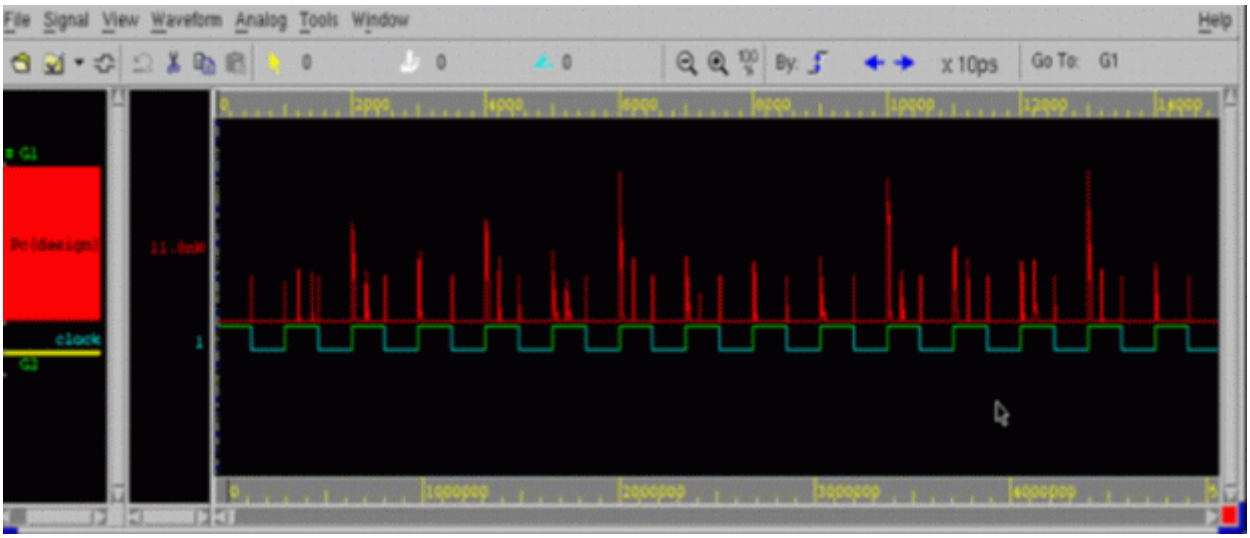
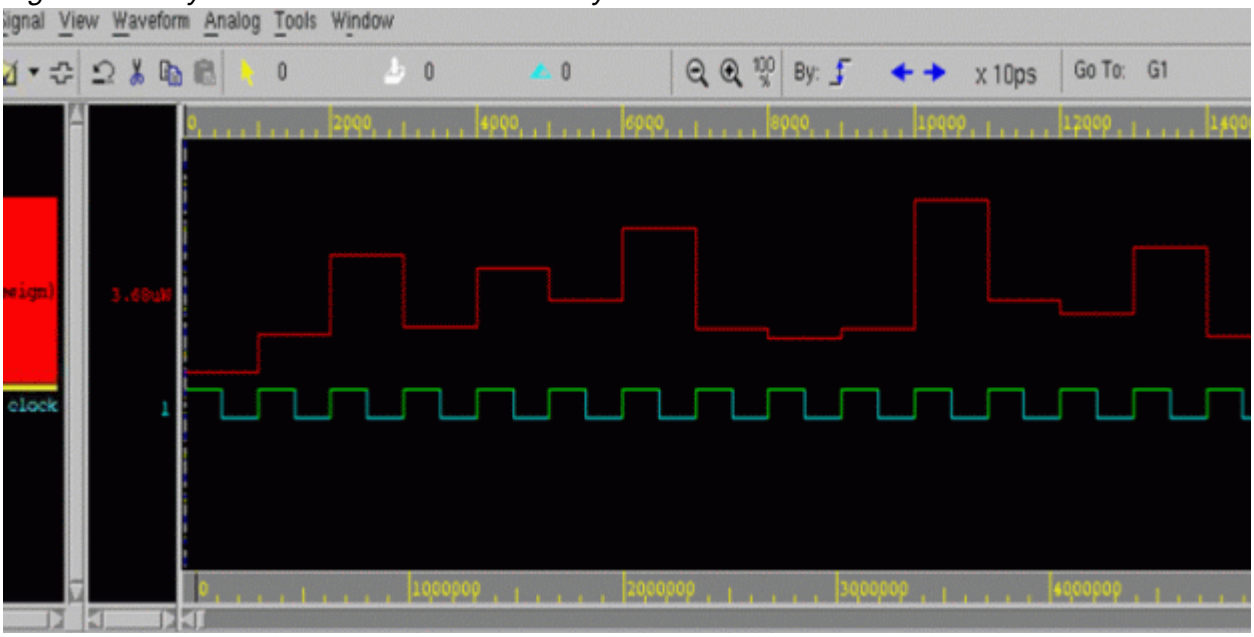


Figure 2-2 Cycle Accurate Peak Power Analysis Waveform



3

Power Analysis Modes in PrimeTime PX

This chapter describes the basic steps in the PrimeTime PX averaged and time-based power analysis modes. It contains the following sections:

Overview

PrimeTime PX supports different types of power analysis modes. In each of the modes the tool supports various options to suit different types of designs and applications. To perform power analysis for a design successfully, you must select the required power analysis mode and the specific options supported for the selected mode.

The user interface supports variables, commands and command options to select the power analysis mode and the supported options of the mode. The two basic types of power analysis mode supported are the averaged power analysis and the time-based power analysis. In each of the modes, the tool supports specific command options to perform certain type of analysis. You provide the switching activity information in the SAIF or VCD file formats. Irrespective of the mode you select and the options you choose, power analysis is performed during the execution of the `update_power` command.

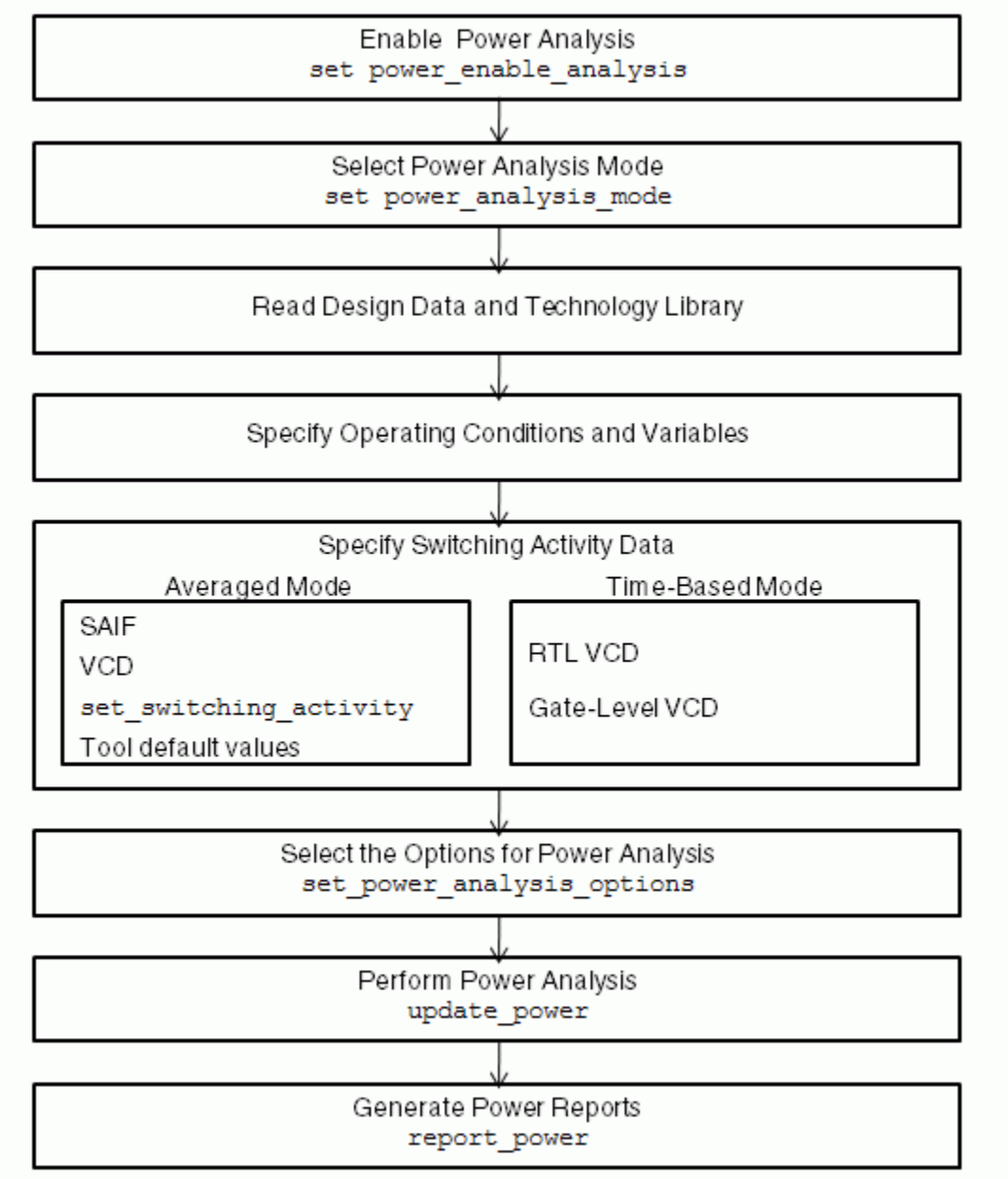
The following sections discuss in detail the usage of the commands, variables, and the command options supported in the various modes.

Power Analysis Flow

The steps for performing power analysis are described in the following sections and shown in Figure 3-3.

1. Enabling Power Analysis
2. Selecting the Power Analysis Mode
3. Reading the Design Data and Technology Library
4. Specifying Operating Conditions and Variables
5. Specifying Switching Activity Data
6. Performing Selective Power Analysis by Specifying Time Windows
7. Specifying the Options for the Power Analysis Mode
8. Vector Analysis
9. Setting the Power Derating Factor
10. Setting the Concurrent Multirail Power Analysis Mode
11. Performing Power Analysis
12. Generating Power Reports

Figure 3-3 PrimeTime PX Power Analysis Flow



Enabling Power Analysis

Set the `power_enable_analysis` variable to `true` to enable power analysis and see power data. The default value of this variable is `false`. If you do not set this variable to `true`, you cannot see power data. A PrimeTime PX license is required to use this variable.

You specify the operating conditions for the analysis using the `set_operating_conditions` command. This enables the tool to use the appropriate set of parameter values from the technology library. These parameter values generally include fabrication process, operating temperature, and power supply voltage as characterized by the ASIC vendor.

Selecting the Power Analysis Mode

You select the power analysis mode by using the `power_analysis_mode` variable. The syntax of this variable is as follows:

```
set power_analysis_mode averaged | time_based
```

You must set this variable before using any of the power commands. If you do not set this variable, PrimeTime PX, by default, performs averaged power analysis.

After power analysis, if you change the value of the variable, the switching activity and power information are lost. So you must re-read the switching activity data to perform power analysis.

For more details on this variable, refer to the man page.

Reading the Design Data and Technology Library

The tool supports netlists in Verilog, VHDL, EDIF, .db, .ddc, and Milkyway formats. The technology library must be in the .db (Synopsys database) format. Both NLPM and CCS libraries are supported.

Design data includes parasitic information, such as port and net loads, wire load models, and back-annotated parasitics. It also includes transition time information and, for averaged power analysis, the design constraints, as compiled with PrimeTime in a Synopsys Design Constraints (SDC) file. For more information, see the PrimeTime documentation.

CCS Power Libraries

CCS power libraries contain unified library data for power and rail analysis and optimization, which ensures consistent analysis and simplification of the analysis flow. By capturing current waveforms in the library, you can provide more accurate identification of potential problem areas.

Both CCS and NLPM data can coexist in a cell description in the .lib file format. That is, a cell description can have only NLPM data, only CCS data, or both NLPM and CCS data. PrimeTime PX uses either NLPM data or CCS data for the power calculation.

You use the `power_model_preference nlpm | ccs` variable to specify your power model preference when the library contains both NLPM and CCS data. The default is `ccs`. Using CCS power libraries does not change the use model.

PrimeTime PX also supports compact CCS power format. The compact format significantly reduces the size of the library without affecting the accuracy.

For more information about CCS power libraries and how to generate them, see the Library Compiler documentation.

Specifying Operating Conditions and Variables

To specify the operating conditions for the analysis use the `set_operating_conditions` command, which enables PrimeTime PX to use the appropriate set of parameter values from the technology library. These parameter values generally include fabrication process, operating temperature, and power supply voltage as characterized by the ASIC vendor.

Specifying Switching Activity Data

The type and format of switching activity data supported by PrimeTime PX depends on the power analysis mode that you select using the `power_analysis_mode` variable. For more details, see “Selecting the Power Analysis Mode” on page 3-4.

Name Mapping

For accurate results power analysis and optimization results occur when the switching activities are accurately associated with the correct design objects in the gate-level netlist. For this to occur, the RTL names must map correctly to their gate-level counterparts.

PrimeTime PX performs default name mapping, as follows:

- RTL registers synthesize to sequential elements whose names are derived by adding “_reg” to the original RTL name. For example, an RTL register named A becomes A_reg at the gate level. PrimeTime PX applies this built-in mapping rule to nets that are not annotated.
- Bus dimension Verilog separators “[ ]” map to “_”. For example, register reg a[7] maps to a_reg_7_ and the Q output maps to a_7_.
- The logical behavior of registers mapped to QB pins of register outputs are inverted in the VCD file and mapped to the appropriate pins. For example, suppose reg_a synthesizes into a DFF signal, a_reg, and a_reg uses the QB output pin. PrimeTime PX inverts the VCD data for reg_a and annotates it to the QB pin of a_reg.

This default name mapping behavior occurs only when the Q pin is not linked to a net. If the Q pin is connected to a net, the tool maps the register to the Q pin without inversion; the activity at the net of the QB pin propagates instead of being annotated.

If name mapping inaccuracies occur during synthesis, use the `set_rtl_to_gate_name` command to map the RTL names to the gate-level names. This command allows you to specify unique name mappings and global substitutions. You can use Power Compiler to generate a name-mapping file formatted especially for PrimeTime PX.

For more information about the `set_rtl_to_gate_name` command, see the man page.

To define signals that must never be mapped or to unmatch erroneously matched signals, specify the `unset_rtl_to_gate_name` command.

Name-mapping inaccuracies can also occur when you generate activity data from mixed-language simulation. For example, because VHDL is case-insensitive, VCD files generated from VHDL testbenches might not maintain the original case. To address this issue, PrimeTime PX provides the following variable to control the handling of mixed-language simulation activity files:

```
power_read_activity_ignore_case true | false
```

By default, this variable is `true`. To enable case-sensitive name mapping set this variable to `false`.

Generating the Name Mapping Report

Use the `report_name_mapping` command to check the SAIF or VCD name mapping that you specified with the `set_rtl_to_gate_name` command. The `report_name_mapping` command returns a list of user-defined name mapping rules. The name mapping rules are case-sensitive unless you set the `power_read_activity_ignore_case` variable to `true`.

After reading a VCD or SAIF file, use the `report_switching_activity` command to report the nets annotated by name mapping.

The name mapping rules have the following priority (highest to lowest):

1. Name mapping not required; direct name matches.
2. Never-match objects as specified with the `unset_rtl_to_gate_name` command
3. User-defined matches as specified with the `set_rtl_to_gate_name` command, including global substitutions
4. Default name mapping

The `report_name_mapping` command reports the value of the inverted flag for each name mapping entry. It also contains a summary of the number of name mapping entries, the number of entries with inverted flag set, and so on, as shown in Example 3-4.

Example 3-4 Report generated by the report_name_mapping command

```
*************************************
Report : report_name_mapping
Design : test
Version: D-2010.06
Date   : Mon Mar 22 10:11:59 2010
*************************************

   Attributes
   ----------
      v - Inverted
RTL Name                  Gate Level Name      Type       Attributes
-------------------------------------------------------------------------
a_cell                    U3                   cell
a_pin                     U3/A                 pin
a_net                     n272                 net
```

Reporting Name Mapping Without Annotating the Switching Activity

PrimeTime PX supports the `report_activity_file_check` command to provide information on how the tool performs the name mapping for a specified activity file, without annotating the activity. This command does not affect the results of the annotation. The report generated by the `report_activity_file_check` command contains the various object types of the activity file and the name mapping method used, as shown in Example 3-5.

Example 3-5 Report generated by the report_activity_file_check command

```
*************************************
Report : Activity File Matching Check
Design : test
*************************************

-----------------------------------------------------------------
File Type : saif
-----------------------------------------------------------------
Activity     Activity  Design   Design
File         File      Object   Object
Object Type  Name      Type     Name
-----------------------------------------------------------------
net          /Ua       Net      a
net          /Ub       Net      b
net          /Ud       Net      c1
```

Switching Activity Formats Supported in Averaged Power Analysis

In the averaged power analysis mode, you can specify switching activity information from multiple sources in the following formats.

- SAIF Format

You use the `read_saif` command to specify the activity information in the SAIF file format that is generated from RTL or a gate-level netlist.

- VCD Format

  You use the `read_vcd` command to read the VCD file that contains the switching activity information. Use the `-rtl` option of the `read_vcd` command to specify that the VCD is generated from an RTL simulation. If the VCD file is generated from a zero-delay simulation use the `-zero_delay` option. When neither option is specified, the tool assumes that the file is a gate-level VCD file.

  In the averaged mode, PrimeTime PX supports reading activity data from multiple VCD files. For the different VCD files, if you specify different time values for the `-time` option, of the `read_vcd` command, the tool issues a warning message.

  When you read a SAIF file or an RTL VCD file for power estimation, use the `set_rtl_to_gate_name` command to map the RTL and gate-level object names. This command is especially necessary if you have performed only the RTL simulation for generating the backward SAIF file. Because the RTL object names can change after synthesis, the `read_saif` or `read_vcd` command is not able to map the names present in the RTL SAIF or VCD file to the gate-level objects, which can result in inaccurate results. You can avoid this by using the `set_rtl_to_gate_name` command. For more information, see the command man page.

  Note:
  When you apply switching activity from multiple sources, the VCD annotation is applied last, irrespective of the sequence in which you specify the switching activity.

- Activity Files Generated from a SystemVerilog Simulation

  You can use the activity files generated from a SystemVerilog simulation if the files are in VPD or .fsdb format. You must specify the format as SystemVerilog, using the `-format` option of the `read_vcd` command.

- `set_switching_activity` Command

  You use this command to specify different kinds of switching activity information, such as toggle count, glitch count, and static probability, on specific nets, pins, ports, and cells of the design. For more details of this command, see “Annotating Switching Activity Using the set_switching_activity Command” on page 5-9.

- Tool Default Values

  When you do not specify switching activity information, the tool assumes default values. The default toggle rates and zero-delay simulation are used to propagate the activity throughout the circuit. This means that all values are updated instantaneously.

  The following commands and variables affect the default toggle rates used by the tool:

  ```
  create_clock
  set power_default_toggle_rate
  ```

```
set power_default_static_probability
set power_default_toggle_rate_reference_clock
set_switching_activity
reset_switching_activity
set_case_analysis
```

The default value of the static probability and toggle rate are 0.5 and 0.1, respectively. The period is set to the clock frequency of the related clock. For more details on default settings, see "Using the Default Switching Activity" on page 5-12.

Note:

If your design contains multicycle path, you should specify a switching activity; using the tool default values does not guarantee accurate results.

Switching Activity Formats Supported in Time-Based Power Analysis

When in the timed-based analysis mode, you must provide event-based activity for the design with VCD, VPD, or .fsdb formats. PrimeTime PX calculates the power for every event for accurate results.

The time-based power analysis mode requires that you specify the switching activity information in VCD format. Use the `read_vcd` command to read the VCD file that contains the activity information. Two types of VCD files are supported for specifying the activity data:

- RTL VCD

  You use the `read_vcd -rtl` command to read the activity file. If the VCD file is generated from a zero-delay simulation use the `-zero_delay` option. When you use the `-rtl` option, the tool performs name mapping and event propagation. You can also use the `set_rtl_to_gate_name` command to map the RTL and gate-level object names. Because the RTL object names can change after synthesis, the `read_vcd` command cannot map the names present in the RTL VCD file to the gate-level objects. This can result in inaccurate results. You can avoid this by using the `set_rtl_to_gate_name` command. For more information, see the command man page.

  When you use the `-rtl` or the `-zero_delay` option, the tool generates cycle-accurate waveform. The clock cycle is used as the default waveform interval.

- Gate-level VCD

  While reading the VCD file using the `read_vcd` command, if you specify neither the `-rtl` option nor the `-zero_delay` option, the tool assumes that the file is a gate-level VCD file.

In time-based mode, PrimeTime PX does not support reading activity information from multiple VCD files. If you read multiple VCD files, PrimeTime PX overwrites the VCD files previously specified, and only the last VCD file specified is used for the analysis.

If you use the `read_vcd -pipe_exec` command, the tool defers reading the VCD file, including the header annotation.

Table 3-4 shows how PrimeTime PX handles different types of VCD files in the time-based power analysis mode. The type of VCD file is indicated by the different option used with the `read_vcd` command.

Table 3-4 Types of VCD supported in the time-based mode

VCD Types	Name Mapping	Event Propagation	Enforce Cycle Accurate Peak Power
Gate-level VCD using the `read_vcd` command without the `-zero_delay` and `-rtl` options	Off	Off	No
Gate-level zero-delay VCD using the `read_vcd -zero_delay` command	Off	Off	Yes
RTL VCD using the `read_vcd -rtl` command	On	On	Yes

Performing Selective Power Analysis by Specifying Time Windows

The `read_vcd` command supports the `-time` and `-when` options that allow you to select specific time slices to use and analyze the selected time slices of a value change dump (VCD) file.

The `-time` option of the `read_vcd` command allows you to analyze the selected time slices of a value change dump (VCD) file. Using this feature you can analyze the power when the design is operating in a particular state or mode. You can also do this by finding the time slices of the simulation in which the testbench caused the design to be in a particular mode and then specify those time slices with the `-time` option of the `read_vcd` command.

The `-when` option to the `read_vcd` command helps you simplify the process of finding which time slices to use. The `-when` option takes a Boolean expression as an argument, and it accepts design net or pin names as variables in the expression. When the Boolean evaluates to `true`, the tool automatically selects time slices from the VCD file. The time slices where the logical values on nets and pins evaluates to false are excluded from analysis.

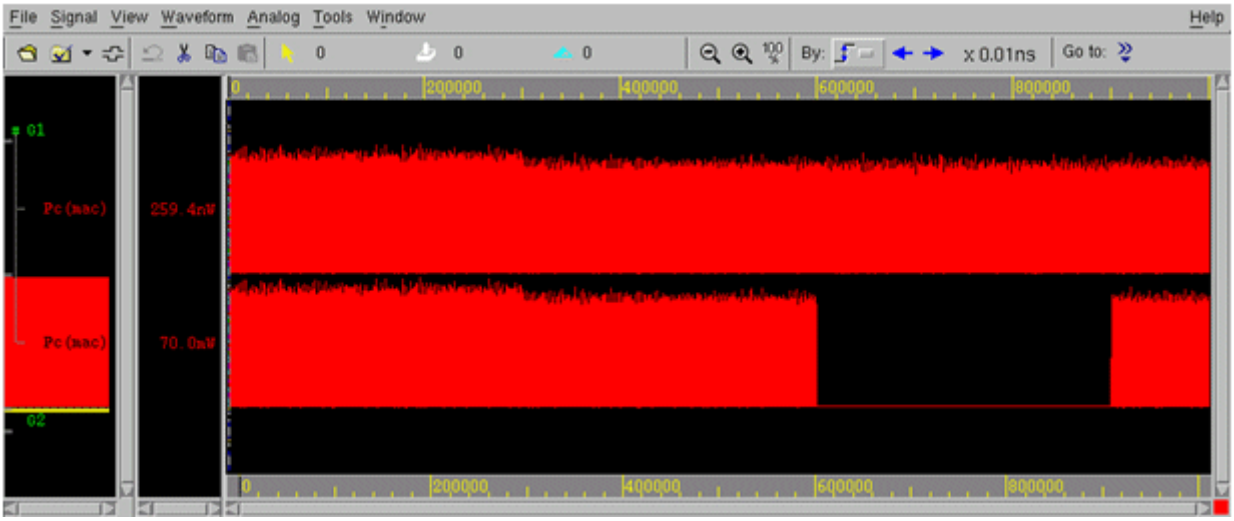
This feature is supported in both averaged and time-based power analysis modes. The reported average power in both modes considers only the time slices in which the Boolean expression is `true`.

The following example shows the portion of the VCD in which the N5 signal is `true`, is selected.

```
read_vcd "../sim/vcd.dump" -strip_path "tb/macinst" -when {N5}
```

In Figure 3-4, the top waveform shows the power without using the `-when` option. The bottom waveform shows the power with the `-when {N5}` option specified with the `read_vcd` command. You can see that the N5 signal is true from the beginning of the simulation until 6µs, false from 6µs to 9µs, and true from 9µs to the end of the simulation.

Figure 3-4 Waveforms with and without using the -when option of the read_vcd command



In the following example, the time slices selected from the VCD are the portions when both N5 and N4 are logical 1:

```
read_vcd "../sim/vcd.dump" -strip_path "tb/macinst" -when {N5 & N4}
```

Specifying the Options for the Power Analysis Mode

After selecting the power analysis mode, you can use the options supported by the mode to control its behavior using the `set_power_analysis_options` command. While some options of this command can be used for all the modes, some options are specific to a certain mode. If you use an option that is not supported for the selected mode, the tool issues an error message. If you specify multiple settings, using the command multiple times, the previous settings are overwritten.

The syntax of the `set_power_analysis_options` command is as follows:

```
set_power_analysis_options
[-static_leakage_only]
[-waveform_interval <float>]
[-cycle_accurate_cycle_count <integer>]
[-cycle_accurate_clock clock]
[-waveform_format fsdb| out| none]
[-waveform_output file]
[-include top| all_without_leaf | all_with_leaf]
[-include_groups list]
[-cells list]
```

The `-static_leakage_only` option is supported only in the averaged power analysis mode.

The `-waveform_interval`, `-cycle_accurate_cycle_count`, `-cycle_accurate_clock`, `-waveform_format`, `-waveform_output`, `-include`, and `-include_groups` options are supported only in the time-based power analysis mode. For more details refer to the command man page.

Vector Analysis

Before you perform power analysis, you can qualify your vectors for power analysis by using the vector analysis feature. This feature provides you with feedback about your vectors by plotting the average toggle rate per interval, over the specified time period. PrimeTime PX reads the VCD file and writes an activity waveform for the top-level module of the VCD file. PrimeTime PX partitions the total VCD simulation time into intervals. For each interval, the tool calculates the average activity using the following formula:

$$\text{Average Toggle Rate} = \frac{\text{Number of Toggles on All the Signals}}{\text{Number of Signals per Interval}}$$

You can view the activity waveforms to determine if the testbench simulated as expected and if the vectors have covered the design well enough to be useful as inputs for power analysis.The coverage per interval is calculated as follows:

$$\text{Coverage} = \frac{\text{Number of Signals with Atleast One Toggle}}{\text{Number of Signals}}$$

You use the `write_activity_waveforms` command to generate the activity waveforms from the VCD file. You can use this command even before you read the design data.The output generated by the vector analysis has the same names as the instances in the original VCD file. The output for the waveforms is in the .fsdb format. You can read the output file in the waveform viewer such as nWave in the same way that you read the .fsdb file containing the power waveforms as shown in Figure 3-5

To ensure that you do not miss the peak activity interval if the windows of high activity fall into separate intervals, use the `-peak_window` option of the command. Using this option you can specify the sliding window period in which to you monitor the average activity per interval. The tool searches for the peak period in the VCD file and reports the peak activity using a sliding window. This option affects the way in which the peak periods are reported in text format. It does not affect waveform generation.

Note:

The value specified with the `-peak_window` option must be a multiple of the `-interval` option.

This command supports the `-peak_type` option. You can specify either `minimum` or `maximum` with this option. PrimeTime PX searches for an interval of maximum or minimum activity based on the argument specified. Otherwise, if you have specified the `-coverage` option, the tool searches for coverage. If you do not use the `-peak_type` option, the tool searches for an interval with maximum activity or coverage by default.

This command also supports options to specify the level of hierarchy for cells to include in the waveform, to specify signals to be excluded from the waveforms, and so on. For more details about this command, refer to the command man page.

Example 3-6 shows how the `-peak_window` option affects the report generated by the `report_activity_waveforms` command. By specifying the `-peak_window 5`, the reported peak interval is 5 ns, although the interval for the waveform is 0.2 ns.

Example 3-6 Activity Waveform Report

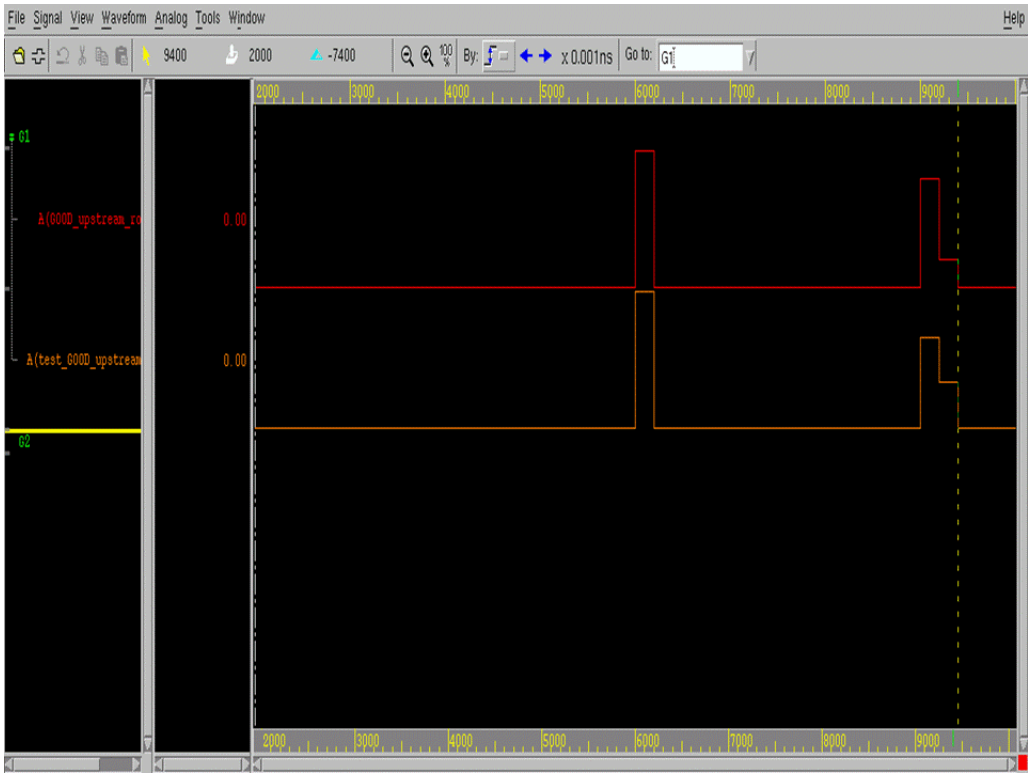
```
write_activity_waveforms -vcd myvcd.vcd -output myvcd.out \
 -interval 0.2 -peak_window 5 -hier 2

report_activity_waveforms
****************************************
Report : Time-Based Switching Activity
Activity File: myvcd.vcd
****************************************

Block                    # Signals  Peak           Peak        Average
Name                                Interval       Toggle      Toggle
                                    (ns)           Rate        Rate
------------------------------------------------------------------------
GOOD_upstream_router_tb     1383     4.400-9.400    0.00145  0.000882
test_GOOD_upstream_router   1335     4.400-9.400    0.000899 0.000548
------------------------------------------------------------------------
```

The following figure shows the waveform for the report.

Figure 3-5 Time-Based Activity Waveform



Setting the Power Derating Factor

The power derating feature allows you to apply a multiplication factor to the switching power, internal power and leakage power. You can apply the power derating factor to different objects such as a design, a cell, a library cell and hierarchical and leaf cells. The power derating feature supports both averaged and time-based power analysis modes and the result of the power analysis is determined by the derating factor that you specified.

Commands Supporting Power Derating

The following commands allow you to set, reset and report the power derating capability.

- `set_power_derate`

  Use this command to set power derating factors on different power components for the current design or a specified list of objects or power groups. When you do not specify any object, this command sets the power derating factor on the current design. The default value of 1.0 is used by the tool, if you do not specify any power derating factor. For more details, see the command man page.

- `reset_power_derate`

  Use this command to reset all the specified power derating factors in the specified object list or power groups. If no object or power group is specified, the command resets the power derating factors of all objects in the current design, including those set on the power groups. For more details see the command man page.

- `report_power_derate`

  Use this command to report the power derating factor set on the current design and the specified objects. For a detailed example, see “Reporting Power Derate Factors” on page 9-9.

Specifying Power Derating Factor on Design Objects

The power derating factor is set on different design objects such as current design, library cells, hierarchical and leaf cells of the design with the `set_power_derate` command. You can also specify derating factors on the various power components such as:

- Internal power
- Switching power
- Leakage power

The power derating factors are specified on the objects in the following order of precedence, from highest to lowest:

1. Leaf cell and power group

A power group is treated as collection of cells by the power derating feature.

2. Hierarchical cell
3. Library cell
4. Design and design cell

Note:
You can set the power derating factor on the same design object and for the same power component multiple times. The last value that you set overrides the previous settings.

Commands Affected by the Power Derating Factor

The following commands use the power derating factor:

- `report_power`

  Reports the power of the design.

  The `report_power` command supports the `-derate` option. This option is used only in the cell-based averaged or time-based power report. When you specify this option, additional columns are displayed in the respective report, that show the power derating factors used in power calculation. The `report_power` command can generate,

  - The derated average power report in the averaged and time-based power analysis modes.
  - The derated time-based power report in either the .fsdb or .out output format and peak power reports.

- `report_power_calculation`

  Calculates and reports power values, using the power derating factors that you specified.

- `reset_design`

  Resets the design and its power derating factors.

  The power derating factors are also reset, while resetting the design using the `reset_design` command.

- `estimate_clock_network_power`

  Estimates the power of the clock networks by applying the power derating factors that you specified to the buffer or the current design. This command is supported only in the averaged power analysis mode.

- `get_attribute`

  This command gets the derated power numbers, based on the power derating factor that you specified. This command also gets the power derating factors on cells and current design using the power derating attributes.

Setting the Concurrent Multirail Power Analysis Mode

PrimeTime PX supports concurrent power analysis on multiple power rails. This feature improves the performance because, a single run of the `update_power` command analyzes multiple power rails or power supply nets in the UPF mode, simultaneously. The concurrent multirail power analysis is supported in both averaged and time-based power analysis modes.

To enable this feature, set the `power_enable_multi_rail_analysis` variable to `true`. When you run the `update_power` command with this variable set to `true`, the tool simultaneously updates all the valid power rails in the design.

Performing Power Analysis

The tool performs power analysis during the execution of the `update_power` command. The analysis is performed based on the power analysis mode that you select and the various options that you specify using the `set_power_analysis_option` command.

Power analysis is triggered consistently for the various power analysis modes during the execution of the `update_power` command. After the analysis, the `update_power` command generates the power data.

In the averaged mode, if you do not specify activity data, the tool assumes default values. However, in the time-based mode, you must specify the activity data in the VCD file format before using the `update_power` command. Otherwise, PrimeTime PX issues an error message.

Generating Power Reports

Use the `report_power` command to generate an averaged or time-based power report that contains the power consumption for the design.

The following example shows the power report in the averaged power analysis mode:

```
****************************************
Report : Averaged Power
Design : mac
Version: B-2008.12
Date   : Tue Nov 18 06:48:38 2008
****************************************

 Attributes
 ----------
 i  -  Including register clock pin internal power
 u  -  User defined power group
```

```
                          Internal    Switching  Leakage    Total
Power Group               Power       Power      Power      Power   (
%)  Attrs
--------------------------------------------------------------------
--------
io_pad                    0.0000      0.0000     0.0000     0.0000  (
0.00%)
memory                    0.0000      0.0000     0.0000     0.0000  (
0.00%)
black_box                 0.0000      0.0000     0.0000     0.0000  (
0.00%)
clock_network             8.041e-04   0.0000     0.0000   8.041e-04
(21.28%)  i
register                  4.705e-04   1.095e-04 6.660e-08 5.801e-04
(15.35%)
combinational             9.012e-04   1.219e-03 1.836e-07 2.121e-03
(56.14%)
sequential                5.301e-05   2.200e-04 9.163e-09 2.730e-04 (
7.23%)

 Net Switching Power  = 1.549e-03   (41.00%)
 Cell Internal Power  = 2.229e-03   (59.00%)
 Cell Leakage Power   = 2.594e-07   ( 0.01%)

Total Power            = 3.778e-03  (100.00%)
1
```

The following example shows the power report with peak power for time-based power analysis. You can also use the nWave utility to view power waveforms.

```
****************************************
Report : Time Based Power
Design : mac
Version: B-2008.12
Date   : Tue Nov 18 06:23:15 2008
****************************************

 Attributes
 ----------
 i  -  Including register clock pin internal power
 u  -  User defined power group

                           Internal   Switching  Leakage    Total
Power Group               Power      Power      Power      Power    (    %)
Attrs
--------------------------------------------------------------------------
-------
io_pad                    0.0000     0.0000     0.0000     0.0000   (0.00%)
```

```
memory                  0.0000     0.0000     0.0000     0.0000   (0.00%)
black_box               0.0000     0.0000     0.0000     0.0000   (0.00%)
clock_network           7.133e-04  0.0000     0.0000    7.133e-04
(19.44%)  i
register                4.701e-04  1.095e-04  6.660e-08 5.797e-04
(15.79%)
combinational           8.967e-04  1.207e-03  1.836e-07 2.104e-03
(57.33%)
sequential              5.297e-05  2.200e-04  9.163e-09 2.730e-04 (
7.44%)

 Net Switching Power  = 1.537e-03   (41.87%)
 Cell Internal Power  = 2.133e-03   (58.12%)
 Cell Leakage Power   = 2.594e-07   ( 0.01%)
 ---------
Total Power           = 3.670e-03  (100.00%)

X Transition Power    = 2.171e-07
Glitching Power        = 3.018e-05

Peak Power             =    0.0968
Peak Time              =   1956.00

1
```

The following example uses the `-rails` option of the `report_power` command to specify a list of design rail groups or power supply net groups in the UPF mode, to be reported.

```
****************************************
Report : Averaged Power
Design : testcase
Version: D-2010.06
Date   : Tue Apr  6 14:22:30 2010
****************************************

Current Power Rail: all
Power Report For Rails: VDD1 VDD2

Attributes
 ----------
 i  -  Including register clock pin internal power
 u  -  User defined power group

               Internal  Switching  Leakage  Total
Power Group    Power     Power      Power    Power  (    %)  Attrs
--------------------------------------------------------------------
io_pad         0.0000    0.0000     0.0000   0.0000 ( 0.00%)
memory         0.0000    0.0000     0.0000   0.0000 ( 0.00%)
black_box      0.0000    0.0000     0.0000   0.0000 ( 0.00%)
clock_network  0.0000    0.0000     0.0000   0.0000 ( 0.00%)    i
register       0.0000    0.0000     0.0000   0.0000 ( 0.00%)
combinational 1.842e-05 4.886e-07 9.315e-11 1.891e-05 (100.00%)
sequential     0.0000    0.0000     0.0000   0.0000 ( 0.00%)

   Net Switching Power  = 4.886e-07   ( 2.58%)
   Cell Internal Power  = 1.842e-05   (97.42%)
   Cell Leakage Power   = 9.315e-11   ( 0.00%)
                          ---------
Total Power             = 1.891e-05  (100.00%)
```

You can also generate your own reports by using tool command language (Tcl) constructs and accessing the power attributes. To save and restore your power parameters, make sure that you set the `power_enable_analysis` variable to `true`.

Saving and Restoring PrimeTime PX Sessions

The `save_session` and `restore_session` commands allow you to save the current state of a PrimeTime PX session and restore the same session later. When you need to examine the results of an earlier power analysis, using the save/restore feature avoids repeating the time-consuming `update_power` portion of the analysis. Using the `restore_session` command takes you to the same point in the analysis and uses only a small fraction of the original runtime.

You can also restore any PrimeTime or PrimeTime SI timing analysis sessions and continue doing power analysis.

4

Invocation and Graphical User Interface

The PrimeTime PX graphical user interface (GUI) provides a way to view design data and analysis results in graphical form, including histograms and waveforms. This chapter contains the following sections:

Starting and Ending a Shell Session

Before you can use PrimeTime PX, the software must be installed and licensed at your site. For information on installation and licensing, see the documentation that comes with the software release.

To start pt_shell (the PrimeTime PX command-line interface), enter the following command at the operating system prompt:

```
%> pt_shell
```

PrimeTime PX automatically checks out a PrimeTime license and checks whether you have a PrimeTime PX license or not. If yes, the following initial message displays. The PrimeTime PX license is checked out when the first power-related command is executed.

```
                    PrimeTime PX(R)
          Version 2005.06 -- June 1, 2005
     Copyright (c) 1988-2005 by Synopsys, Inc.
                ALL RIGHTS RESERVED

This program is proprietary and confidential information of
Synopsys, Inc. and may be used and disclosed only as
authorized in a license agreement controlling such use and
disclosure.

pt_shell>
```

You enter commands at the pt_shell prompt (pt_shell>). PrimeTime PX responds with text messages in the terminal window below your command entry line.

To end the PrimeTime PX session, enter `quit` at the shell prompt.

Analysis Flow With the GUI

The power analysis driver in the GUI helps you visualize and understand the nature of power problems in the design, including the type, number, magnitude, and locations of the problems. You can use the visual analysis tools after you have read, linked, and calculated the power of the design.

Typically, an analysis session consists of the following tasks:

1. Read and calculate the power for the design.
2. Start the GUI.

3. For a high-level overview of the design's power consumption and to identify specific items that contribute to the highest power (such as high activity and capacitance), generate a power histogram.
4. Generate a toggle histogram to find the instances with the most activity.
5. View the instance and net capacitance profiles to determine the source of high power.
6. Examine the logic vectors and power waveforms to determine the cause of peak power.

Starting and Stopping a GUI Session

Before you start the GUI, make sure the DISPLAY environment variable is set to your UNIX display name.

To start a PrimeTime PX session that includes the GUI, enter the following at the system prompt:

```
%> pt_shell -gui
```

You can optionally set the DISPLAY environment variable when you invoke the PrimeTime PX GUI. For example,

```
%> pt_shell -gui -display 192.180.50.155:0.0
```

You can also start the GUI from a pt_shell session. To start the GUI, use the following command at the pt_shell prompt:

```
pt_shell> start_gui
```

The `start_gui` command does not work when used in a script.

To stop the GUI while still keeping the original pt_shell session going in the terminal window, use the `stop_gui` command or choose File > Close GUI from the menu. To exit from PrimeTime PX entirely, choose File > Exit.

Using the GUI

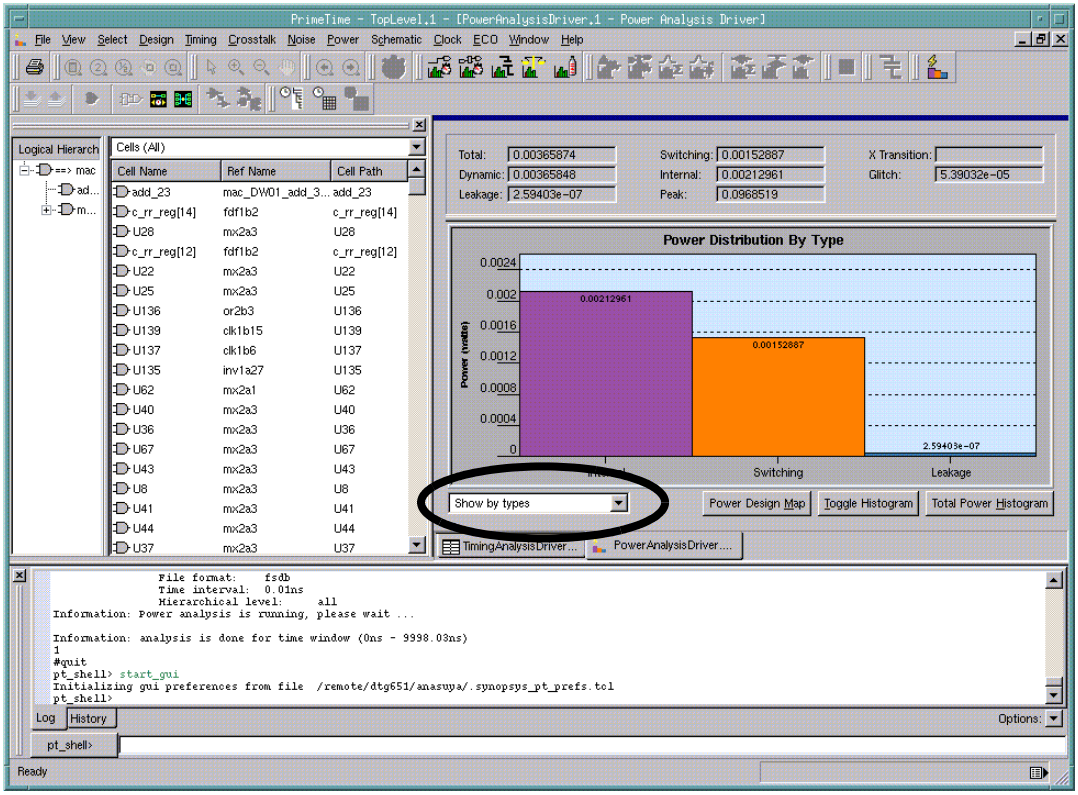
This section describes the power-related features in the PrimeTime PX GUI.

After you have read and calculated power and opened the GUI, you can view and analyze your power data by right-clicking within the main window and choosing Power > Analysis Driver or by clicking the power icon, which is as shown in Figure 4-6.

Figure 4-6 Power Icon of PrimeTime PX GUI

Power Analysis Driver. The analysis driver allows you to easily analyze dynamic power and static power. As shown in Figure 4-7, the GUI opens with the analysis driver window on the right to a view of the total power consumption for the current design with its distribution among dynamic, switching, internal, and leakage consumption.

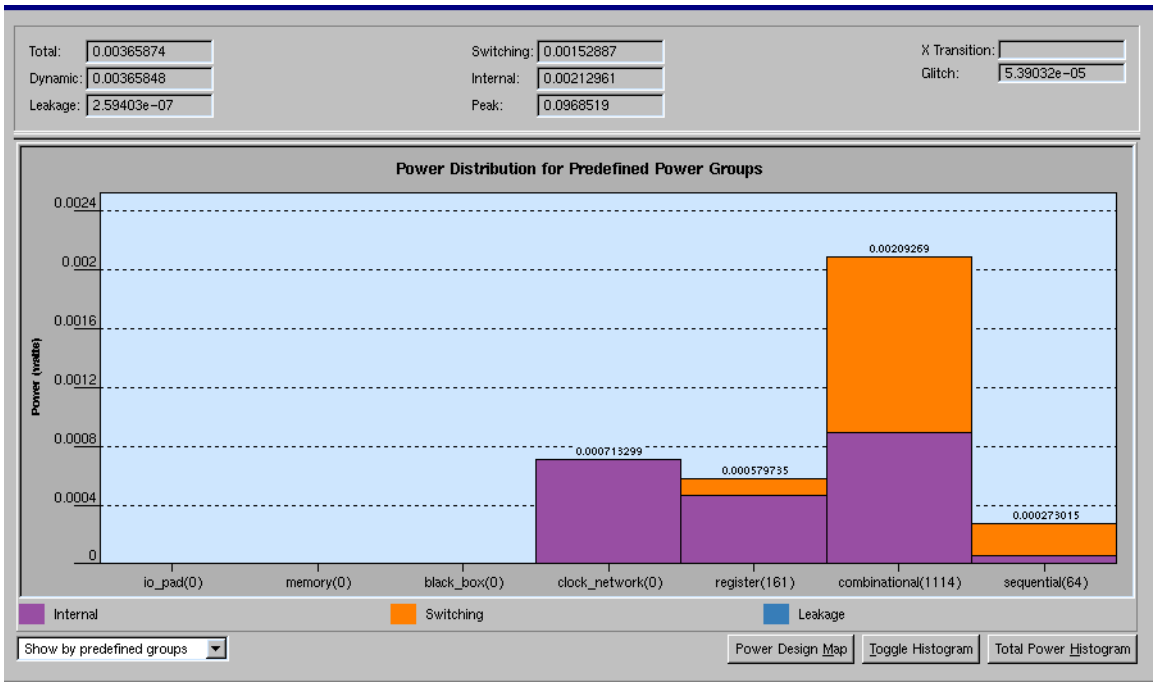
Figure 4-7 Analysis Driver Window



The list box in the lower-left corner of the Power Distribution display (circled) allows you to show the power distribution by power types (the default), by predefined groups, and by user-defined groups.

Figure 4-8 shows the power distribution for predefined groups. The numbers inside parentheses indicate the number of cells in the associated groups.

Figure 4-8 Power Distribution of Predefined Power Groups



Design Map. Click the Power Design Map button shown in Figure 4-8 to view the total power density as a map of the design. The power density displays as a treemap, which is a visualization tool that can help you identify anomalies in large hierarchical data sets.

Treemaps convey hierarchical data with squares that represent cells in the hierarchy. The thickness of the lines between the squares tells you where you are in the hierarchy; that is, the thicker the line, the closer you are to the root of the hierarchy tree.

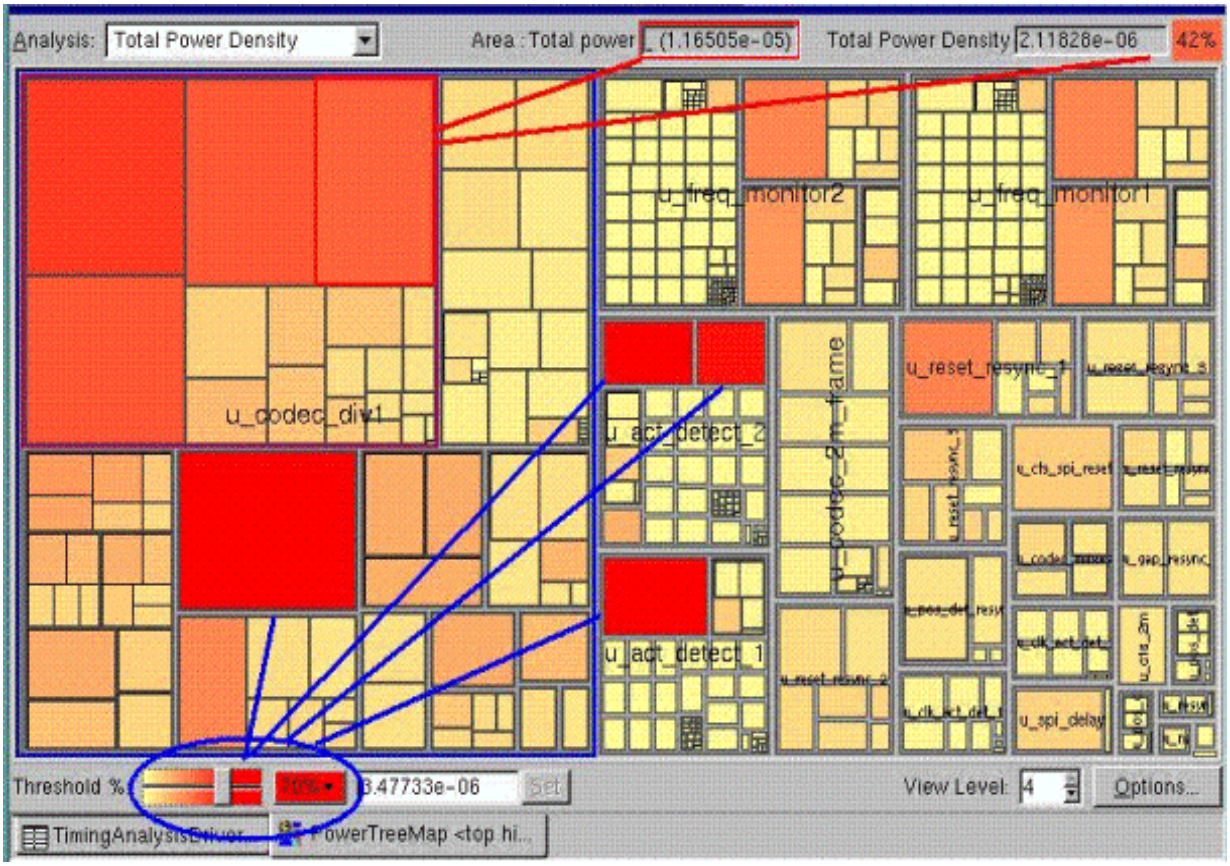
Use the Analysis box to change the parameters used for mapping how the color and node size indicators are used. Your choices are shown in Table 4-5.

Table 4-5 Treemap Analysis Types

Analysis types	Color indicator	Node size indicator
Total Power Density	Relative degree of total power consumption	Relative total power
Internal Power Density	Relative degree of internal power consumption	Relative internal power
Switching Power Density	Relative degree of switching power consumption	Relative switching power
Leakage Power Density	Relative degree of leakage power consumption	Relative leakage power
Toggle Coverage	Percentage of nets with one or more toggles	Relative net count
Toggle Average	Toggles per net count	Relative net count
% Net Annotated	Percentage of annotated nets	Relative net count
% Voltage Threshold	Relative percentage of threshold voltage cells	Relative leakage power
Total Power	Total power	Relative cell size within design
Internal Power	Internal power	Relative cell size within design
Switching Power	Switching power	Relative cell size within design
Leakage Power	Leakage power	Relative cell size within design

For example, Figure 4-9 shows a treemap for total power density in which the area of each node represents its corresponding relative total power within the design. The color gradient progresses from red to yellow, with red indicating the highest power consumption.

Figure 4-9 Treemap



Right-click within the treemap to access a context-sensitive menu that allows you view histograms.

The Threshold % slider circled in blue adjusts the value at which a parameter is considered critical. In Figure 4-9, sliding the control to the left causes more cells to appear red. A context-sensitive menu appears when you right-click the slider; you can customize the color selection and minimum and maximum scale colors.

The View Level option controls the number of hierarchical levels shown in the treemap.

Click the Customize button to view the Power Treemap Options dialog box from which you can choose to open new treemaps in new windows and specify the toggle coverage threshold, as needed.

Design Hierarchy Browser. Use the panes to the left of the analysis driver to browse the design hierarchy. As shown in Figure 4-7, the Cells (All) view displays cell attributes. Use the arrow to also view the attributes for the following menu selections:

- Cell Hierarchical
- Pins/Ports
- Pins of Child Cells
- Nets
- Clocks

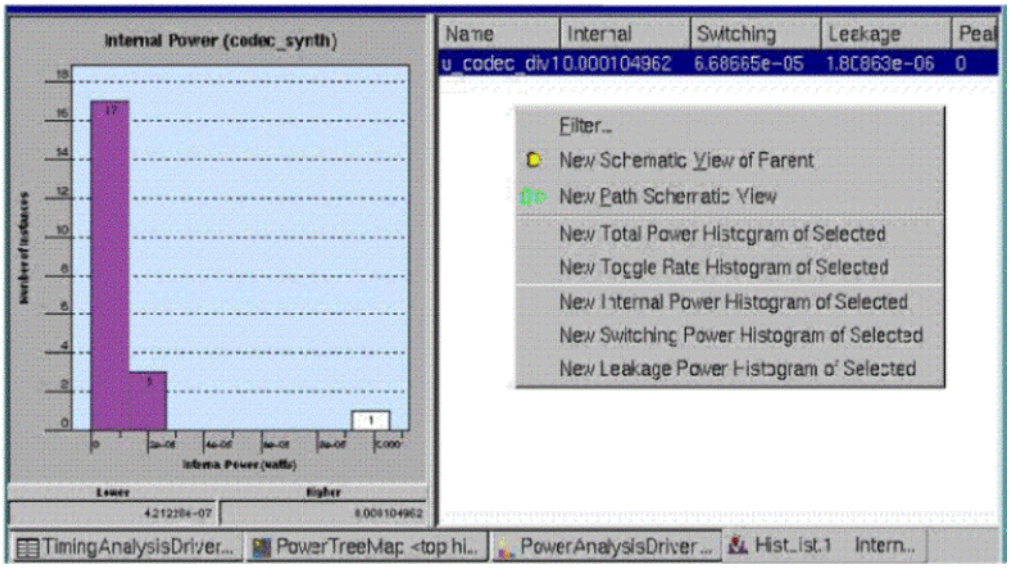
Histograms. A histogram is a graph of the frequency distribution for a class of data. In PrimeTime PX, the distribution can be plotted for power distribution, as shown in Figure 4-8, or for toggle rates, as shown in Figure 4-11.

A context-sensitive menu (not shown) provides quick access to histogram data depending on which segment of the power distribution bar chart you select. For example, selecting the internal-power bar shown in Figure 4-8 provides choices for viewing a histogram of the internal power as well as a total power or toggle histogram. These menus are also available for predefined groups.

In addition, click the Toggle Histogram button to view a toggle histogram for hierarchical cells in the design and the Total Power Histogram button to view a total power histogram for hierarchical cells in the design.

A context-sensitive menu accessible from within the histogram view is available when you select one or more bars on the histogram. As shown in Figure 4-10, this menu allows you to generate a further series of histograms based on the selected bars.

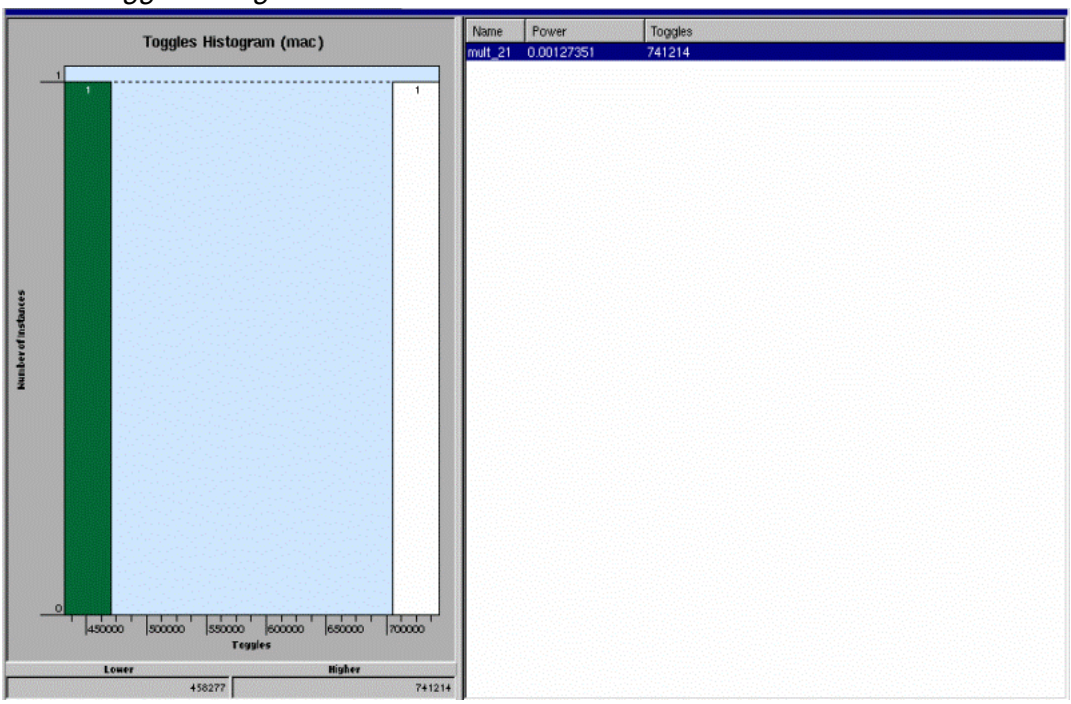
Figure 4-10 Histogram Context-Sensitive Menu



When you select a hierarchical item, the cells within the hierarchical cell are used to plot the next histogram using the hierarchical cell name in the title of the new histogram.

Figure 4-11 shows an example of a toggle rate histogram. Select multiple cells in the design hierarchy browser to view their histograms.

Figure 4-11 Toggle Histogram

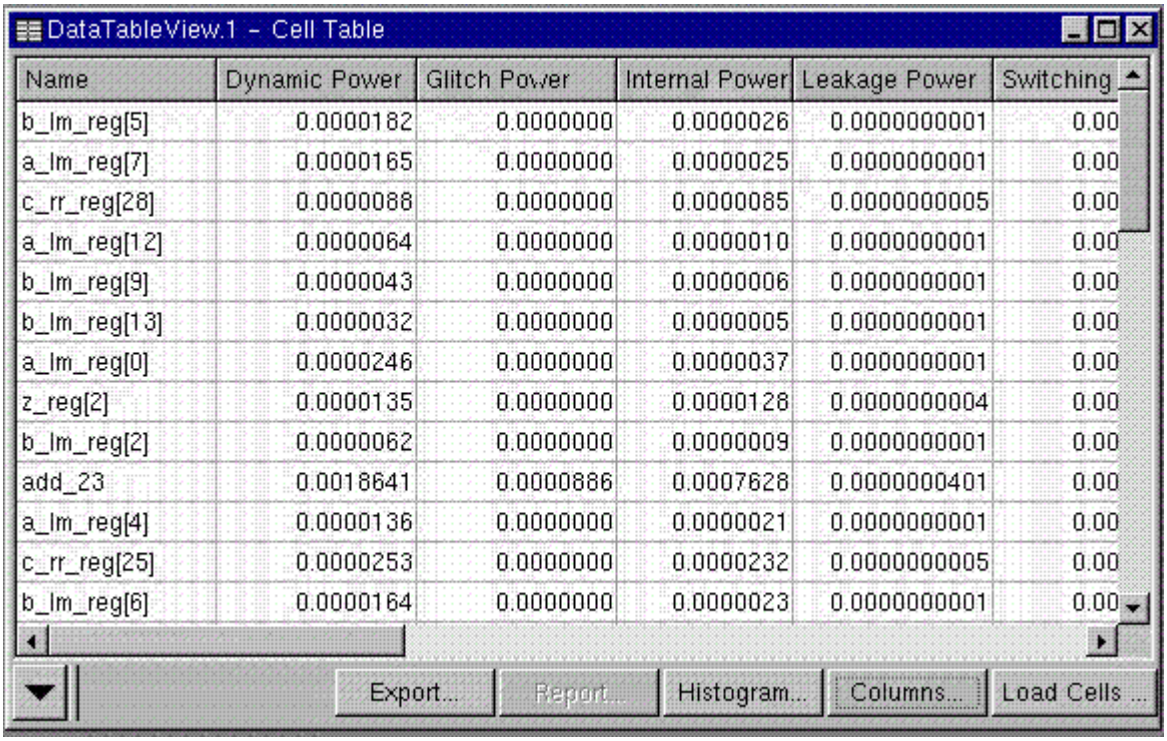


When you select a histogram bar, it turns yellow and its associated cells are displayed in the right pane. Given multiple cells listed on the right (not shown), you can select any combination of them to view their histograms.

Cell Data Table.To view the design's power attributes, you can create data tables for cells, pins or ports, and nets. To do this, choose Design > New Table View for Selection if you have already selected cells, pins or ports, or nets. Otherwise use Design > New Table View.

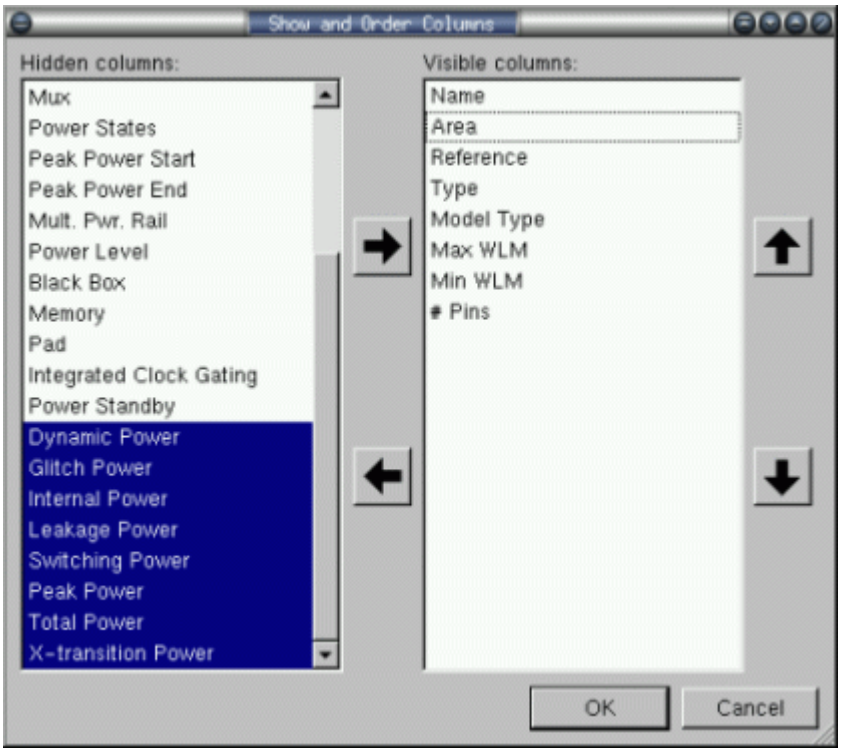
Figure 4-12 shows a cell data table.

Figure 4-12 Cell Data Table



By default, power attributes do not appear in the data tables. To view them as shown in Figure 4-12, click the Columns button. The Show and Order Columns dialog box opens, as shown in Figure 4-13. From here you can select the applicable power-related columns for visibility.

Figure 4-13 Show and Order Columns for Data Tables



Schematic Viewer. To view schematics, right-click the name of a cell (in the design hierarchy browser) to access a menu that allows you to display a path schematic of the cell either in its parent cell context or not. Within a schematic, you can right-click again to choose viewing options, report cell timing parameters, and highlight the multivoltage cells.

To view a net's capacitance, select the net and then choose Net Capacitance Profile from the Timing menu.

Waveform Viewer. To view your switching activity in a waveform format, choose Power > View Waveforms to open the nWave waveform viewer.

5

Averaged Power Analysis

This chapter describes how to annotate switching activity so that PrimeTime PX can perform averaged power analysis and explains how PrimeTime PX uses the information to calculate the averaged power and cycle-averaged power waveforms.

PrimeTime PX calculates averaged power based on toggle rates. The annotated activity can come from default toggle rates or switching activity specifications, or in the form of simulation-generated SAIF or VCD files. Zero-delay simulation is used to propagate switching activity to nonannotated nets for calculating the power consumption for the complete design.

This chapter contains the following sections:

- Power Calculation Variables

Annotating Switching Activity

The power consumption of a design depends on the switching activity of the nets and cell pins of the design. The higher the switching activity, the more power the design consumes. To calculate the averaged power, you must first annotate the design with the switching activity as described in this section.

The required switching activity information can be annotated on design objects such as nets, ports, and pins and then used by the `update_power` command during power analysis.

You can annotate the following types of switching activity on design objects:

- Simple switching activity on design nets, ports, and cell pins

  Simple switching activity consists of the static probability and the toggle rate. The static probability is the probability that the value of the design object has logic value 1. It defines the percentage of time the signal is at a high state. By default, the value is 0.5, which means the signal is high for half the time and low for half the time. The default values might be appropriate for data bus lines but might not be appropriate for some signals, such as `reset` or `test_enable` signals.

  The toggle rate is the rate at which the design object switches between logic values 0 and 1 within a time period.

- State-dependent toggle rates on input pins of leaf cells

  The internal power characterization of an input pin of a library cell can be state-dependent. The input pins of instances of such cells can be annotated with state-dependent toggle rates.

- State-dependent and path-dependent (SDPD) toggle rates on output pins of leaf cells

  The internal power characterization of output pins can be state-dependent, or path-dependent, or both. Output pins of cells with such characterization can be annotated with SDPD toggle rates.

- State-dependent static probability on leaf cells

  Cell leakage power can be characterized with state-dependent leakage power tables. Such cells can be annotated with state-dependent static probability.

Annotate the switching activity with one of the following methods. These are described in the sections that follow.

- Generate a VCD file using RTL or gate-level simulation and read the VCD file using the `read_vcd` command.
- Generate one or more SAIF files using RTL or gate-level simulation and read the SAIF files using the `read_saif` or `merge_saif` command.

- Use the `set_switching_activity` command on individual design objects.
- Use the default switching activity.

Deterministic and Implied Activity

The switching activity derived from the `create_clock` and `set_case_analysis` commands is separated from the switching activity derived from random simulation. The deterministic switching activity is propagated during the reporting of the switching activity, as in the `report_switching_activity` command. This enables you to see the deterministic propagation in advance.

PrimeTime PX also supports a type of activity source called implied activities which are activities on the nets that are derived from annotation of the deterministic propagation. For instance, if you set the switching activity on the Q pin of the sequential cell, activity that is on the Q bar pin can be implied activity based on the activity on the Q pin.

Annotating Switching Activity Using VCD Files

A VCD file records logic value changes in a design. You can provide either RTL or gate-level VCD and read it into PrimeTime PX for averaged power analysis.

For averaged power analysis, PrimeTime PX automatically derives switching activity from the VCD file and converts the data into toggle rates used to calculate the average power of every cell. For information about `read_vcd`, see Chapter 6, “Time-Based Power Analysis.”

The behavior of the `read_vcd` command in the averaged power analysis is as follows:

- It is not necessary to use the `read_vcd` command. Activity files can be read in SAIF format or using the `set_switching_activity` command. If you do not specify activity data, PrimeTime PX uses default values.
- Supports reading of multiple VCD files. Issues warning message when different values are specified with the `-time` option of the `read_vcd` command.
- The `read_vcd` command issues an information message about reading the file and annotating the toggle rates onto the design.
- The name mapping is enabled.

When you use `read_vcd` command for averaged power analysis, PrimeTime PX checks the percentage of nets that are annotated and extracts the toggle information for the nets available in the VCD file.

The `read_vcd` command reads the specified file and applies the activity data to the attributes on the design instances. Use the `report_switching_activity` or `get_switching_activity` command to view the activity data derived from the VCD file.

PrimeTime PX applies default annotation to primary ports that are not annotated. The tool uses zero-delay simulation to propagate the activity to the nonannotated nets in the design. In addition, an internal algorithm calculates the switching activity for nets that do not have annotated switching activity.

When the `read_vcd` command is successfully executed PrimeTime PX prints a summary that includes the number of nets in the design, the number of leaf cells in the design, and the number of leaf cells with switching activity annotation on all of its input pins. The following example is a summary of the `read_vcd` command.

```
pt_shell> read_vcd rtlvcd.dump -strip_path tb/mypath
checked out license 'PrimeTime-PX'

****************************************************************************
Summary:
Total number of nets = 1625
Number of annotated nets = 437 (26.89%)
Total number of leaf cells = 1339
Number of fully annotated leaf cells = 114 (8.51%)
****************************************************************************
```

The total number of nets shown in the example is the number of nets in the design. The nets that exist at several levels of design hierarchy are considered only once for the count.

The number of annotated nets is the number of nets for which switching activity is specified in the VCD file. The total number of leaf cells is the number of leaf cells in the design. The number of fully annotated leaf cells is the number of leaf cells in the design for which switching activity is annotated for all the pins.

Note:

The `read_saif` command also provides a summary with similar information of the nets and cells, of the design being read.

To ensure proper switching activity annotation, ensure that your name mapping is correct. For more information, see "Name Mapping" on page 3-5.

Annotating Switching Activity Using SAIF Files

The more accurate the switching activity, the more accurate the power analysis. Wherever possible, provide switching activity generated from simulation. You can provide either RTL or gate-level SAIF to PrimeTime PX for analysis.

Generating SAIF Files

You can generate SAIF files directly from HDL simulation or with the UNIX vcd2saif utility provided with PrimeTime PX.

HDL Simulation

The Synopsys VCS MX simulator is capable of generating SAIF files directly with built-in tasks. For other HDL simulators, the SAIF generation programmable language interface (PLI) must be linked, and the testbench must include the appropriate PLI commands for outputting the toggle information in the SAIF format.

For more information on SAIF generation from HDL simulation, see the *Power Compiler User Guide*.

vcd2saif Utility

Not all simulators write SAIF files. However, because many simulators support the VCD format, you can use the vcd2saif utility to convert both RTL and gate-level VCD data into the SAIF format. This utility supports Verilog, VHDL, and mixed-language VCD outputs from VCS, VCS MX, Verilog-XL, MTI ModelSim, and NC-SIM. The vcd2saif utility also supports the VHDL generate construct. The supported formats for input VCD files read into vcd2saif can be compressed into .gz, .Z, VPD, and .fsdb formats.

This utility can be used with or without a forward SAIF file.

The vcd2saif utility does not support SDPD toggle rates.

The following syntax for the vcd2saif utility can be used for RTL simulation (without a forward SAIF file) and gate-level simulation:

```
vcd2saif -i vcd_file -o bsaif
      [-instance path ...]
      [-format lang] [-testbench lang]
      [-verilog_instance path] [-vhdl_instance path]
      [-no_div] [-keep_leading_backslash]
      [-time number number]
```

Execute `vcd2saif -h` for the complete list of options.

Reading SAIF Files

Use the `read_saif` command to annotate switching activity from SAIF files for both RTL and gate-level SAIF files. The tool reads the VCD data and immediately annotates the switching activity information to the attributes of the design instances.

RTL SAIF

If no gate-level simulation data is available, provide RTL-level SAIF files that contain the switching activity of the primary inputs, hierarchical ports, and synthesis-invariant elements such as sequential elements.

For proper switching activity annotation, ensure that your name mapping is correct. For more information, see "Name Mapping" on page 3-5.

Gate-Level SAIF

The same `read_saif` command is used to parse the gate-level SAIF file, although you can apply different options. If the gate-level SAIF contains activity annotations for all elements in the design, PrimeTime PX can accurately calculate the average power consumption of the design.

SAIF File Commands

PrimeTime PX provides the following commands:

- `read_saif`
- `merge_saif`
- `write_saif`

Reading SAIF Files With read_saif

Within the SAIF file, the current design is instantiated as an instance in the testbench used to generate the SAIF file. Use the following command to read SAIF:

```
read_saif
      -strip_path prefix
      [-path prefix] [-ignore ignore_name]
      [-exclude exclude_filename]
      [-derate_glitch value]
      [-quiet] filename
```

See the man pages for option descriptions.

You can specify a gate-level or RTL SAIF file. The command reads the specified file and applies the activity data to the attributes on the design instances.

Apply the `-strip_path` option to specify the path to the current design instantiated in the simulator environment. For example, the following command reads in a SAIF file, my.saif, in which the current design is instantiated as TB/DUT in the SAIF file:

```
pt_shell> read_saif -strip_path TB/DUT my.saif
```

Reading SAIF Files With merge_saif

The `merge_saif` command can read switching activity information from multiple SAIF files. Input SAIF files are given individual weights, and a weighted sum of the switching activities is annotated. You can use this command for flows in which different SAIF files are generated for different modes of the same design. The switching activity from all the different modes can then be used for power calculations and optimization.

For example, assume that your design has three modes (standby, slow, and fast) and that the SAIF files standby.saif, slow.saif, and fast.saif are generated for these modes. Based on the expected usage of the design, you give the following weight to the SAIF files:

standby.saif 80%
slow.saif 5%
fast.saif 15%

You can then use the following command to read the SAIF files:

```
pt_shell> merge_saif -input_list \
              {-input standby.saif -weight 80 \
               -input slow.saif   -weight  5 \
               -input fast.saif   -weight 15 } \
              -strip_path tb/i
```

Use the `-output` option of the `merge_saif` command to generate a SAIF file containing the weighted sum of the switching activities.

After `merge_saif` reads each individual SAIF file, it uses a switching activity propagation mechanism to estimate the switching activity of design nets that are not included in the file. Therefore, you can use the following command to generate a gate-level SAIF file with estimated switching activity information from an RTL SAIF file:

```
pt_shell> merge_saif -input_list {-input rtl.saif \
           -weight 100} -strip_path tb/i \
           -output estimate.saif
```

Use the `-simple_merge` option to switch off the switching activity propagation mechanism when the information in the SAIF files is being merged.

Currently, SAIF files with SDPD information cannot be used by the `merge_saif` command.

By default, this command outputs all warnings.

The syntax of the `merge_saif` command is the same as that of the `read_saif` command, with the following exceptions:

- A weighted input file list is specified instead of a single input file.
- The `-simple_merge` and `-output` options can be used with the `merge_saif` command.

```
merge_saif
      -input_list saif_file_and_weight_list
      -strip_path prefix
      [-path prefix]
      [-ignore ignore_name]
      [-exclude exclude_filename]
      [-derate glitch_value]
      [-output merged_saif_name] -simple_merge]
      [-quiet]
```

See the man page for more details.

Writing out SAIF files in PrimeTime PX

PrimeTime PX can write out SAIF files containing user-annotated and propagated switching activity.

Use the following command to write the SAIF files:

```
write_saif [-cells cell_list]
      [-rtl]
      [-propagated]
      [-no_hierarchy]
      [-exclude_sdpd]
      [-derate_glitch value]
      file_name
```

See the man pages for option descriptions.

Annotating Switching Activity Using the set_switching_activity Command

Wherever possible, provide switching activity of the primary inputs that mimics the true behavior if you don't have simulation-generated toggle rate information.

Use the `set_switching_activity` command to annotate design objects with the different types of switching activity. You can annotate switching activity on nets, pins and ports with the `-toggle_count` and `-static_probability` options.

The toggle count represents the number of toggles within a given time period where the period can be specified with the `-period`, `-clock,` or `-base_clock` options. Internally, the toggle count is converted to an absolute toggle rate by dividing it by the period. If no period is specified, PrimeTime PX assumes that the toggle count is relative to the library time unit. For example,

```
pt_shell> set_switching_activity [get_net net1] \
      -static_probability 0.2 -toggle_count 10 -period 1000
```

This command specifies that the value of net1 is logic 1 for 20 percent of the time and that it transitions between logic values 0 and 1 an average of 10 times in 1,000 time units. The toggle rate applied to net1 is 10/1000, or 0.01.

The time unit used for the toggle rate is the target library time unit. The `-period` option is optional; a default value of 1 is used when this option is not specified. If the `-period` option was omitted in the previous example, the toggle rate for net1 would be 10/1, or 10.

Use the `-state_condition` option to annotate state-dependent toggle rates. For example,

```
pt_shell> set_switching_activity [get_pin ff1/Q] \
      -toggle_count 0.01 -state_condition "D"
pt_shell> set_switching_activity [get_pin ff1/Q] \
      -toggle_count 0.03 -state_condition "! D"
```

This command specifies that the pin ff1/Q toggles 0.01 times when the pin D is at logic 1 and 0.03 times when the pin D is at logic 0. You can use the `-rise_ratio` option with state-dependent toggle rates to specify the ratio of rise transitions to fall transitions for the specified state. For example,

```
pt_shell> set_switching_activity [get_pin xor1/Y] \
      -toggle_count 0.01 -state_condition "A" -rise_ratio 0.9
```

This command specifies that the pin xor1/Y toggles 0.01 times when the cell is in state A and that 90 percent of these toggles are rise transitions.

Use the `-path_sources` option to annotate path-dependent toggle rates. For example,

```
pt_shell> set_switching_activity [get_pin and1/Y] \
      -toggle_count 0.02 -path_sources "A"
pt_shell> set_switching_activity [get_pin and1/Y] \
      -toggle_count 0.00 -path_sources "B"
```

These commands specify that the pin and1/Y toggles 0.02 times due to a toggle on the input pin A but never toggles due to a toggle on B. You can specify toggle rates that are both state and path dependent by using the `-state_condition` and `-path_sources` options together.

Use the `-state_condition` option to annotate state-dependent static probabilities. For example,

```
pt_shell> set_switching_activity [get_cell and1] \
      -static 0.1 -state_condition "A & B"
pt_shell> set_switching_activity [get_cell and1] \
      -static 0.7 -state_condition "A & !B"
pt_shell> set_switching_activity [get_cell and1] \
      -static 0.2 -state_condition "! A"
```

These commands specify that the cell and1 is at state A & B for 10 percent of the time, at state A & !B for 70 percent of the time, and at state !A for 20 percent of the time.

You can also use the `-clock` and `-base_clock` options to specify the time period for the toggle count. The `-clock` option defines a period based on the object's `power_base_clock` period.

The `power_base_clock` attribute is set for all the nets in the design and references the clock domain to which they belong. If they belong to multiple clock domains, the `power_base_clock` attribute is set to the fastest of the clocks. The `-base_clock` option allows you to specify the period as that of the specified clock, regardless of the clock domain to which it belongs. It does not modify the `power_base_clock` attribute value; it simply controls the toggle rate applied to the node. For more information, see the man page.

You can apply appropriate switching activity to all primary inputs of the design by defining a toggle count relative to the clock period of the clock domain with which the inputs are associated.

To assign toggle rates that are based on the related clocks, apply the `-clock` option. In the following example,

```
pt_shell> set_switching_activity -type registers \
      toggle_count .1 -clock {clk1 clk2}
```

the command specifies that all registers in the design have a toggle rate of 0.1 relative to their related clock. If clk1 has a period of 25 ns and clk2 had a period of 10 ns, PrimeTime PX applies a toggle rate of .1/25, or .04, to registers in the clk1 domain and a toggle rate of .1/10, or .01, to registers in the clk2 domain.

Note:

When setting the toggle rate on registers and clock-gating cells, first apply the command to the registers, followed by the clock-gating cells because these are, in fact, registers as well.

The following syntax gives more information about the `set_switching_activity` command:

```
set_switching_activity
      [-state_condition state]
      [-toggle_count toggle_rate]
      [-period period_value]
      [-clock_derate value]
      [-glitch_count count]
      [-path_sources name_list]
      [-static_probability static_prob]
      [-rise_ratio rise_ratio]
      [-period period_value]
      [-base_clock clock]
      [-type object_type_list]
      [-no_hierarchy] [-quiet]
      [-clocks clocks] object_list
```

See the man pages for option descriptions.

Use the `report_switching_activity` command to view switching activity.

Removing Switching Activity Annotation

Use the `reset_switching_activity` command to remove switching activity annotation from individual design objects. For example,

```
pt_shell> reset_switching_activity objects
```

This command removes the simple and SDPD switching activity annotation from the specified objects.

When no options are provided to the `reset_switching_activity` command, it removes all previously annotated and propagated switching activity for the complete design hierarchy.

By default, this command outputs all warnings.

To selectively remove switching activity, apply the following `reset_switching_activity` command options:

```
reset_switching_activity
      [-state_condition state]
      [-path_sources path]
      [-toggle_rate]
      [-no_hierarchy] [-static_probability]
      [-quiet] [object_list]
```

See the man pages for option descriptions.

It is recommended that you remove switching activity information from previous switching activity annotation by using the `reset_switching_activity` command before reading new SAIF files.

Using the Default Switching Activity

If no simulation data is available, you can use the built-in default toggle rate to analyze the power consumption.

The default is specified with the following variables:

```
power_default_toggle_rate (default 0.1)
power_default_static_probability (default 0.5)
power_default_toggle_rate_reference_clock
           [fastest|related] (default related)
```

This toggle rate is assigned to all primary inputs, tristate outputs, and black box outputs and propagated throughout the rest of the design.

Reporting Switching Activity

The `report_switching_activity` command provides information about the various sources of switching activity data. The syntax is

```
report_switching_activity
      [-gate]
      [-rtl]
      [-cells cell_list]
      [-list_not_annotated]
      [-list_low_activity]
      [-list_by_source source_string]
      [-base_clock clock_name]
      [-average_activity]
      [-coverage]
      [-hierarchy]
      [-show_pin]
      [-sort_by hier | toggle]
      [-exclude string]
      [-include_only string]
      [-only_related_clock clock]
```

See the man pages for option descriptions. You can use this command after either the `read_saif` or `read_vcd` command.

As shown below, the default report shows the percentage of design objects annotated from the activity file, `set_switching_activity` command, `set_case_analysis` command, and `create_clock` command. In addition, it categorizes the nets driven by certain types of drivers.

Example 5-7 Example Report Generated by the report_switching_activity Command

```
pt_shell> report_switching_activity

report_switching_activity
****************************************
Report : Switching Activity

Design : test1
Version: B-2008.12
Date   : Tue Nov  4 15:29:05 2008
****************************************

Object Type       File (%)          SSA (%)      SCA (%)      Clock (%)    Default (%)
Propagated(%)   Implied(%)      Annotated(%)    Total
--------------------------------------------------------------------------------------------
--------------------------------------------
Nets             3489(100.0%)      0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)
0(0.00%)         0(0.00%)         0(0.00%)         3489
--------------------------------------------------------------------------------------------
--------------------------------------------
Nets Driven by
--------------------------------------------------------------------------------------------
--------------------------------------------
Primary Input     150(100.0%)       0(0.00%)     0(0.00%)     0(0.00%)     0(0%)
0(0.00%)         0(0.00%)         0(0.00%)         150
Tri-State         300(100.0%)       0(0%)        0(0%)        0(0%)        0(0%)
0(0%)            0(0%)            0(0%)            300
Black Box         80(100.0%)        0(0%)        0(0%)        0(0%)        0(0%)
0(0%)            0(0%)            0(0%)            80
Sequential        956(100.0%)       0(0%)        0(0%)        0(0%)        0(0%)
0(0%)            0(0%)            0(0%)            956
Combinational     2012(100.0%)      0(0.00%)     0(0.00%)     0(0.00%)     0(0.00%)
0(0.00%)         0(0.00%)         0(0.00%)         2012
Memory              0(0%)            0(0%)        0(0%)        0(0%)        0(0%)
0(0%)            0(0%)            0(0%)            0
Clock Gate          0(0%)            0(0%)        0(0%)        0(0%)        0(0%)
0(0%)            0(0%)            0(0%)            0

--------------------------------------------------------------------------------------------
--------------------------------------------
1
```

Each net is counted once even if some nets are driven by multiple drivers. For nets with multiple drivers, the nets are assigned to the highest-priority driver as follows: primary input, black box, tristate, sequential, combinational.

Use the `-list_not_annotated` option to display the design objects that do not have user-annotated switching activity information.

Use the following attributes to get the value of toggle rate and signal probability:

```
toggle_count
toggle_rate
glitch_rate
static_probability
```

Debugging the Annotation

PrimeTime PX provides several commands to help you debug activity annotation and propagation. These commands return average activity data that can help you pinpoint problem areas in your design. In addition, you can use this information to improve the quality of your testbenches through analysis of the actual data against the activity you expect to see.

Use the following PrimeTime PX commands to view the annotated information used for power analysis:

- `report_switching_activity`
- `get_switching_activity`

The `set_switching_activity` command and user-provided SAIF/VCD files are parsed as soon as these commands are issued. You can query the annotation (but not the net propagation until after `update_power`) immediately after specifying the `read_saif`, `read_vcd`, and `set_switching_activity` commands.

Specifically, the following `report_switching_activity` options provide data that can help you analyze the annotation:

`-average_activity`

Reports the average switching activity for the design or a specified block.

`-hierarchy`

Reports the average switching activity for the top-level design and its subhierarchies for average activity, coverage, and default reports.

`-sort_by hier | toggle | name`

Used with the `-hierarchy` and `-average_activity` options, sorts the report by the hierarchy (the default) or the toggle rate. Use `toggle` and `name` to sort switching activity reports as well as list reports.

`-coverage`

Reports the percentage of nets that are covered, where covered indicates nets that toggle, by default, one or more times. When used with the `-hierarchy` option, reports coverage hierarchically. As needed, use the `-toggle_limit` option to change the toggle limit from the default 1.

Use the `-exclude` and `-include_only` options to calculate the average switching activity or coverage percentage after filtering for the specified design parts. These options allow you to filter for types of nets (that is, primary inputs) or nets that were annotated from a file. You can specify net types or the annotation type to isolate possible issues. For more information, see the man page.

Suppose you suspect that propagation unsuccessfully propagated low-activity rates through a series of sequential elements, which results in zero activity at the output of some sequential elements. The following series of commands helps you verify your problem:

```
#For propagation debugging, you must first
#specify the update_power command
update_power
#To provide information about nets driven by sequential
#cells that have zero toggles
report_switching_activity -coverage \
      -include_only sequential
report_switching_activity -list_zero_activity \
      -include_only sequential
```

To determine average switching activity at the output of sequential elements with non-zero switching activities, use the following commands:

```
update_power
report_switching_activity -average_activity \
      -include_only sequential -exclude low_activity
```

As shown below, the average activity report includes both toggle rates and the toggle counts. The toggle rates are calculated with respect to the primary clock, which you can designate with the `-primary_clock` option.

```
pt_shell> report_switching_activity -average_activity
              -hierarchy

****************************************
Report : Switching Activity
-average_activity -hierarchy
Design : counter
Version: V-2009.06
Date :   Thu May 14 11:09:34 2009
****************************************

Switching Activity Statistics for "counter"
--------------------------------------------------------------------------
Switching Activities per clock is with respect to clock 'CLK'
Simulation time 10ns (1000 clock periods of clock 'CLK')
--------------------------------------------------------------------------
                  Toggle Count  Toggle Rate   Glitch Count    Glitch Rate
Object            Per Net       Per Clock     Per Net         Per Clock
--------------------------------------------------------------------------
counter              200           0.2           1              0.001
counter_low_bits     150           0.15          1              0.001
counter_high_bits    50            0.05          0              0
reset_block          1             0.001         0              0
overflow_detect      1             0.001         0              0
--------------------------------------------------------------------------
```

For more information about the `report_switching_activity` options, see the man page.

Use the following `get_switching_activity` option to help you analyze the annotation:

`-toggle_rate`

Returns the toggle rate along with the information source. For example:

```
pt_shell> get_switching_activity \
      -toggle_rate smas_ira_read_data[29]
{"smas_ira_read_data[29]" 0.000017}
```

Estimating Nonannotated Switching Activity

PrimeTime PX needs switching activity information on all design nets and SDPD information on all design cells and pins in order to calculate power. Switching activity that is not user-annotated is estimated automatically before power is calculated.

This estimation is performed in three stages:

- Design nets whose switching activity can be calculated accurately or cannot be propagated are set to default values. Such nets are said to be default-annotated.
- The user-annotated and default annotated switching activities are then propagated to derive the simple static probability and toggle rate information for the rest of the design nets.

- The simple switching activity information (user-annotated or estimated) is then used to derive the nonannotated SDPD switching activity.

Annotating Design Nets With Default Switching Activity Values

Design nets are annotated with default switching activity values when the switching activity can be accurately derived or when the switching activity cannot be estimated using the propagation mechanism described in this section.

The first type of nets includes nets driven by clocks, where the switching activity information can be accurately derived from the clock waveform. For the second type of nets, the propagation mechanism uses the functionality of design cells to propagate the input switching activity to the cell outputs. Black box cells do not have functional descriptions in the technology library; therefore, the switching activity of black box outputs cannot be derived using the propagation mechanism. Black box outputs that are not user-annotated are annotated with a default value.

The following types of design nets are annotated by default values:

- Nets driven by constants: A default toggle rate value of 0.0 is used. A static probability value of 0.0 is used for logic 0 constants, and a value of 1.0 is used for logic 1 constants.
- Nets driven by clocks: The default values for the toggle rate and static probability are derived from the clock waveform.
- Nets driving or driven by buffers: If the buffer input or output net is user- or default-annotated, then the nonannotated buffer output or input is default-annotated with the switching activity values on the annotated input or output.
- Nets driving or driven by inverters: If the inverter input or output net is user- or default-annotated, then the nonannotated inverter output or input is default-annotated with the same toggle rate value and with the inverted static probability value. If the annotated static probability value is sp, the inverted static probability value is 1.0 – sp.
- Flip-flop outputs: If a flip-flop cell has both Q and QN output ports and only one of the outputs is annotated, then the other output is default-annotated with the same toggle rate value and with the inverted static probability value.
- Primary inputs and outputs of black box cells: The switching activity of primary inputs and black box outputs cannot be propagated. Default switching activity based on the value of the `power_default_static_probability` and `power_default_toggle_rate` variables is used.

  The default static probability value is the value of the `power_default_static_probability` variable. The default toggle rate value is the value of the `power_default_toggle_rate` multiplied by the related clock frequency. The

related clock can be specified using the `-clocks` option of the `set_switching_activity` command. The default static probability is 0.5 and the default toggle rate is 0.1, unless you change the values with the aforementioned variables.

Without Clock Specification

If no clocks have been defined, the default clock period is assumed to be the default library time unit (generally 1 ns). Therefore, specify the frequency of operation by defining the clocks with the `create_clock` command. If no clock exists, you can define virtual clocks with this command as well.

With Clock Specification

By default, the `power_default_toggle_rate_reference_clock` variable is set to `related`. In this case, PrimeTime PX locates the related clocks for these ports and assigns a toggle rate relative to the appropriate clock. For example,

- Primary input D1 feeds a flip-flop U1 clocked by clk1 (period 10)
- Primary input D2 feeds a flip-flop U2 clocked by clk2 (period 20)

The switching activity for D1 and D2 varies because the related clocks differ. The clock-based toggle rate is calculated as follows:

toggle_rate = clock_frequency * `power_default_toggle_rate`

If the default, 0.5, is assigned to `power_default_toggle_rate`, then

- The toggle rate for D1 = (1.0/10) * 0.5 = 0.05
- The toggle rate for D2 = (1.0 /20) * 0.5 = 0.025

If `power_default_toggle_rate` is set to 0.1, then

- The toggle rate for D1 = (1.0/10) * 0.1 = 0.01
- The toggle rate for D2 = (1.0/20) * 0.1 = 0.005

To determine the related clock, PrimeTime PX checks the SDC constraints. If input transitions are provided relative to a clock, the tool assumes that the specified clock is the related clock. Otherwise, PrimeTime PX traces the fanout of the port to a point where the clock is bound. If more than one clock is found, the fastest of the related clocks is used.

If no related clock is found, the clock with the fastest design is used to determine the switching activity. You can also set the `power_default_toggle_rate_reference_clock` variable to `fastest`, which causes the fastest design clock to determine the toggle rate of the starting point ports or pins.

In this example, both D1 and D2 are assigned the same default toggle rate based on the fastest clock, clk1. Therefore,

- The toggle rate for D1 = (1.0/10) * 0.1 = 0.01
- The toggle rate for D2 = (1.0/10) * 0.1 = 0.01

The `power_base_clock` attribute is placed on the appropriate nets and ports. In the previous example, the value of `power_base_clock` on D1 is clk1 and on D2 is clk2.

Propagating Switching Activity

The switching activity of design nets that are not user-annotated or default annotated is derived using a propagation mechanism. This mechanism is basically a zero delay simulator. Random simulation vectors are generated for the user- and default-annotated nets, depending on the annotated toggle rate and static probability values. The zero-delay simulator uses the functionality of the design cells and the random vectors to obtain the switching activity on nonannotated cell outputs.

User- and default-annotated switching activity values are never overwritten by values derived by the propagation mechanism.

Deriving SDPD Switching Activity

If an RTL SAIF file or a gate-level SAIF file without SDPD information is used to annotate the design switching activity, then PrimeTime PX needs to estimate the required SDPD switching activity information. After obtaining the simple switching activity, PrimeTime PX estimates the state-dependent static probability information for every cell and the SDPD toggle rate information for every cell pin. This information is obtained from the switching activities of each cell input and output pin. Although the SDPD estimation mechanism produces fairly accurate power calculations, use gate-level SAIF files with SDPD information for the most accurate power results.

Based on the activity on the outputs and inputs of every leaf cell, PrimeTime PX calculates the Boolean difference for each power arc to determine the contribution of each power arc. The power arcs are obtained from the library power table.

Correlation

While propagating switching activity through the design, PrimeTime PX makes certain statistical assumptions. However, the logic states of gate inputs can have interdependencies that affect the accuracy of any statistical model.

Such interdependency of inputs is called correlation. Correlation affects the accuracy of PrimeTime PX propagation of toggle rates. Because accurate analysis depends on accurate toggle rates, correlation also affects the accuracy of power analysis.

PrimeTime PX considers correlation within combinational and sequential logic, resulting in more accurate analysis of switching activity for many types of designs. The types of circuits that exhibit high internal correlation are designs with reconvergent fanouts, multipliers, and parity trees. However, PrimeTime PX has no access to information about correlation external to the design. If correlation exists between the primary inputs of the design, PrimeTime PX does not recognize the correlation.

Annotating Internal and Leakage Power on Black Box Cells

As needed, use the `set_annotated_power` command to annotate unresolved black boxes with internal and leakage power values (in watts). The syntax is

```
set_annotated_power
      [-internal_power int]
      [-leakage_power leak]
      cell_list
```

Although this command is normally used for black box cells for which the internal and leakage power cannot be estimated, you can also specify leaf cells. Annotated internal and leakage power values override any internally estimated internal and leakage values.

Remove the annotated values with the `remove_annotated_power` command. If available, internally estimated values are reassociated with the specified cells.

See the man pages for details about these commands.

In the cell power report, cells with annotated powers are displayed together with cells whose powers are internally calculated, but they are labeled to show that their powers are annotated. Annotated power overrides estimated power in the report.

Use the `report_annotated_power` command to view a summary report of annotated power values. For example,

```
pt_shell> report_annotated_power -list_annotated

****************************************
Report: annotated power -list annotated
Design: mac
Version: Y-2007.06
Date: Thu Jan 19 15:43:48 2007
****************************************

Annotated powers:
```

```
-----------------------------------------
1. m1_r_reg[15](internal: 0.498leakage: 0.0037]
2. m1_r_reg[14](internal: 0.498leakage: 0.0037]
3. m1_r_reg[13](internal: 0.498leakage: 0.0037]
4. m1_r_reg[12](internal: 0.498leakage: 0.0037]

                            |         |             |    NOT     |
    Cell type               | Total   | Annotated   | Annotated  |
----------------------------+---------+-------------+------------+
unresolved black-box cell   |     5   |         4   |         1  |
leaf cell                   |  7482   |         0   |      7842  |
----------------------------+---------+-------------+------------+
                            |  7487   |         4   |      7843  |
```

Annotating Switching Activity on Power Rails

When you enable the concurrent power analysis mode by setting the `power_enable_multi_rail_analysis` variable to `true`, you can use the `set_annotated_power` command with the `-rails` option to specify annotated power values on design objects of multiple power rails or power supply nets in the UPF mode. The value that you specify with the `-rails` option is applied to each rail in the list. During power analysis, the `update_power` command uses these values along with the calculated power values and generates the power data for the specified power rails or power supply nets in the UPF mode.

If you do not use the `-rails` option of the `set_annotated_power` command, the value that you specified is evenly divided for each valid power rail of the design object.

As shown in Example 5-8, the cell U1 has rail connections of VDDA, VDDB, and VDDC. The first `set_annotated_power` command annotates the internal power to 0.1 and leakage power to 0.0 for the cell U1, on rail VDDA. The second `set_annotated_power` command annotates the internal power to 0.2 and leakage power to 0.1 for cell U1, on both VDDB and VDDC power rails. The total annotated internal power on cell U1 is 0.5 and the total annotated leakage power on cell U1 is 0.2.

Example 5-8 Setting the annotated power on multiple rails

```
set_annotated_power -internal 0.1 -leakage 0.0 -rails VDDA U1
set_annotated_power -internal 0.2 -leakage 0.1 -rails {VDDB VDDC} U1
```

Example 5-9 shows rail independent values applied to cell U1. The `set_annotated_power` command annotates internal power of 0.3 and leakage power of 0.0 on cell U1. During power analysis, these rail independent annotated power values are divided equally over all the associated power rails of the cell. When you run the `report_power -rails VDDA -cell U1` command, the tool reports the annotated internal power of 0.1.

The `report_power` and the `report_annotated_power` commands support the `-rails` option. When you specify the list of rails with the `-rails` option, the tool reports the rail based information and statistics for the annotated power. To report the rail information for all the rails, specify `report_annotated_power -rails all`.

In Example 5-9, the `report_power -rails VDDA -cell U1` command reports the annotated power of 0.1 for the cell. The rail based annotated power values set by using the `set_annotated_power -rails` command in PrimeTime PX are used in the rail analysis flow in PrimeRail.

Example 5-9 Setting and reporting the annotated power on power rail

```
set_annotated_power -internal 0.3 -leakage 0.0 U1
report_power -rails VDDA -cell U1
```

Example 5-10 shows the report generated by the `report_annotated_power` command using the `-rails` and `-list_annotated` options.

Example 5-10 Report generated by the report_annotated_power -rails command

```
****************************************
Report : annotated_power -rails -list_annotated
Design : testcase
Version: D-2010.06
Date   : Fri May 14 17:40:08 2010
****************************************

Annotated cell powers:
------------------------------
1.  U0(VDD1)  (internal: 1e-07  leakage: 1e-11)
2.  U0(VDD2)  (internal: 2e-07  leakage: 2e-11)

                            |         |            |    NOT     |
 Cell type                  | Total   | Annotated  | Annotated  |
-----------+--------------+---------+------------+------------+
unresolved black-box cell | 0       | 0          | 0          |
leaf cell                 |    14   | 1          | 13         |
```

Checking for Potential Errors that Might Affect Power Accuracy

Before calculating power or reporting power, use the `check_power` command to identify potential power calculation problems that might affect your results. The syntax is

```
check_power
      [-verbose]
      [-significant_digits digits]
      [-override_defaults check_list]
      [-include check_list]
      [-exclude check_list]
```

This command checks the design structure for potential power violations.

By default, the `check_power` command performs the checks defined by the `power_check_defaults` variable if you don't specify any options. To update your list of default checks, either update this variable or use the `-override_defaults` option.

If specified, the `-override_defaults`, `-include`, and `-exclude` checklists define the power checks to be used. The `check_power` command can perform out_of_table_range, missing_table, no_arrival_window, no_base_clock, and missing_function checks. By default, PrimeTime PX performs the out_of_table_range and missing_table checks.

For further details about `check_power` and `power_check_defaults`, see the man pages. The man pages also describe the power checks.

Specifying the Options for Power Analysis

After you select the power analysis mode, you can specify options for the power analysis mode using the `set_power_anaysis_options` command. Power analysis is performed during the `update_power` command, considering the options you specified using the `set_power_analysis_options` command. Using this command without specifying any option resets all the current settings; default values of all the options will be considered for the power analysis. You use the `report_power_analysis_options` command to see the value set for the various power analysis options. This command reports both default as well as the values you set.

For more details about this command, see the command man page.

Power Analysis with Different Clock Frequencies

PrimeTime PX supports scaling the average power consumption as per the clock definitions in the SDC file. With this support, PrimeTime PX is capable of performing power analysis for a specific clock speed as well as calculating the power at various frequencies. To enable this feature set the `power_enable_clock_scaling` variable to `true`. The default value of this variable is `false`. Use the `set_power_clock_scaling` command to specify the simulation clock used in the VCD or SAIF file generation and to specify the period or scaling ratio to be applied. The tool does not scale the switching activities that are set using the `set_switching_activity` and `set_case_analysis` commands. Example 5-11 shows a sample script using this feature.

Example 5-11 A Sample Script Using the Clock Definitions from SDC File

```
set power_enable_analysis true
set power_analysis_mode averaged
set search_path         "../src/hdl/gate ../src/lib/snps . "
```

```
set link_library        " * core_typ.db"
read_verilog            mac.vg
current_design          mac
link
read_sdc ../src/hdl/gate/mac.sdc
read_parasitics ../src/annotate/mac.spef.gz
read_saif "../sim/mac.saif" -strip_path "tb/macinst"

set_power_clock_scaling -period 24 [get_clock clk]
set power_enable_clock_scaling true
update_power
report_power
quit
```

Power Calculation Variables

The `power_limit_extrapolation_range` variable controls some aspects of the power calculation for both event- and toggle-based analysis. The default value of this variable is `false`.

By default, PrimeTime PX extrapolates indefinitely if the data point for internal power lookup is out of range. If however, the `power_limit_extrapolation_range` variable is set to `true`, PrimeTime PX limits the extrapolation. The NLPM power table stops extrapolation at one additional index grid. The SPPM power table stops extrapolation at the specified range.

If there are many high-fanout nets (such as clock or global reset nets), you might want to limit the extrapolation for more accurate power values.

PrimeTime PX calculates the average power for the entire SAIF file or until there is convergence when no SAIF data is provided.

6

Time-Based Power Analysis

This chapter describes the commands and user options for performing time-based power analysis with PrimeTime PX.

For PrimeTime PX to perform time-based power analysis, you set the `power_analysis_mode` variable to `time_based`. To specify additional options for the analysis, you can use the `set_power_analysis_option` command.

Traditional time-based power analysis relies on gate-level simulation activity to perform peak power calculation. Detailed time-based power information is generated, enabling you to determine both average and peak power. An analysis technique called cycle-accurate peak power analysis allows you to use an RTL netlist as well as RTL VCD files to determine the peak cycle.

This chapter contains the following sections:

Specifying the Activity File

In the time-based power analysis mode you must specify an activity file in the VCD format. You can use either an RTL VCD file or gate-level VCD files to specify the activity.

PrimeTime PX processes activity files generated from simulation. When you use the `read_vcd` command, PrimeTime PX checks for the VCD files and verifies that the nets in the design are identified in that file.

The `read_vcd` command reports a summary of the percentage of nets covered by VCD annotation after reading the VCD file. The tool extracts the toggle rates and the static probability for the identified nets, and reports the switching activity with the `report_switching_activity` command.

During the execution of the `read_vcd` command the tool reads the header information in the VCD file to determine the mapping between the VCD signals and the netlist objects. However, the processing of the events in the VCD file, such as determining the related pins and tracking the SDPD toggles is performed during the execution of the `update_power` command.

The syntax of the `read_vcd` command is as follows:

```
read_vcd
      [-pipe_exec exec_command]
      [-path path] [-strip_path prefix]
      [-strip_path strip_path]
      [-zero_delay]
      [-rtl]
      [-format format]
      [-time time_list]
      filename
```

In the time-based power analysis mode, the tool does not support multiple VCD files. If you use multiple `read_vcd` commands, PrimeTime PX issues a warning message and uses the activity information from the last `read_vcd` command specified before the `update_power` command.

Using RTL VCD Files

You must use the `-rtl` option of the `read_vcd` command when you specify the activity file in the RTL VCD format. When you use the `-rtl` option, PrimeTime PX enables name mapping and event propagation and performs cycle accurate peak power analysis on the design. For more information, see “Cycle-Accurate Peak Power Analysis” on page 6-4.

Using Gate-Level VCD Files

In the time-based power analysis mode, when you do not specify either the `-rtl` or the `-zero_delay` options of the `read_vcd` command, PrimeTime PX assumes that the activity file is a gate-level VCD file. For gate-level VCD files, the tool does not perform activity propagation and name mapping.

In the time-based mode, the `read_vcd` command without the `-rtl` option issues a warning message if less than 95% of the nets in the VCD files are annotated.

Note:

Perform the required type of power analysis regardless of the percentage of annotation. Also, do not run the vcd2saif utility internally to obtain toggle rates and static probability before the power analysis. After the execution of the `update_power` command, use the `report_switching_activity` command to see the toggle rates obtained from processing the events in the VCD file.

Using Zero-Delay VCD files

If the VCD file is generated from a zero-delay simulation, you need to provide SDC constraints as well as specify the `-zero_delay` option with the `read_vcd` command to ensure accurate calculation of the peak power and waveform data.

This option indicates that the VCD file does not contain SDF delay annotation information and, therefore, PrimeTime PX cannot isolate the exact time of the events to produce an instantaneous power profile. In this case, PrimeTime PX performs cycle-accurate peak power analysis on the design. For more information, see “Cycle-Accurate Peak Power Analysis” on page 6-4.

Cycle-Accurate Peak Power Analysis

Cycle-accurate peak power analysis is a technique for calculating power that allows you to use a RTL VCD file to obtain the overall power profile for your design for different time periods and operating modes. You can also generate the waveforms.

Cycle-accurate peak power analysis generates peak power with time resolution at the cycle level. PrimeTime PX reports the peak power as the maximum cycle power along with the time when the maximum peak power occurred in the RTL VCD file. Cycle-accurate peak power analysis does not require precise time-based events at every net but only at nets that are synthesis invariant.

The tool propagates the events to specific blocks that you specify with the `-cells` option of the `set_power_analysis_options` command and to specific time windows that you specify with the `-time` option of the `read_vcd` command. However, if you do not specify these options, the tool propagates the events to all the blocks and the entire simulation time window.

Example 6-1 is a script that can be used to perform cycle-accurate peak power analysis using the RTL VCD file.

Example 6-1 Cycle-Accurate Peak Power Analysis Using RTL VCD File

```
set power_enable_analysis true
set power_analysis_mode time_based

# read in designs and libraries, link,
# run timing analysis, and source the name-mapping
# file (if necessary)

read_vcd -rtl yourRTL.vcd -time
set_power_analysis_options -cells
report_power
```

Use the `missing_function` check of the `check_power` command to search the design for blocks that do not have a logic function, which can occur when a block does not have a logic model. This check can be important for cycle-accurate peak power analysis because the analysis relies on the ability to propagate events through combinational logic, which requires the blocks to have logic models. For more information about `check_power` command, see the man page.

Because the RTL VCD files do not contain events for combinational gates, the resulting waveforms use time resolution at the cycle level rather than instantaneous time resolution. Each cycle shown in the waveform stems from an independent power analysis.

Specifying Activity With the set_case_analysis Command

When the RTL VCD file does not contain activities for certain nets, you can set the activities on the nets, by using the `set_case_analysis` command. By using the `set_case_analysis` command you can accurately calculate power during the cycle-accurate peak power analysis. During time-based power analysis, the cycle-accurate peak power simulation takes the values set by the `set_case_analysis` command and accurately calculates the peak power. To set a signal value for time-based or averaged mode the order of preference should be:

- Use the VCD or Switching Activity Interchange Format (SAIF) files
- Use the `set_case_analysis` command
- Connect to a logical constant net

For both time-based and averaged mode, the report includes the source of the switching activity.

After the power calculation a warning message is issued when you set a signal using more than one source and the settings conflict with each other. For example, the conflict can arise between the value in the VCD file against a constant net or with the value set using the `set_case_analysis` command.

Activity File Formats

The PrimeTime PX analysis engine processes VCD activity files. However, you can input other activity file formats that are converted within PrimeTime PX to VCD with the appropriate utility. To specify the format, PrimeTime PX relies on the file extension. It then applies the corresponding conversion utility, which must be in the search path. For VPD files, PrimeTime PX uses the utility located at $VCS_HOME/bin/vpd2vcd if you have set the `VCS_HOME` variable appropriately.

The following table lists the formats, extensions, and conversion utilities supported by PrimeTime PX.

File type	Extension	Utility
VCD	Default file type	None
VPD	.vpd	$VCS_HOME/bin/vpd2vcd
Compressed VCD	.Z	uncompress
Gzipped VCD	.gz	gunzip
FSDB	.fsdb	$SYNOPSYS/bin/fsdb2vcd
Bzip2	.bz2	bunzip2

Note:

If the file extension is not one of those listed, PrimeTime PX treats the file as a VCD file and reads the data directly.

Mapping the Testbench Instance to the Design Module

The activity file is generally created at the testbench level, where the design under test is an instance within the HDL testbench hierarchy. To map the testbench instance to the design module specified as the current design, apply the `-strip_path` option of the `read_vcd` command.

For example, if the testbench TB instantiates the design module TOP as U, you would specify the `-strip_path` option as follows:

```
pt_shell> read_vcd -strip_path TB/U1 ...
```

The `report_vcd_hierarchy` command reports the hierarchy structure in your VCD file, which can help you decide on the correct strip path to specify in the `read_vcd` command.

Note:

Even if the top-level design was not specified in the $dumpvars task, the VCD file still lists the complete hierarchy starting with the testbench.

Handling Large Activity Files

Several choices need to be made regarding the handling of large amounts of activity data. Although PrimeTime PX supports the processing of VCD files larger than two gigabytes on all platforms, you might want to avoid generating and storing large files.

You can generate compressed formats such as FSDB or gzipped VCD to reduce the activity file size. Within PrimeTime PX, the data from the compressed files is converted to the VCD format and piped to the analysis engine. The converted data is not stored.

Another option is to pipe the activity data directly from the HDL simulation to PrimeTime PX. In this case, the activity data is fed directly from the HDL task within the simulation to the power analysis engine instead of being saved into a file. To do direct piping, follow these steps:

1. The HDL simulation must be set up to generate an activity file. For example, if you are generating a large VCD file from a Verilog simulation, the testbench must include the $dumpvars or $dumpfile tasks to generate the VCD data. Use the same file name in both $dumpfile and `read_vcd`.

   This file name is used as a pipe name, so both tools know where to write and read the data. The process does not create a large data file on disk.

2. Specify the commands or run a script to invoke the HDL simulation with the `-pipe_exec` option. For example, you might have a run script, say run_vcs that executes a VCS simulation that contains the following command:

```
vcs -R -f arguments -l log
```

You can invoke the simulation within PrimeTime PX to pipe the VCD data directly by adding either of the following options to the `read_vcd -pipe_exec` command:

```
-pipe_exec run_vcs
```

or

```
-pipe_exec "vcs -R -f arguments -l log" ...
```

Handling VCD Files With Nonmodule Scopes

Some VCD files contain scopes such as tasks and functions. The VCD file parser does not distinguish between the VCD hierarchy defined by functions and tasks and the VCD hierarchy defined by modules.

Modeling Special Conditions

The power consumption for the conditions listed below is considered part of the dynamic switching power. It affects both internal power and switching power.

- Glitches
- Z states
- X states

Glitch Power

To calculate glitch power, PrimeTime PX determines whether the pulse is small enough to be considered a glitch and then calculates the power based on the following formula:

Glitch power = Ratio * (dynamic power when there is no glitch)

where

Ratio = $[(\text{pulse\_width} * 2) / (\text{rise\_ramp} + \text{fall\_ramp})]^2$ when pulse width < (rise_ramp + fall_ramp).

You can obtain the pulse width from the simulation, and you can obtain the rise/fall transition times by executing the `report_power_calculation` command on the desired instance.

Z State

Transitions through the Z state are assumed to consume no power. PrimeTime PX tracks transitions, and if a transition after the Z state differs from the original transition, it is considered to be one full transition.

For example, 0 -> Z -> 1 or 1 -> Z -> 0 is considered 1 transition; however, 1 -> Z -> 1 or 0 -> 0 is not a transition and consumes no power.

X State

By default, every time a net transitions to or from the X state, it is considered as 1/2 of a power transition of a normal transition.

For example, 0 -> X is a 1/2 transition and 0 -> X -> 0 or 0 -> X -> 1 is 1 transition.

This is because the default value of the controlling variable, `power_x_transition_derate_factor`, is `0.5`. To change the scale factor, modify the value of the variable. Setting this variable to 1 results in pessimistic power reports.

The `power_match_state_for_logic_x` variable determines how PrimeTime PX interprets X state in the Boolean function. The default value of this variable is 0. Hence, PrimeTime PX, by default, considers X state as logic 0.

Debugging Problems With the VCD File

The following errors can occur during the execution of the `read_vcd`, `update_power` commands:

1. The VCD hierarchy does not match that of the current design.

   In this case, the total number of annotated nets and cells in the `read_vcd` summary report is zero. PrimeTime PX issues warnings for all the nets in the design that cannot be found in the VCD file.

   The mismatch generally occurs when the `read_vcd -strip_path` option is used. Either the `-strip_path` option was incorrectly specified or the VCD file does not contain all levels of design hierarchy.

2. Syntax errors occur during parsing of the activity file.

   Problems with parsing the activity file can have several causes:

- The file format might not be one of those listed. If an enhanced VCD (EVCD) file is supplied, PrimeTime PX generates syntax error messages. PrimeTime PX does not support the enhanced VCD file format.
- The conversion utility might not match the activity file format type or version. PrimeTime PX invokes the appropriate utilities to convert the activity data to VCD where necessary. The utility must be in the search path and match the version of the data.

3. RTL to gate-level design object name mapping might be incorrect.

   If this is the case, provide a name-mapping file that contains a list of `set_rtl_to_gate_name` commands. For more information, see "Name Mapping" on page 3-5.

Performing Time-Based Power Analysis

After you select the time-based power analysis mode, using the `power_analysis_mode` variable, you set specific options for the power analysis mode, using the `set_power_analysis_options` command. The `update_power` command, while performing power analysis, considers the values you specified with the `set_power_analysis_options` command.

If you are performing cycle-accurate peak power analysis, you can restrict the activity propagation to specific blocks by specifying the `-cells` option with the `set_power_analysis_options` command. To restrict the propagation to specific time windows use the `-time` option of the `read_vcd` window. To complete the analysis faster restrict the propagation to the required time windows and blocks.

Using the `set_power_analysis_options` command without any options causes previously specified values to be reset; default values for the various options will be used for any further power analysis.

You can use the `report_power_analysis_options` command to see the values you set for the various options of the `set_power_analysis_options` command.

In the time-based mode, PrimeTime PX writes out the time-based power data calculated during the processing of the activity information in the VCD file. Before processing the events in the activity file, PrimeTime PX calculates the input transition time and output load for every instance in the design. These values are stored for use in accessing the appropriate energy values in the library power tables for the events in the activity file.

For transitions on cell output pins, PrimeTime PX looks for the input transitions that caused the output transition to ascertain the related pin. After identifying the pin, PrimeTime PX accesses the appropriate power arc and analyzes the energy for that transition based on the power table indexes.

The largest power value (peak power) and the time at which it occurred is included in the report file generated with the `report_power` command.

By default, the time-based power waveform data for all the hierarchies of the design is generated during the execution of the `update_power` command. To generate the waveform data for the leaf cells or for the top-level of the design hierarchy use the `-include` option of the `set_power_analysis_option` command appropriately before you specify the `update_power` command. You use the `report_power` command to see the waveform data.

Another approach is to set the `waveform_format` option to `none` with the `set_power_analysis_options` command. In this case, no waveform data is generated, but peak values are included in the power report.

Peak Power Analysis in PrimeTime PX

In the time-based power analysis, the `-waveform_interval` option of the `set_power_analysis_options` command is used to specify the sampling interval for the power waveform. When you do not specify a value for the sampling interval, the tool uses the timescale in the VCD file as the default value for this option.

PrimeTime PX performs the following tasks to arrive at the peak power numbers:

1. Analyzes every event in the VCD file
2. Checks for associated power arcs and related pins
3. Accesses the appropriate energy table in the technology library
4. Distributes the energy evenly for every event over the transition time for the inputs, and over the path delay for the outputs.
5. Generates power waveforms from the superposition of the distributed energy.

Figure 6-1 shows peak power analysis in PrimeTime PX for gate-level VCD with default time interval. The tool calculates energy for every event and distributes it over the path delay.

Figure 6-1 Peak Power Analysis for Gate Level VCD with Default Time Interval

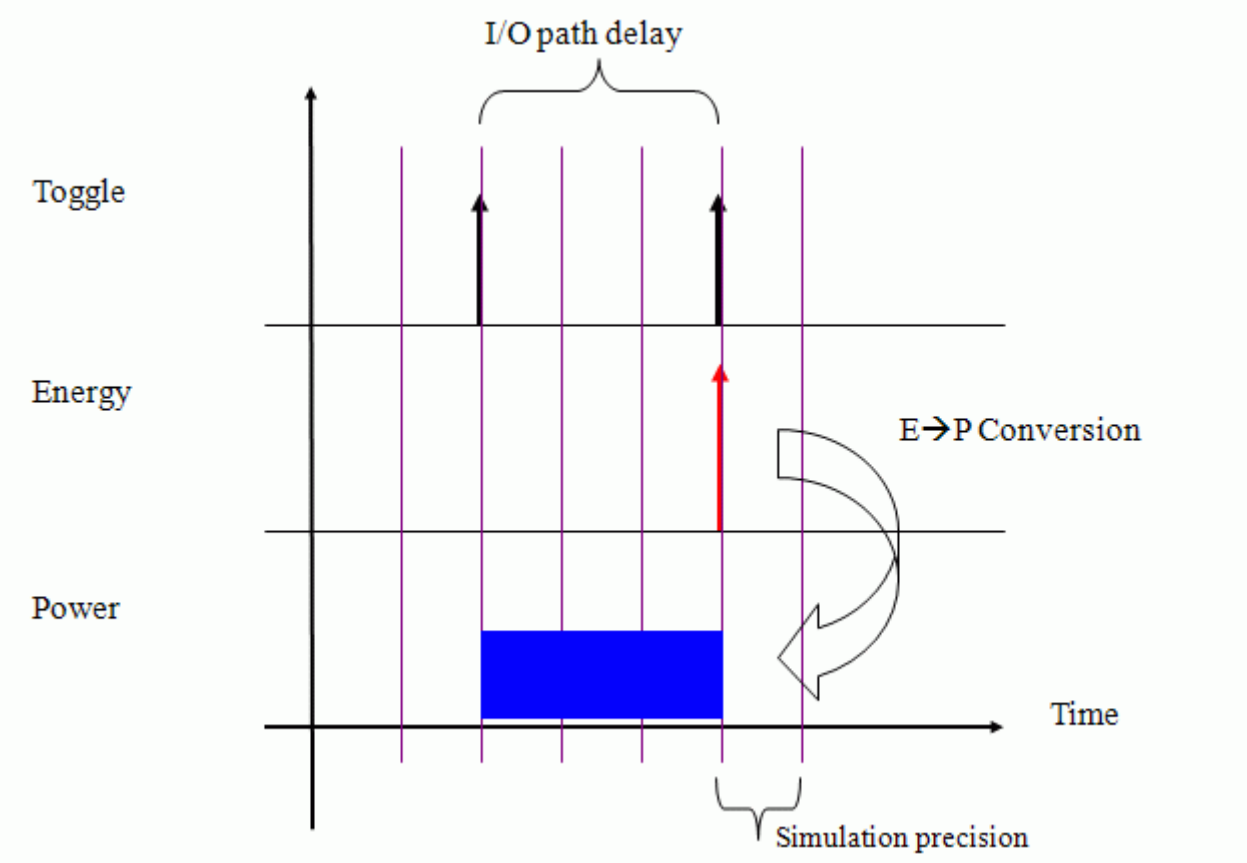
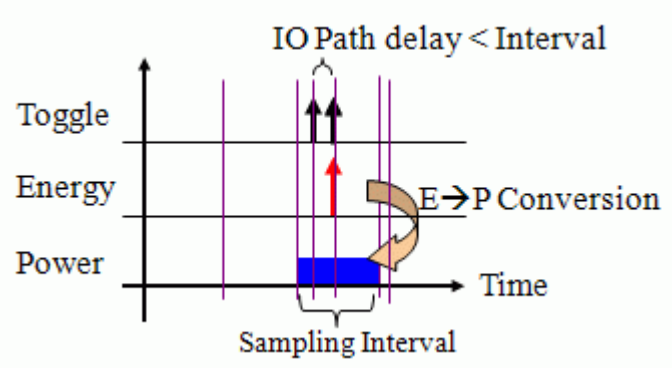


Figure 6-2 shows peak power analysis using gate-level VCD with user specified time interval. PrimeTime PX calculates the energy for every event and distributes it over the specified time interval if the time interval is greater than the I/O path delay. If the time interval is less than I/O path delay, the tool distributes the energy over the I/O path delay.

Figure 6-2 Peak Power Analysis for Gate-Level VCD with User Specified Time Interval



Understanding the Peak Power Calculation

During peak power analysis, PrimeTime PX generates a time-based power waveform and reports the peak power. It analyzes the energy and calculates the power for every event on every instance in the design to create a power profile over time. You can also generate power waveforms and view the peak power with the appropriate options.

The following sections describe the variables and command options that affect the power calculation.

Power Table Switching Activity

The `power_table_include_switching_power` variable indicates how the power tables have been characterized.

By default, the value is set to `true`. PrimeTime PX assumes that the library power tables were characterized such that half the switching activity was subtracted from both the rising and the falling edges, as shown in the following formulas.

Formula 1

```
Internal_energy_rise =
      Total_energy_rise - leakage_energy -
      0.5*switching_energy
```

Formula 2

```
Internal_energy_fall =
      Total_energy_fall - leakage_energy -
      0.5*switching_energy
```

Because switching energy (CV^2) is incurred only on the rising edge, the tables' values in this case do not reflect the true internal energy.

$$Eint\_table(rise) = Eint(rise) + 0.5CV^2$$

$$Eint\_table(fall) = Eint(fall) - 0.5CV^2$$

When the variable is set to `false`, PrimeTime PX assumes that the library internal energy tables contain only the internal energy. The rising and falling energy is derived during characterization using the following formulas.

Formula 3

```
Internal_energy_rise =
      Total_energy_rise - leakage_energy - switching_energy
```

Formula 4

```
Internal_energy_fall =
      Total_energy_fall - leakage_energy
```

The `power_table_include_switching_power` variable does not affect the average power because multiple events even out the difference, but it does affect the power waveforms, particularly for small designs and vector sets. For optimal peak power analysis, set this variable correctly.

The best way of finding out the formula used in the characterization is to consult the library vendor. A workaround is to inspect the falling power tables. If most of them have negative numbers, it is likely that formulas 1 and 2 are used in characterization.

Initial X-State Handling

When processing VCD data, PrimeTime PX processes every event. Many VCD files contain X states at initialization when the input values are being propagated.

The `power_include_initial_x_transitions` variable determines whether PrimeTime PX calculates the power during initialization or not.

By default, the variable is set to `false` and PrimeTime PX ignores the switching and internal power consumed by the transition of nets from an unknown to a known state. If, however, you are concerned with the power at initialization, you can set the value to `true`, so that the power peak produced as the circuit initializes can be included in the power analysis.

Unmatched States

When PrimeTime PX processes the events in the activity file, events for which there is no matching condition in a power table might occur. PrimeTime PX outputs statistical information that includes

```
X % tables on output pins matched
X % tables on input  pins matched
```

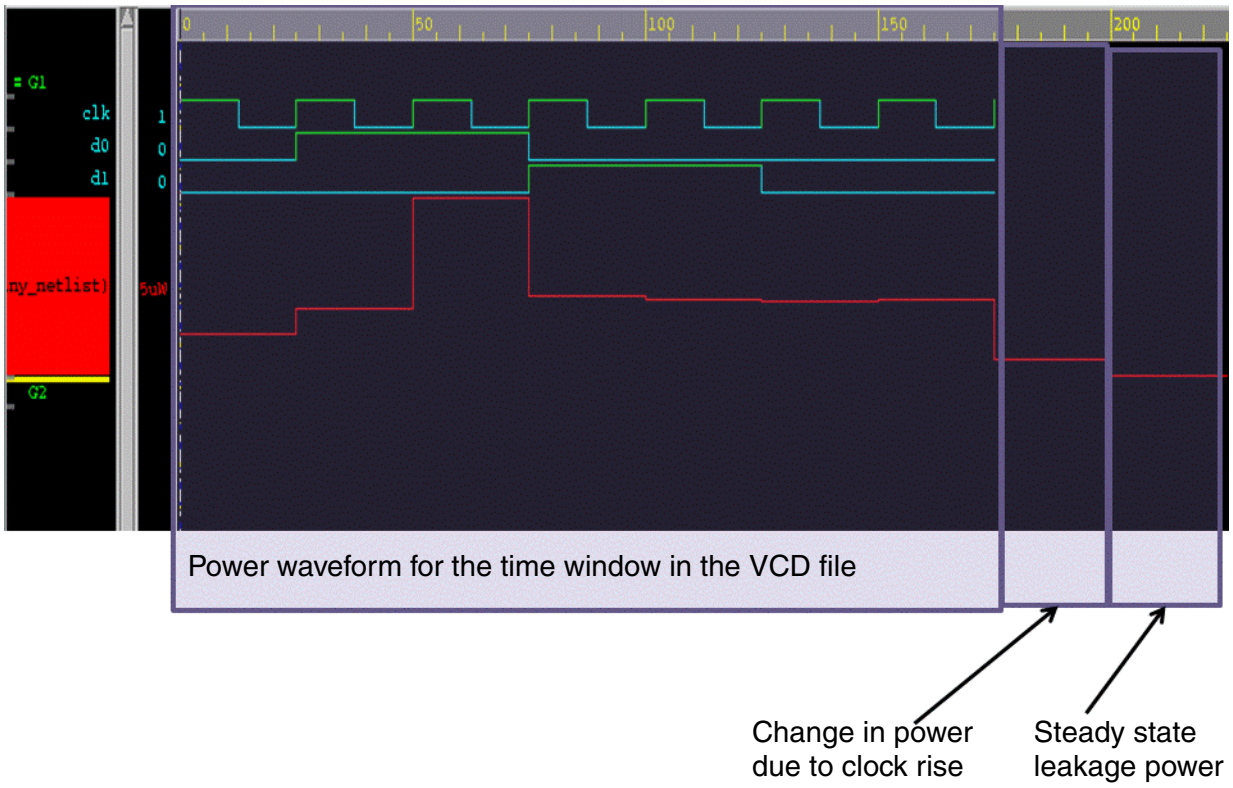
Most cells do not have input-pin-based tables resulting in 100 percent pin matching. However, for output cells, if the event does not match any of the state definitions in the power table, the event reduces the percentage of tables on output pins that matched. For these events, by default PrimeTime PX provides an estimate of the power consumption for that event based on the average value of all the available tables for that output pin.

If you want PrimeTime PX to ignore the event, you can set the `power_estimate_power_for_unmatched_event` variable to `false`.

Understanding the Peak Power Waveform

During peak power analysis, PrimeTime PX generates a time-based power waveform and reports the peak power over time. The power variation at every clock cycle is reported and is not limited by the VCD time window. Sometimes the waveform contains more cycles than in the VCD file because the tool monitors the cycles until it detects no further power variations. The power reported in the waveform matches the power reported by the `report_power` command.

Figure 6-3 Understanding Peak Power Waveform



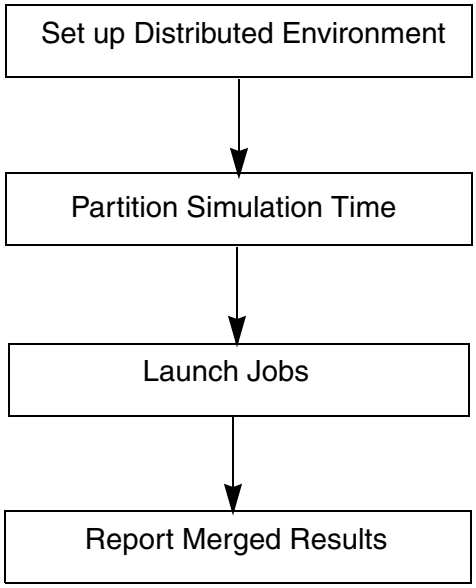
Distributed Peak Power Analysis

To speed up the peak power analysis process, PrimeTime PX performs distributed peak power analysis using the distributed multi-scenario analysis infrastructure available in PrimeTime. The distributed multi-scenario analysis flow and the usage model is similar in both PrimeTime and PrimeTime PX, except that, in PrimeTime the analysis is for timing, while in PrimeTime PX, the analysis is for power. In timing analysis, a scenario is a set of operating conditions and operating modes for the specified design. For power analysis, a scenario is a time window in the VCD activity file. Distributed multi-scenario analysis evaluates several scenarios in parallel by using multiple processors. This mechanism provides faster turnaround time and the ability to analyze the results from the multiple scenarios in parallel.

The steps involved in setting up the design for a distributed environment, mapping the analysis to scenarios, launching multiple scenario runs in parallel and merging the results are similar to that of PrimeTime and compatible for both timing and power analysis.

The distributed peak power analysis flow involves four steps as shown in Figure 6-4.

Figure 6-4 Distributed Peak Power Analysis Flow i



The first step in the flow is to set up a distributed computational environment that can run multiple tasks simultaneously. This step includes specifying a working directory, the log files, the number of PrimeTime and PrimeTime PX licences that can be used. This step also includes setting up and starting a distributed pool of machines. The variables and commands you use in setting up the distributed environment for peak power analysis are the same as the ones that you use in PrimeTime for timing analysis.

The following example script shows how to set up a distributed environment for peak power analysis when you have two machines in the LSF and you have two licenses each of PrimeTime and PrimeTime PX. The `power_enable_analysis` variable setting is required for reporting the power after the peak power analysis process.

```
set power_enable_analysis true

set multi_scenario_working_directory ./work
set multi_scenario_merged_error_log ./work/error_log.txt

set_multi_scenario_license_limit -feature PrimeTime 2
set_multi_scenario_license_limit -feature PrimeTime-PX 2

add_distributed_hosts   -32bit  -farm lsf  -num_of_hosts  2 \
        -submission_script /lsf/bin/bsub \
        -options {-o ./work -R rusage[mem=5000] }

create_distributed_farm
```

For more information about the variables and commands used in the setup, see *PrimeTime Multi-Scenario Analysis User Guide.*

The second step in the flow is partitioning the simulation time. This step involves dividing the total simulation time for the peak power analysis into many smaller time windows. The partitioning of the simulation window should be such that each time window takes almost the same time to run the peak power analysis. The number of smaller time windows into which the entire simulation time window is to be divided, depends on the number of machines in your pool and the number of licenses you have access to.

To run peak power analysis on each of the smaller time windows, create a scenario for every small time window using the `create_scenario` command. Use the `-common_data` and `-common_variables` options to specify data and variables that are common across the scenarios. Use the `-specific_data` option to specify scenario specific data.

In the following example, the entire simulation time 0 to 10000 is divided into two smaller windows, 0 to 5000 and 5001 to 10000. So, two scenarios are created, one for each smaller time window. Reading and linking the design, reading and applying the constraints and parasitics, and updating the timing are common across the two scenarios. Reading the VCD file is specific to each scenario.

An example of a partition script is as follows:

```
set test_dir .

create_scenario -name run_0 -specific_data run_0.tcl \
-common_data common.tcl -common_variables {test_dir}

create_scenario -name run_1 -specific_data run_1.tcl \
-common_data common.tcl   -common_variables {test_dir}
```

The script containing common data, in the file common.tcl, is as follows:

```
set power_enable_analysis true
set power_analysis_mode time_based
```

```
set search_path "."
set link_library    " * link_library.db"
read_verilog        mac.vg
current_design    mac
link
read_sdc $test_dir/mac.sdc
set_disable_timing [get_lib_pins ssc_core_typ/*/G]
read_parasitic $test_dir/mac.spef.gz
update_power
```

The scripts specific to the two scenarios, run_0.tcl and run_1.tcl, are as follows:

```
# The script run_0.tcl, specific to the run_0 scenario
read_vcd -time {0 5000} $test_dir/vcd.dump.gz -strip_path "tb/macinst"

# The script run_1.tcl, specific to the run_1 scenario
read_vcd -time {5001 10000} $test_dir/vcd.dump.gz -strip_path "tb/
macinst"
```

The third step in the distributed peak power analysis flow is to launch the jobs on the remote machines so that peak power analysis can run simultaneously on each of the smaller time windows. When your distributed environment is set up and you have created the scenarios, you must select a set of scenarios to be analyzed, using the `current_session` command. Use the `remote_execute` command to execute one or more commands on the remote machines. The `remote_execute` command builds a buffer of commands and triggers the execution of the commands in the buffer on the remote machines. The commands in the buffer are executed in the order in which they are mentioned. You can run this command only after you have selected the scenario(s) using the `current_session` command. For more details on these commands, refer to the command man page.

The following example script shows how to launch your jobs to run peak power analysis in a distributed environment.

```
current_session -all

remote_execute { \
   report_power
}
```

The final step in the distributed peak power analysis process is to display the merged power report. You must first set the `power_enable_analysis` variable to `true`. You can then execute the `report_power` command on the master process. Currently, the options for `report_power` are supported only when the command is executed in the slave processes.

The license requirement for using the distributed multi-scenario analysis feature is similar to that of PrimeTime. For more information about this feature, see *PrimeTime Multi-Scenario Analysis User Guide.*

Viewing and Scaling the Power Waveforms

You can view waveform data by using nWave. To invoke nWave, specify the following command on the UNIX command line:

```
%> nWave
```

To view the waveform data in nWave, choose File > Open and select the desired file. This loads the desired .fsdb or .out file. You can select signals from within nWave by choosing Signal > Get All Signals from the menu bar.

This viewer requires a snps_fs_nwave license.

By default, nWave scales all the individual waveforms to a default height. To view the waveforms scaled to the total power consumption, do the following:

1. Select all the signals in the Signal column.
2. Choose Analog > Zoom Value from the nWave menu bar.
3. In the dialog box that appears, select Full and then Close.

7

Multivoltage Power Analysis

This chapter describes multivoltage power analysis in the following sections:

Introduction to the Multivoltage Infrastructure

PrimeTime PX can analyze the power for designs with different power supply voltages for different cells. The multivoltage infrastructure allows you to choose between two distinct methods for calculating power for designs with different power supplies. These methods are not interchangeable.

- The IEEE 1801™ (UPF), Unified Power Format method
  - Use the UPF to specify your power intent
- Alternate or non-UPF method
  - Use power domains, power nets and power and ground pins to specify your power intent
  - Group cells based on the design power rails, and identify the power consumption of the design per rail

In addition, PrimeTime PX supports power analysis of designs that include modules that are powered on and off. Rather than assuming that the power supply to the cells is always on, PrimeTime PX takes into account the power control signal for accurate analysis of power-down modes.

The multivoltage infrastructure supports the following capabilities:

- Accurate power calculation for cells with multiple power supplies by using internal and leakage power tables with liberty power-and-ground pin syntaxes
- Power-gating-aware power calculation in which the tool recognizes power on and off states for design blocks and reflects power-saving technology in leakage power calculation
- Voltage scaling for power calculation

Multivoltage Libraries

The Synopsys multivoltage design flow supports composite current source (CCS) and multiple and merged nonlinear power model (NLPM) libraries. In particular, for power analysis with multivoltage designs, PrimeTime PX uses the following power-and-ground pin library data:

- `pg_current` group inside the `dynamic_current` group in CCS; The `related_pg_pin` attribute for pin internal power table in NLPM
- `pg_current` group inside the `leakage_current` group in CCS; The `related_pg_pin` attribute for cell leakage power table in NLPM

By using pin-related power tables, PrimeTime PX can accurately calculate power for cells with multiple power or ground rails.

The tool also supports rail-based power reporting by calculating internal, switching, and leakage power for single supply and multi-supply cells in multivoltage designs and reports power consumption for each individual power-and-ground rail.

Library Power and Ground (PG) Pin Conversion at Runtime

To specify power intent for UPF designs, PrimeTime and PrimeTime PX convert and update the library power and ground (PG) pins automatically using the specifications provided in a Tcl file. For more details, see *PrimeTime Advanced Timing Analysis User Guide*.

Power-Scaling CCS and NLPM Libraries

To improve power analysis accuracy, PrimeTime PX allows you to scale CCS and NLPM libraries that have different voltages and temperatures. Scaling reduces the need to have separate libraries for each incremental voltage and temperature.

The tool supports power scaling for averaged and time-based power calculations for all power analysis modes. To perform scaling, specify the following command:

```
define_scaling_lib_group library_list
```

When this command is specified, PrimeTime PX interpolates (based on voltage and temperature) the cell characteristics from the listed libraries, which are the scaling group. All power consumption components—total, dynamic, internal, switching, leakage, x-transition, and glitch—are scaled. The output of power analysis reflects the effect of scaling.

During the scaling operation, you can use the `set_operating_conditions` and `set_rail_voltage` commands to specify the voltage and temperature that are different from the ones defined in the libraries. Set the operating condition within the range covered by the scaling group. PrimeTime PX checks that the user-specified conditions are within the range of the applicable scaling group. If they are outside the range, it issues an error message and uses the voltage or temperature from the primary linked library instead.

The following conditions apply if you decide to scale libraries:

- Specified libraries must satisfy the following consistency checks:
  - The same set of cells is characterized across the libraries. If this check fails, the scaling group is not created.
  - The same set of power tables is presented for each cell across the libraries within a scaling group. If this check fails, scaling does not occur and PrimeTime PX issues a warning.

    - The power tables are uniquely identified. If this check fails, the tool assumes that the power tables are presented in identical order across all libraries.

- Before scaling the libraries, you must have defined the scaling relationships between the libraries with the variables described later in this section.
- The specified libraries should characterize the cells at different voltage and temperature coefficients.
- Specify the `define_scaling_lib_group` command after you have read and linked the design.
- You can scale different groups of libraries together to cover desired portions of the design; however, specify each library once only.
- The library specified within the `link_path` variable is the default library, and the tool uses the values within this library if you are not performing scaling. When scaling occurs, the remaining libraries are read automatically after the design is linked. For example, given

  ```
  pt_shell> define_scaling_lib_group {lib1.db lib2.db lib3.db}
  ```

  you could set `link_path` as follows:

  ```
  pt_shell> set link_path "* lib2.db"
  pt_shell> link_design
  pt_shell> define_scaling_lib_group {lib1.db lib2.db lib3.db}
  ```

- If you remove a library with the `remove_lib` command, the scaling group that contains that library is removed also.
- If by linking the design again you cause a library within a scaling group to be read again, you must specify the scaling group again with the `define_scaling_lib_group` command.

PrimeTime PX uses a nonlinear interpolation method for voltage scaling of leakage power and internal power. For temperature scaling of leakage power, the tool uses the nonlinear interpolation method. For temperature scaling of internal power, a linear interpolation method is used.

Using the `link_path_per_instance` variable is another way to provide multivoltage library data, for PrimeTime PX to use different libraries for different cell instances. Before the linking process, if you set the `link_path_per_instance` variable, the tool overrides the default `link_path` variable setting, for the specified leaf cell or hierarchical cell instances. For more details on the `link_path_per_instance` variable, see the *PrimeTime Advanced Timing Analysis User Guide*.

Gate Leakage Power Scaling

During power analysis, if you specify an operating condition, other than those for which the cells are characterized for CCS gate-leakage power, PrimeTime PX appropriately scales the gate-leakage power based on the voltage and temperature values specified for the cells. The scaling of the gate-leakage power is done by interpolating the gate-leakage values from the characterized scaling libraries. However, the operating condition that you specify should be within the range covered by the scaling libraries. The tool supports gate-leakage power scaling for both averaged power analysis and time-based power analysis.

The following example shows a technology library cell that has intrinsic leakage and gate-leakage characterization supported for the cells:

```
cell (cell_name) {
 ...
  leakage_current () {
    when : state_1;
    pg_current(pg_pin_1) {
      value : val_1_1;
    }
    ...
    pg_current(pg_pin_n) {
      value : val_1_n;
    }
    gate_leakage(input_1) {
      input_low_value : val_low_1_1;
      input_high_value : val_high_1_1;
    }
    ...
    gate_leakage(input_n) {
      input_low_value : val_low_1_n;
      input_high_value : val_high_1_n;
    }
  }
  leakage_current() {
    when : state_2;
    ...
  }
  leakage_current() {
    /* default state */
   ...
  }
}
```

Multirail Scaling Support

PrimeTime PX supports multirail scaling by default for power, timing, and noise analysis. Based on the availability of libraries in the scaling library group, the tool automatically detects multiple rails and performs multirail scaling. If libraries are not available in the scaling library group, the tool performs single rail scaling. For more details, see *PrimeTime Advanced Timing Analysis User Guide*.

The tool provides accurate multirail scaling and supports automatic selection of the correct library from the scaling library group. The selection is based on the exact matching of the operating condition of a design cell or pin with the operating condition of a library in the scaling library group. Also, as a result of this feature, you can avoid using the `link_path_per_instance` variable on multirail cells.

Multivoltage Cells

For PrimeTime PX to determine the contribution of each power rail in a multi-supply cell, you must provide separate power tables for each rail in the library model. For example, for a multivoltage cell with VDD1 and VDD2, you would use the following:

```
pin(Y) {
   ...
   internal_power() {
     related_pg_pin : PVDD1 ;
     power (power_r1) {
       values ("1.934150. 2.148130");
     }
   }
   internal_power() {
     related_pg_pin : PVDD2 ;
     power (power_r1) {
       values ( "1.634150, 2.548130" );
     }
   }
}
```

The following example shows how you can determine the power consumption of each power and ground rail by using power and ground pins. The `related_pg_pin` attribute differentiates power consumption based on the power and ground rails.

```
cell (SHIFTER) {
   ...
   pg_pin(PVDD1){
     voltage_name: "VDD1";
     pg_type : primary_power;
}

   pg_pin(PVDD2) {
     voltage_name : "VDD2";
     pg_type: primary_power;
}

   pg_pin (SVSS) {
     voltage_name : "VSS";
     pg_type: primary_ground;
}
...
```

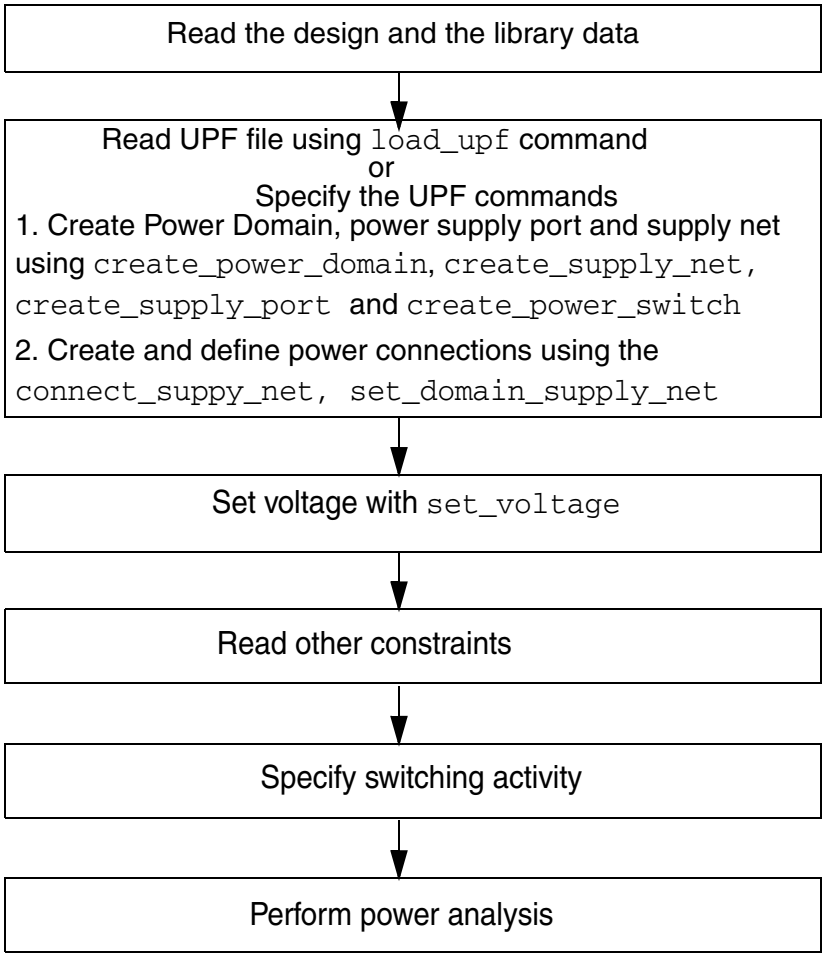
```
Pin (Y) {
   ...
   related_power_pin : PVDD1;
   related_ground_pin : PVSS;

   internal_power () {
      related_pg_pin : PVDD1;
      power (power_r1) (
        values ("1.934150. 2.148130");
      }
    }
   internal_power () {
      related_pg_pin : PVDD2;
      power (power_r1) (
        values ( "1.634150, 2.548130" );
      }
   }
 }
}
```

Multivoltage Power Analysis Using IEEE 1801™ (UPF)

In PrimeTime PX, you can use UPF to specify your power intent for the analysis of your multivoltage design. UPF provides the ability to specify the power intent early in the design process, and it supports the entire design flow. Figure 7-5 shows the UPF flow for multivoltage power analysis using PrimeTime PX.

Figure 7-5 Multivoltage Power Analysis Flow Using UPF



For more information about low-power flow and various Synopsys tools that support UPF, see the *Synopsys Low-Power Flow User Guide*. When you invoke PrimeTime PX, by default, the power analysis using UPF is enabled. You can specify the UPF commands at the command prompt. Alternatively, you can define the UPF commands in a separate text file, generally with a .upf extension. You can use the `load_upf` command to read the UPF file into PrimeTime PX. For more information about the `load_upf` command, see the `load_upf` command man page. For a list of UPF commands supported in PrimeTime PX UPF mode, use the help command as follows:

```
pt_shell> help UPF
```

Power Domains

Multivoltage designs contain design partitions which have specific power behavior compared to the rest of the design. A power domain is a basic concept in the Synopsys low-power infrastructure, and it drives many important low-power features across the flow. A power domain describes a design partition, bounded within logic hierarchies, which has a specific power behavior with respect to the rest of the design.

Note:
A power domain is strictly a logic construct, not a netlist object.

Logical hierarchy where the power domain is created is called the scope of the power domain. Design elements that belong to a power domain are said to be in the extent of the power domain. You can use the `set_scope` command to specify the scope or level of hierarchy. The `set_scope` command sets the scope or the level of hierarchy to the specified instance. For more information, see the *Synopsys Low-Power Flow User Guide*.

Use the `create_power_domain` command to create a power domain. The syntax of the `create_power_domain` command is as follows:

```
create_power_domain
[-elements element_list]
[-include_scope]
[-scope instance_name]
domain_name
```

Use the `-elements` option to specify the list of hierarchical cells, input and output pad cells, and macro cells, that are added as extent of the power domain. The `-include_scope` option specifies that all the elements in the current scope share the primary supply of the power domain, but are not necessarily added as extent of the power domain. Use the `-scope` option to specify the logical hierarchy or the scope at which the power domain is to be defined. `domain_name` is the name of the power domain to be created.

Use the UPF commands such as `create_supply_net`, `create_supply_port`, and `connect_supply_net`, to create ports, nets and connect them and to build the power supply network for your design. For more information, see the command man page.

Every power domain must have a primary supply power net and a primary ground net. Use the `set_domain_supply_net` command to define the primary supply and primary ground net for an existing power domain.

Isolation Cells

In a design with power switching, an isolation cell is required where each logic signal crosses from a power domain that can be powered down to another power domain that is not powered down. The isolation cell operates as a buffer when the input and output sides

of the isolation cell are both powered up. When the input side of the cell is powered down, the isolation cell provides a constant output signal. An enable input controls the operating mode of the isolation cell.

Isolation Commands

Use the isolation commands, `set_isolation` and `set_isolation_control` to specify the strategy for inserting isolation cells at the outputs of switched (power-down) domains.

set_isolation

The `set_isolation` command specifies the isolation strategy for a power domain and the elements in that domain on which the strategy is applied. An isolation strategy includes specification of the enable signal net, the clamp value, and the location (inputs, outputs, or both). This command creates explicitly connection for isolation power and ground nets to the power and ground pins of the isolation cell.

If both `-elements` and `-applies_to` arguments are not specified, the isolation strategy is applied to all the output ports of the power domain. Every isolation strategy defined by a `set_isolation` command must have a corresponding `set_isolation_control` command unless the strategy is `-no_isolation`.

The isolation strategies have the following decreasing level of precedence, irrespective of the order in which they are executed:

- Stategies defined by specifying explicit list of pins or ports with the `-elements` option; with or without using the `-applies_to` option.
- Strategies defined by specifying the list of design elements with the `-element` option; with or without using the `-applies_to` option.
- Strategies defined by using the `-applies_to` option, but not the `-elements` option.

For more details, see the command man page.

set_isolation_control

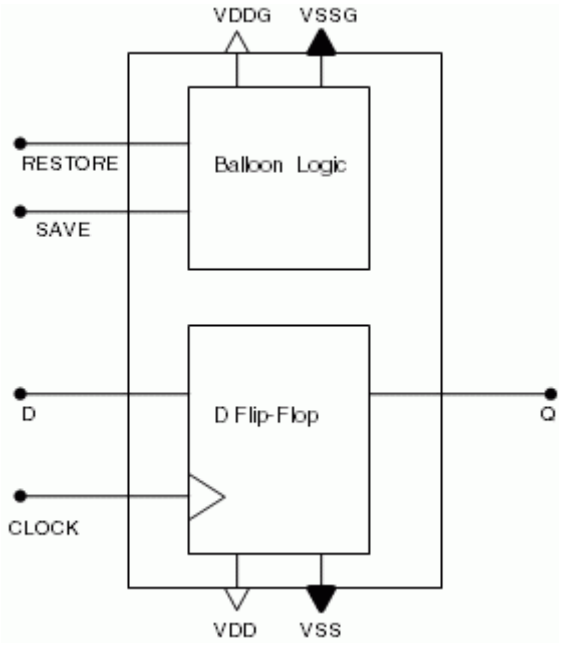
The `set_isolation_control` command allows the specification of the isolation control signal and the logical sense of that signal. The command identifies an existing isolation strategy and specifies the isolation control signal for that strategy.

For more details, see the command man page.

Retention Registers

In multivoltage designs, when a power domain is shut down and later powered up, it is often necessary for the power domain to resume operation based on its last good state. Special cells called retention registers can store the state during the shutdown. As shown in Figure 7-6, a retention register has two control signals, save and restore, to save and restore the data. Retention cells occupy more area than regular flip-flops. These cells continue to consume power when the power domain is powered down.

Figure 7-6 Two Pin Retention Register



Single Control Pin Retention Register

Library Compiler supports modeling of retention registers with only one control pin called the `save_restore` pin. The `save_restore` pin saves and restores the state of a cell. In general, when the cell is in save mode, it operates as a flip-flop or a latch. When the cell is in restore mode, the previously saved value is available on the Q pin of the cell. For more details, see the Advance Low-Power Modeling chapter in the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*.

Retention Commands

The retention commands specify the strategy for inserting retention cells inside switched (power-down) domains.

set_retention

The `set_retention` command specifies which registers in the domain are to be implemented as retention registers and identifies the save and restore signals for the retention functionality. At a minimum, either `-retention_power_net` or `-retention_ground_net` must be specified. If both are used, they specify the supply nets to use as the retention power and ground nets. If only the retention power supply net is specified, the primary ground net is used as the retention ground supply. If only the retention ground net is specified, the primary supply net is used as the retention power supply. The retention power and ground nets are automatically connected to the implicit save and restore processes and shadow register.

The following strategies have decreasing level of precedence, irrespective of the order in which they are executed:

```
set_retention -domain -elements
set_retention -domain
```

Every retention strategy defined by a `set_retention` command must have a corresponding `set_retention_control` command.

set_retention_control

The `set_retention_control` command allows the specification of the retention control signal and the logical sense of that signal. The command identifies an existing retention strategy and specifies the save and restore signals and logical senses of those signals for that strategy.

For more details, see the command man page.

Power Gating in UPF Mode

In the UPF mode, PrimeTime PX can do power gating aware power calculation to reduce the leakage power. When cells in the design are turned off, power gates disconnect the cells from the power supply resulting in reduced or no leakage power consumption. A power switch is an logical object to transform a supply net from an input port to an output port based on the state of the control port. The state of the control port is defined by the boolean expression.

Using the `create_power_switch` command, you can define a power switch in a specific power domain. The power switch is created within the scope of the specified power domain. This command also lets you specify the power down control so that when cells are turned off, they are disconnected from the power supply. When the Boolean expression associated with the `on_state` option evaluates to `true`, the switch is in the on state and the value at the input supply port is propagated to the output supply port. The power pin of a cell is switched off only when the following conditions are satisfied.

- The power domain that the cell belongs to has power switches specified.
- The power supply net of the cell is connected to the output supply net of one of the power switches.
- The value of the boolean expression, associated with the on_state of the power switch, is FALSE.

PrimeTime and PrimeTime PX only support single input, single output, multi control switches.

The syntax of the `create_power_switch` command is as follows:

```
create_power_switch
-domain domain_name
-output_supply_port {port_name supply_net_name}
-input_supply_port {port_name supply_net_name}
-control_port {port_name net_name}
-on_state {state_name port_name {boolean_function}}
[-off_state {state_name {boolean_function}}]
[-on_partial_state {state_name port_name {boolean_function}}]
[-error_state {state_name {boolean_function}}]
[-ack_delay {port_name ack_delay}]
switch_name
```

You use the `-domain` option to specify the power domain in which the power switch is to be created. The `-output_supply_port` option is used to specify the output port of the power switch and the supply net to which this output port should be connected. The `-control_port` option specifies the control port of the power switch and the net to which this control port is to be connected to. Because multiple control ports are supported for the power switch, you can specify multiple pairs of control port and the associated net, using the `-control_port` option. Use the `-on_state` option to specify the state, the associated input port and the boolean function.

For more information, see the `create_power_switch` command man page.

Note:

There are more options to the `create_power_switch` command, such as `-ack_port`, `-on_partial_state`, `-ack_delay`, `-off_state`, and `-error_state`. These options are not supported in PrimeTime PX. These options are ignored by the tool without issuing any warning message. Other UPF commands related to power switches such as the `map_power_switch` and `set_power_switch` commands are not supported by PrimeTime PX. These commands are ignored by the tool with a warning message.

The `report_power_switch` command is supported in PrimeTime PX to report all the power switches defined in the design. This command reports details of the power switch such as the names of the power switches, the power domains in which the switches are defined, input, output and control port details. This command however is not a UPF standard command. For more information, see the `report_power_switch` command man page.

Voltage Scaling in UPF Mode

In multivoltage designs voltage scaling improves the accuracy of the power calculation. Voltage scaling is supported for both internal power and leakage power calculation. The voltage scaling is supported in both CCS and NLPM power library format. The library cells are characterized at different operating conditions and present in separate libraries. Because of the support of power scaling, during power analysis, the data can be interpolated from these separate libraries for the desired operating condition.

Use the `set_voltage` command, supported by UPF standard, to define the operating voltage on power and ground pins and the power and ground nets. By defining the operating voltage, the parts of the design powered by these nets and power and ground pins are optimized for the specified voltage. The effective voltage used by the tool for power calculation is the difference between VDD and VSS.

Because the `set_voltage` command is available in the UPF mode to define the operating voltage, the `set_rail_voltage` command is not supported in PrimeTime and PrimeTime PX in the UPF mode.

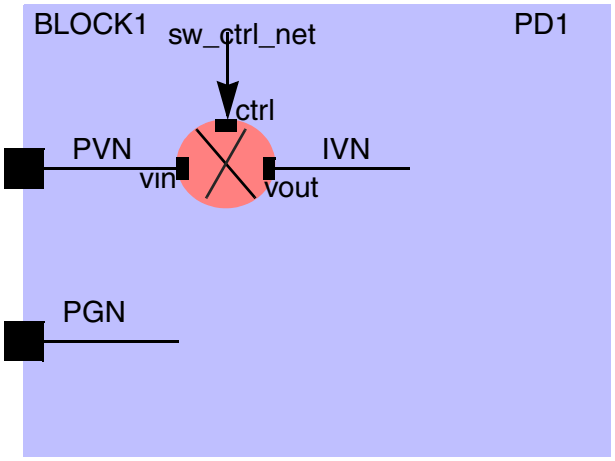
Note:

The operating voltage defined on power nets using the `set_voltage` command is not propagated across the power switch. To have similar operating voltage for nets across the power switch, use the `set_voltage` command on the supply net on both sides of the power switch.

Power Analysis in UPF Mode

Figure 7-7 shows a power switch for a block. The block, Block1, has power domain PD1, two supply ports PVN and PGN, three supply nets PVN, PGN and IVN, a control net, sw_ctrl_net and a power switch, sw1. The power switch sw1 has one input supply port, vin, one output supply port, vout, and one control port, ctrl. All the cells in the block, Block1, can be powered down by switch sw1 based on the boolean function for the `-on_state` option specified while creating the power switch using the `create_power_switch` command.

Figure 7-7 Power Switch



The following example shows a typical script used for power analysis.

```
# Read the target libraries and input designs

set power_enable_analysis true
set power_analysis_mode time_based

set link_library lib.db
read_verilog design.v
current_design design_top
link

# Read the UPF file containing following UPF commands
# load_upf design_upf.tcl

# Alternately specify the UPF commands
create_power_domain PD1
create_supply_net PVN -domain PD1
create_supply_port PVN -domain PD1
connect_supply_net PVN -ports PVN

create_supply_net IVN -domain PD1

create_power_switch sw1 -input_supply_port {vin PVN} -output_supply_port
{vout IVN} -control_port {ctrl sw_ctrl_net} -on_state {state1 vin
{sw_ctrl_net}}
create_supply_net PGN -domain PD1
create_supply_port PGN -domain PD1
connect_supply_net PGN -ports PGN

set_domain_supply_net PD1 -primary_power_net IVN -primary_ground_net PGN
set_voltage on_supply_net
set_temperature on_blocks

# Specify other constraints
read_sdc design.sdc

# Read parasitics
read_parasitics design.spef

# Specify switching activities
# The input file can be in VCD or SAIF formats.
read_vcd file_name

# Do the power analysis, averaged or timed-based analysis
update_power

# Get the report after the analysis
report_power
```

To handle the serial and parallel connection of power switches, PrimeTime PX supports the `set_supply_net_probability` command. This command allows you to specify the static probability that the specified net is on. This is useful in averaged power analysis. The `set_supply_net_probability` is not a UPF standard command. The syntax of the `set_supply_net_probability` command is as follows:

```
set_supply_net_probability [-remove] supply_net float
```

You can specify the UPF supply nets to apply the probability. The supply net specified can be a single supply net, or a collection of supply nets, returned by the `get_supply_nets` command. If supply net is not specified, annotation is not performed.

The float value specified is the probability that the supply net is powered on. The value for the probability should be between 0 and 1.

Use the `-remove` option to remove an existing annotation on the specified supply nets.

Support for UPF Footer Switches

PrimeTime PX supports UPF power switches on both the power supply net and the ground net. Power switches specified on the ground net are also known as UPF footer switches. This support is available in the averaged and time-based power analysis modes. For single- and multi-ground cells that are controlled by UPF footer switches, PrimeTime PX provides the support in the following ways:

- Single-ground cells

  In this type of cell, when the ground net is switched off, power consumption of the entire cell is shut off.

- Multi-ground cells

  In this type of cell, when all the grounds are switched off, the power consumption of the entire cell is shut off. If any of the grounds are not switched off, the cell is not shut off and it continues to consume power.

Note:

To use this feature, your target library must comply with the PG pin Liberty library syntax. PrimeTime and PrimeTime PX support the conversion and update of the power and ground (PG) pins of the library. They also support converting non-PG pin libraries to PG pin libraries. For more details, see *PrimeTime Advanced Timing Analysis User Guide*.

Power Domain Based Power Reporting in UPF Mode

In the UPF mode, PrimeTime PX supports reporting power for every power domain or every power net based on your choices. The tool also gets the power domains and power nets selected for power analysis. The use model of this feature is similar to the currently supported rail-based power reporting in non-UPF mode.

To use this feature, you must first specify the power intent for your design using UPF. Before the power analysis step, you must select the power domains or the power nets that you want to report. During the reporting, the tool generates a single power report for the power domains or the power nets that you selected. If you do not specify or select power domains or power nets, the tool, by default, reports power for all the power domains and all the power nets. This feature is supported for all the flows and all four types of reporting: summary, cell-based, net-based and hierarchy-based reporting.

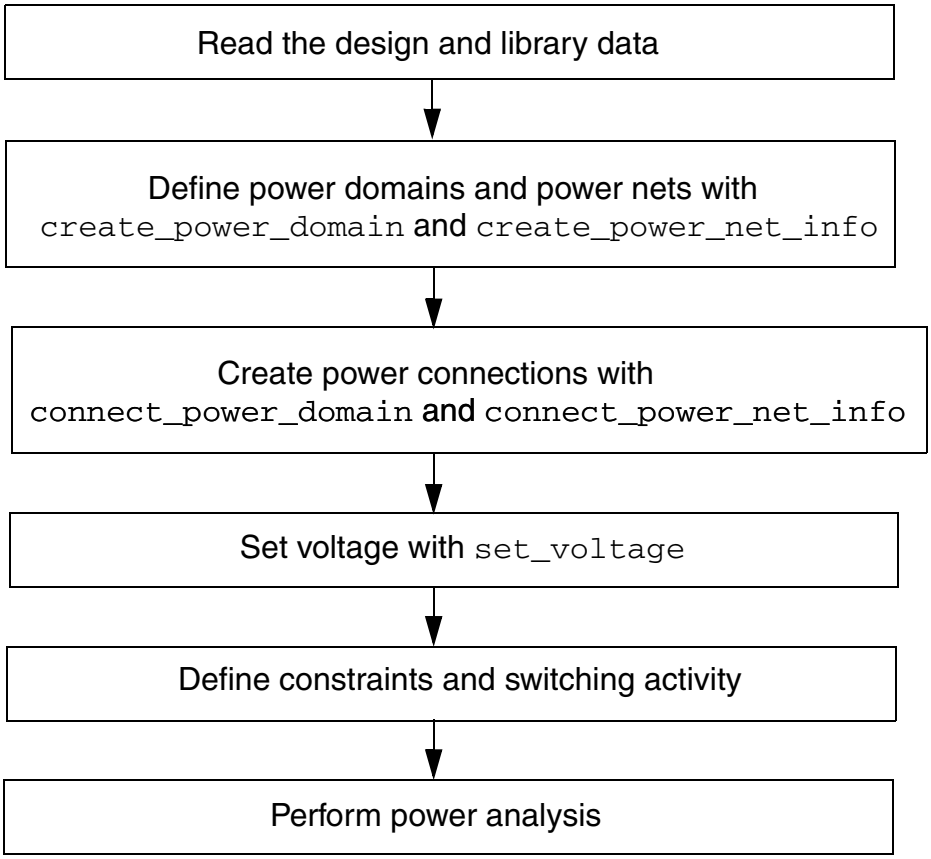
Multivoltage Power Analysis Using Power Domains

To invoke PrimeTime PX in the non-UPF mode or Power Domain mode, set the `power_domains_compatibility` variable to `true`. Setting this variable to `true` lets you use the power domains and power rail method of power analysis. The default value of the `power_domains_compatibility` variable is `false`. To get a list of commands supported in this mode use the help command as follows:

```
pt_shell> help "power domains"
```

Figure 7-8 shows the usage flow for multivoltage power analysis when you use power domains. For more information about domains, see *PrimeTime Advanced Timing Analysis User Guide*.

Figure 7-8 Multivoltage Power Analysis Flow Using Power Domains



The following commands support multivoltage power analysis using domains:

- The `create_power_domain` command specifies the name of a power domain and lists the hierarchical cells associated with the domain. The power domain applies to the top level if you don't specify a list. There can be only one top-level domain. Use the following options for power-down specifications:
  - `-power_down` indicates that the specified domain can be powered down.
  - `-power_down_ctrl` *`net_list`* specifies the nets that control the power to the specified domain. These are nets that can cause the domain to be powered down.
  - `-power_down_ctrl_sense 0 | 1` specifies the falling or rising status for the powered-down control nets specified with the `-power_down_ctrl` option.

  For example, suppose power domain A is controlled by signal B. PrimeTime PX needs to know if A's power goes down when B is high (1) or low (0). The following command indicates that A's power goes down when signal B is high:

```
create_power_domain A -power_down \
     -power_down_ctrl B -power_down_ctrl_sense 1
```

- The `create_power_net_info` command specifies the name of a power supply net or ground net in the design. For power supplies, the command also specifies the voltage and whether the power can be switched off.
- The `connect_power_domain` command creates logical power connections for a specified power domain. It specifies the primary, backup, and internal power and ground nets for a specified power domain. All cells in the power domain inherit the specified power connections.
- The `connect_power_net_info` command makes power net connections for a specific power pin of a leaf cell. The pin-level connections override the domain-level connections made with the `connect_power_domain` command.
- The `set_voltage` command defines the operating voltage on the power nets defined by the `create_power_net_info` command. You can specify a single voltage or minimum and maximum voltages for the power net. If you do not use this command, the available operating condition settings are used. To report the operating voltages of power nets, use the `report_power_net_info` command.

Linking the design again causes the tool to discard power domain information.

The following sample script demonstrates how to perform multivoltage power analysis using domains:

```
# Read libraries, design, enable power analysis
# and link design
set power_enable_analysis true
set link_library slow_pgpin.db
read_verilog power_pins.v
link

# Create back-up power nets
create_power_net_info vdd_backup -power
create_power_net_info vss_backup -gnd

# Create domain power nets
create_power_net_info t_vdd -power -switchable \
     -nominal_voltages{1.2} -voltage_ranges{1.1 1.3}
create_power_net_info a_vdd -power
create_power_net_info b_vdd -power

# Create domain ground nets
create_power_net_info t_vss -gnd
create_power_net_info a_vss -gnd
create_power_net_info b_vss -gnd

# Create internal power nets
create_power_net_info int_vdd_1 -power \
```

```
      -nominal_voltages{1.2} -voltage_ranges[1.1 1.3} \
      -switchable
create_power_net_info int_vdd_2 -power \
      -nominal_voltages{1.25} -voltage_ranges{1.1 1.3}
create_power_net_info int_vdd_3 -power \
      -nominal_voltages{1.2} -voltage_ranges{1.1 1.3}
create_power_net_infor int_vdd_4 -power

# Create power domains
create_power_domain t
create_power_domain a -object_list[get_cells PD0_inst]\
      -power_down -power_down_ctrl[get_nets a] \
      -power_down_ctrl_sense 0
create_power_domain b -object_list [get_cells PD1_inst]\
      -power_down

# Connect rails to power domains
connect_power_domain t -primary_power_net t_vdd \
      -primary_ground_net t_vss
connect_power_domain a -primary_power_net a_vdd \
      -primary_ground_net a_vss \
      -backup_power_net vdd_backup \
      -backup_ground_net vss_backup
connect_power_domain b -primary_power_net b_vdd \
      -primary_ground_net b_vss

# Set voltages of power nets
set_voltage 1.15 -object_list{t_vdd a_vdd b_vdd}

# Read ASDC and other timing or power assertions
set_input_transition 0.0395 [all_inputs]
set_load 1.0 [all outputs]
set_switching_activity...
set_switching_activity...
...

report_power
```

Multivoltage Power Analysis Using Power Rails

To report power by rails, you need to group the cells in a design by the power rails or by ground rails if the library contains the power and ground pins. For multivoltage cells, such as level shifters, the library power rails need to be linked to the design power rails.

The following is an example of grouping cells:

```
pt_shell> create_power_rail_mapping V1 -cells “G1 G2 G3”
pt_shell> create_power_rail_mapping V2 -cells “B1 B2”
pt_shell> create_power_rail_mapping V3 -cells R1
```

In this example, G1 refers to one of the green instances and R1 refers to the red cell. Other instances are named in the same fashion. V1 is one power rail in the design.

For libraries with power and ground pins, use the `create_power_rail_mapping` command to group cells based on the ground rail as well. For example,

```
pt_shell> create_power_rail_mapping VSS1 -cells "G1 G2 G3"
pt_shell> create_power_rail_mapping VSS2 -cells "B1 B2"
```

After the design is grouped, the power report covers only the power of the selected rails. Here is a simple sample script of a multivoltage design analysis:

```
link
create_power_rail_mapping \
  V1 -instance "G1 G2 G3" \
current_power_rail "V1"
report_power
```

In this example, the `current_power_rail` command allows you to select the rails for which to report power. You can select one rail or a list of power rails. PrimeTime PX reports the sum of the selected power rails. Without this command, all the rails' power is reported together. The `report_power` command automatically initiates the `update_power` command.

The following sample script uses the `current_power_rail` command to view the power consumption per rail for a design that contains two power rails, VDD1 and VDD2:

```
  current_power_rail VDD1
  report_power > pp_rail_VDD1
  current_power_rail VDD2
  report_power > pp_rail_VDD2
```

The `report_power` command output contains the power consumption only for the last specified rail. The rails for which power is calculated are defined in the Voltage Rail heading of the report file, as shown in the following sample file:

```
****************************************
Report : power
Design : d
Version: Y-2007.06
Date   : Wed Jan 11 09:27:35 2007
****************************************
Sampling Interval: 1 ns
Current Voltage Rail: VDD2

Library(s) Used:

    pg_pin_lib (File: /remote/pwrcae1/power_level.db)

Operating Conditions: WORST   Library: power_level
```

```
Wire Load Model Mode: top

<no wire load model is set>

Power-specific unit information :
    Voltage Units = 1 V
    Capacitance Units = 1 pf
    Time Units = 1 ns
    Dynamic Power Units = 1 W
    Leakage Power Units = 1 W

  Cell Internal Power  = 3.398e-05  (59%)
  Net Switching Power  = 2.335e-05  (41%)

Total Dynamic Power    = 5.733e-05  (100%)
Cell Leakage Power     = 1.902e-10

Peak Power             = 2.947e-04
Peak Time              =         1
```

For multirail cells, you must link the power rails in the design and library to retrieve the information in the library. Use the `create_power_rail_mapping -library_rail` command to map the power rails in the library. For example,

```
pt_shell> create_power_rail_mapping V3 -library_rail Vdd0 -inst "R1"
```

Reporting Power Rail Mapping

If your design has multiple VDD cells, you might want to identify the power rails in the library before you create the power rail mapping between the rails in the design and in the libraries. The `report_power_rail_mapping` command gives you this information. After linking the design and the libraries, this command reports the libraries and their power and ground rails.

After the design instances are grouped with the appropriate power rails, the `report_power_rail_mapping` command reports the mapping information as well. Use this command before or after power rail mapping.

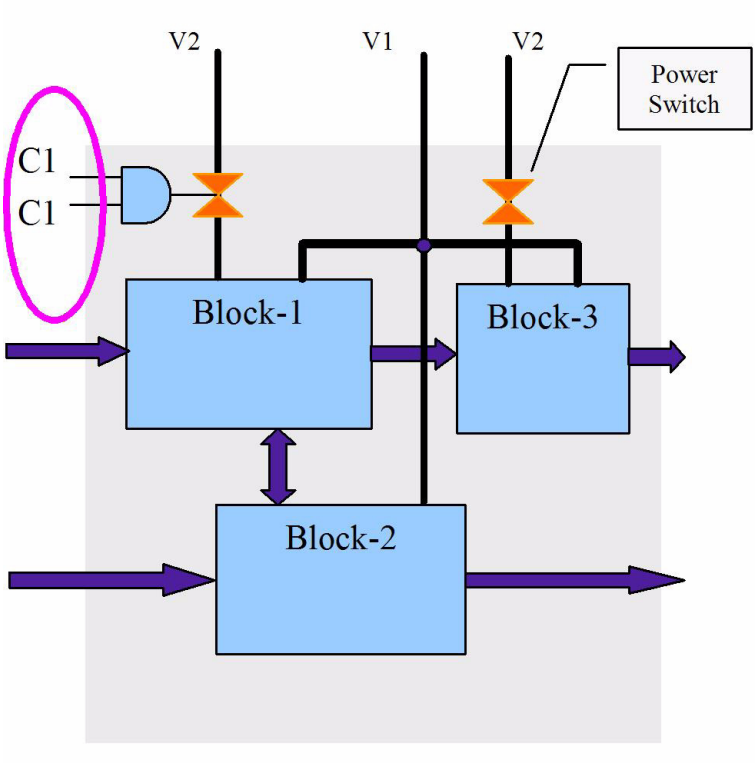
By default, PrimeTime PX reports the total power consumption for all the rails in the design based on the rail voltages defined in the libraries. To report power per rail, PrimeTime PX needs to know the power rails in the design as well as the rails in the library.

Power-Gating Support

Power gating is an effective leakage power-saving technology. It saves power by dynamically cutting off the power supply to certain parts of the design. If a block is not used for a while, its power supply can be cut off.

Figure 7-9 shows a design with multiple power rails. The V2 rail, which feeds Block-3 and portions of Block-1, can be shut off when those instances are not used.

Figure 7-9 Design With Multiple Power Rails



Monitoring the Power-Off Signal

The power switch status affects the power consumption inside the blocks. When these blocks contain retention registers, the state-saving balloon latch within the register cells is powered, whereas the rest of the block power is disabled, thus consuming no power. In this case, the leakage power is minimal and the dynamic power is zero. While the power switch is on, the block works normally. There is both dynamic and leakage power consumption inside the block.

Knowing the on or off status of power switching is important for analyzing the power correctly. Use the `-off_condition` option of the `create_power_rail_mapping` command to capture the power switch status. For example,

```
pt_shell> create_power_rail_mapping V2 -cells "B1 B2" \
              -off_condition "C1 & C2"
```

PrimeTime PX follows certain rules to handle the power-off signal. When the switching activity is from VCD and the power-off condition is true, PrimeTime PX knows that the power supply is in shut-off state with no power consumed. This applies to both leakage and dynamic power. When the power-off condition is false, regular power analysis is performed.

When the switching activity is from SAIF, the tool, by default, uses the static probability of the power-off condition to scale the leakage as default.

If the activity information does not capture the actual toggles during power-off, set the `power_scale_dynamic_at_power_off` variable to `true`. PrimeTime PX scales the dynamic power as well as the leakage power.

Sample Script

For this example, assume that power rail V2 is power-gated as shown in Figure 7-9. The second power rail mapping command is mandatory. It defines the power-off condition and the power rail controlled by power gating. After using that command, you can directly use the `update_power` and `report_power` commands. You can check the rail mapping as well.

```
set link_library "merged.db"
set link_library "* lib1.db"
set link_path_per_instance [list [list {instance1} \
     {* lib2.db}] [list {instance3} {* lib3.db}]]

read_verilog $myvlog
current_design $topdesign
link
read_vcd -strip_path tb/top
vcd.dunp
report_power_rail_mapping

create_power_rail_mapping V1 -library_rail Vdd1 -inst "B1
B2 B3"
create_power_rail_mapping V2 -library_rail Vdd2 \
     -power_off "c1 & c2" -instance "R1"

report_power_rail_mapping

current_power_rail V1
report_power
```

Overriding the Library Voltages

When calculating power for designs with multiple voltage rails, PrimeTime PX checks the value of the `output_signal_level` attribute to calculate the switching power of the net.

For libraries with power and ground pins, PrimeTime PX determines the signal voltages based on the `related_power_pin` and `related_ground_pin` syntax for the library pin.

For partial-swing cases, the switching power calculation takes into account the Vh/Vl voltage swing range for the net.

PrimeTime PX supports the ability to set the values of single and multirail cells and allows you to modify the default rail voltages of single-rail or multirail cells. The values specified are used to calculate the correct energy.

You can use the `set_rail_voltage` command for a particular cell to cause PrimeTime PX to override the default voltage. This is particularly useful for what-if analysis. Another application for `set_rail_voltage` is to apply operating conditions to hierarchical blocks and then back-annotate rail voltages that are due to voltage (IR) drop on individual cells to ascertain the effects.

8

Clock Network Power

This chapter describes how to estimate clock network power consumption, view the clock network, and report power consumption from clock network and registers.

PrimeTime PX provides commands that allow you to view the clock network and connect register power. If the clock network has not been inserted, you can estimate its power consumption. If it has been inserted, PrimeTime PX provides commands to report power consumption from the clock network and registers.

This chapter contains the following sections:

Estimating Clock Network Power Consumption

You can estimate the additional power incurred by a clock network prior to its insertion. You enable this feature with the `estimate_clock_network_power` command, which provides a more accurate analysis of the total design power when clock network synthesis has not yet been performed. Before issuing the `estimate_clock_network_power` command, you must identify the design clocks.

Note:
The `estimate_clock_network_power` command is supported only in the averaged power analysis mode.

Based on the constraints and the type of the buffer you specify, PrimeTime PX builds a virtual balanced clock network and calculates its power consumption. The following guidelines are used to build the clock network:

- The delay from the root to the leaf of the network is minimized.
- The driven registers are put at the same and lowest level of the network.
- The deviation of buffer fanouts at the same network level is minimized.
- For clock-gating cells, by default separate networks are built based on the fanout of the clock-gating cell and using the same rules as specified above. If you want PrimeTime PX to consider clock-gating cells to be part of the clock network, set the `power_clock_network_include_clock_gating_network` variable to `true`. The default value of this variable is false.

The output load of each buffer is calculated using the wire load model, and the input transition of each buffer is propagated from the root. The switching activity is derived from the clock specification or from the SAIF or VCD file.

For more information about the `estimate_clock_network_power` command, see the man page.

The following sample output shows the output of `estimate_clock_network_power`. It includes the power consumption as well as detailed information about the pseudo-clock network, driven registers, and timing propagation.

```
****************************************
Report : estimated clock network power
Design : reg16
Version: Y-2007.06
Date   : Tue May 10 08:41:09 2007
****************************************

Library(s) Used:
    mylib.db (File: /usr/mylib.db)
```

```
Operating Conditions: <lib_default>     Library:mylib.db
Wire Load Model: wlm1

Buffer Used:       BUF1    Library:          mylib.db
Buffer Max Fanout: 4

Power-specific Unit Information:
    Voltage Units = 1V
    Capacitance Units = 1pf
    Time Units = 1ns
    Power Units = 1W

CLOCK: clk1    (source: clk1)
-----------------------------------------------------------

  Clock Period:      20.000000
  Clock Toggle Rate: 0.100000
  Clock Static Prob: 50.000000%

Operating Voltage:      5.000000
  Clock Input Transition: 0.000000 (rise)    0.000000 (fall)

  Number of Driven Regs: 16
  Number of Existing CT Cells: 0
  Number of Buffer Inserted: 5
  Depth of Clock Tree: 2

       Clock Tree Latency            min        ave        max
-------------------------------------------------------------
             rise                  0.0654     0.0654     0.0654
             fall                  0.0648     0.0648     0.0648

  Clock Tree Output Transition       min        ave        max
-------------------------------------------------------------
             rise                  0.0205     0.0205     0.0205
             fall                  0.0139     0.0139     0.0139

  Power Estimation:
    Cell Internal Power  =  5.855e-06
    Net Switching Power  =  1.120e-04
                          ------------
    Total Dynamic Power  =  1.179e-04

    Cell Leakage Power   =  3.965e-08
```

Retrieving Clock Network Objects

Use the `get_clock_network_objects` command to retrieve clock network objects such as nets, cells, pins, and registers in specified clock domains. The command can help you generate customized clock network power reports. The syntax is

```
get_clock_network_objects
      -type object_type
      [-include_clock_gating_network]
      [clock_list]
```

This command provides a view of the clock network of the design, which can aid in debugging. If you don't specify a list of clock domains with *clock_list*, the command returns the specified objects for all clock domains.

By default, this command returns the integrated clock-gating cells unless you specify the `-include_clock_gating_network` option, which causes the discrete logic structure of the network also to be included. For more information, see the man page.

A clock network is a special logic portion of the design that propagates the clocks from the clock sources to the clock pins of latches, flip-flops (which function as anything but propagating clocks), or black boxes. The propagation also stops at design output ports, dangling pins or nets, or the sources of other clocks. The `get_clock_network_objects` command retrieves specified objects from the direct clock network (including the latches, flip-flops, and black boxes driven by the clock network) but not from a generated clock source network, if the specified clock is a generated clock.

Annotating Clock Network Power

Clock network power is one of the major sources of power consumption in a design. When the clock network is synthesized, PrimeTime PX can calculate the clock network power. For gate-level design, PrimeTime PX can estimate the clock network power based on certain parameters controlling the clock tree synthesis. However, the analysis of the clock network power is affected by the clock structure implementation. Designs that use clock mesh to distribute the clock signals generally contain multidriven nets. Estimating clock power for such designs is generally not very accurate. Also, the estimation of the clock network power is affected by the limited physical information available to the tool.

In PrimeTime PX you can specify annotated power values, the ones that are estimated or calculated by other tools such as Design Compiler topographical mode. You can also use the clock network power values obtained from transistor-level simulators and annotate them for power analysis in PrimeTime PX. By specifying the related clock, you can annotate the power of each clock domain separately in the clock network. You can also annotate the total power either as switching power, internal power, and leakage power separately or as total power.

To specify the annotated values for the clock network power, use the `set_annotated_clock_network_power` command. The tool saves the annotated values on the current design. By default, the specified power values are annotated on the entire clock network in the design. Use the `-clock` option for the power values to be annotated on

the collection of clock network objects of the corresponding clock domain. Use the `set_annotated_clock_network_power` command multiple times to specify annotated power for multiple clock domains separately.

Use the `-total_power` option of the command to specify the total power. Alternatively, you can use a combination of `-internal_power`, `-switching_power` and `-leakage_power` options to annotate the values for internal power, switching power and leakage power, respectively. You can use these three options individually or a combination. However you cannot combine any of these three options with the `-total_power` option. For more details of this command and its options, see the command man page.

When you specify annotated values for clock network power, PrimeTime PX replaces the estimated or calculated clock network power by the annotated power values. The succeeding `report_power` command uses the annotated clock network power in the design summary reports. As a result, the power report generated by the `report_power` command is as accurate as those reported by transistor-level simulation tools or the physical aware tools such as the Design Compiler topographical technology.

To remove the previously annotated power values, use the `remove_annotated_clock_network_power` command. For more information, see the command man page.

Reporting Clock Networks and Register Power

To report clock networks, specify the `report_power -groups clock_network` command. To report registers, specify the `report_power -groups register` command. For more information about using and reporting power groups, see Chapter 9, “Generating Reports”.

Use the `report_power -clocks` command to generate a clock-related power report. In addition, use `report_power -include_estimated_clock_network` to report the clock network power estimated by the `include_estimated_clock_network` command. For more information, see Chapter 9, “Generating Reports”.

9

Generating Reports

PrimeTime PX can generate a wide range of reports that provide information about the power consumption.

The following sections describe the mechanisms for generating power reports:

- Power Report
- Power Group Reports
- Sample Power Reports
- Custom Report Generation

The primary command for generating power dissipation reports is the `report_power` command. After you have successfully performed the power analysis with the `update_power` command, you can use the `report_power` command to view the results.

Power Report

By default, the `report_power` command outputs the top-level power consumption. The syntax is

```
report_power [-verbose]
      [-cell_power][-net_power]
      [-hierarchy] [-levels level_value]
      [-power_greater_than threshold] [-leaf]
      [-include_boundary_nets]
      [-nworst_number] [-sort_by sort_mode]
      [-clocks clock_list] [-nosplit]
      [-groups group_list]
      [-include_estimated_clock_network]
      [object_list]
```

You can generate four types of reports:

- Cell-based, with the `-cell_power` option

  Further filter the report by using the `-sort_by` option to sort by `name`, `cell_internal_power` (the default), `cell_leakage_power`, or `dynamic_power`.

- Net-based, with the `-net_power` option

  Further filter the report by using the `-sort_by` option to sort by `name`, `net_static_probability`, `net_switching_power` (the default), `net_toggle_rate`, or `total_net_load`.

- Hierarchy-based, with the `-hierarchy` option

- Summary, when `-cell_power`, `-net_power`, or `-hierarchy` is not specified

The `-groups` and `-clocks` options are filters that apply to all report types. Using these in combination with the `current_instance` command allows you to generate reports per clock domain, power group, and hierarchy.

The `-include_estimated_clock_network` option applies only to summary reports and only when you have previously specified the `estimate_clock_network_power` command.

For information about the `report_power` command, see the man page.

The report file can also provide power dissipation data (power = energy/time) for the desired instances if you provide an instance list with the command.

Typically, you use the power dissipation report to identify general problem areas in a design. You can narrow your analysis by using other PrimeTime PX options.

Example 9-2 shows a summary power report when the `-verbose` and `-include_estimated_clock_network` options are used:

Example 9-2 Report Generated by the report_power -verbose -include_estimated_clock_network Command

```
****************************************
Report : Statistical Average Power
         -verbose
Design : testcase
Version: 2007.06
Date   : Sun Jun 19 15:45:24 2007
****************************************

Library(s) Used:

    power_lib (File: /remote/libraries/power_lib.db)

Operating Conditions:
Wire Loading Model Mode: enclosed

Cell    Design           Wire Loading Model      Library
------------------------------------------------------------
        testcase            0.5K_TLM          power_lib.db
sub     submodule           0.5K_TLM          power_lib.db

Power-specific unit information :
    Voltage Units = 1 V
    Capacitance Units = 1 pf
    Time Units = 1 ns
    Dynamic Power Units = 1 W
    Leakage Power Units = 1 W
  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User defined power group
                Internal  Switching  Leakage    Total
Power Group     Power     Power      Power      Power   (     %)  Attrs
------------------------------------------------------------------
io_pad          0.0000    0.0000    0.0000      0.0000  ( 0.00%)
memory          0.0000    0.0000    0.0000      0.0000  ( 0.00%)
black_box       0.0000    0.0000    0.0000      0.0000  ( 0.00%)
clock_network   0.0000    0.0000    0.0000      0.0000  ( 0.00%)
register        8.442e-05 1.114e-05 9.208e-09 9.557e-05 (29.97%)  i
combinational   0.0000    0.0000    0.0000      0.0000  ( 0.00%)
sequential      0.0000    0.0000    0.0000      0.0000  ( 0.00%)

  Attributes
  ----------
      i  -  Including driven register power
 Internal  Switching  Leakage    Total Clock
 Power     Power      Power      Power   (     %)  Attrs
------------------------------------------------------------------
```

```
clk        1.813e-04  4.199e-05  4.129e-10  2.233e-04
------------------------------------------------------------------
Estimated Clock1.813e-04 4.199e-054.129e-102.233e-04 (70.03%)

  Net Switching Power  = 5.313e-05   (16.66%)
  Cell Internal Power  = 2.657e-04   (83.33%)
  Cell Leakage Power   = 9.627e-09   ( 0.00%)
                         ---------
Total Power            = 3.188e-04  (100.00%)
1
```

Reporting the Intrinsic and Leakage Powers

Using the `report_power` command, you can get the cell leakage power as the sum of gate leakage power and intrinsic leakage power. To get the breakdown of cell leakage power as a sum of gate leakage power and intrinsic leakage power,

- The technology library should have gate leakage and intrinsic leakage (power to ground leakage) characterized for the library cells.
- Set the `power_report_leakage_breakdowns` variable to true. The default value of the `power_report_leakage_breakdowns` variable is false.

The following example shows the part of the power report with the gate leakage and intrinsic leakage information:

```
Net Switching Power  = 1.128e-08   (20.29%)
Cell Internal Power  = 4.311e-08   (77.55%)
Cell Leakage Power   = 1.199e-09   ( 2.16%)
 Intrinsic Leakage  =  4.772e-10
 Gate Leakage       =  7.220e-10
 --------------------------------
Total Power =           5.559e-08  (100.00%)
```

Reporting Potential Toggle Savings From Clock-Gating

The `report_clock_gate_savings` command reports the estimated toggle savings obtained from the clock-gates in a design. You can use this command in both averaged and time-based power analysis modes.

You must apply switching activities to the design before you use this command. In the time-based mode, you can apply switching activity using the `read_vcd` command followed by the `update_power` command. In the averaged power analysis mode you can apply switching activities by using the `read_saif`, `read_vcd` or `set_switching_activity` commands. Also, in the averaged mode, if the switching activities are partially annotated, the tool propagates the switching activities.

The accuracy of the analysis performed by this command on the clock gating depends on the quality of the switching activities that you apply on the design. Simulation can generate switching activities of good quality. Also, the switching activities should be representative of the typical behavior of the design, while covering all modes of operation that are part of the typical behavior of the design.

When you run the `report_clock_gate_savings` command, the generated report provides an overview of each clock domain in the design as shown in Example 9-3 on page 9-7. For each clock domain, the tool computes the toggle savings from clock gating on the clock tree and displays the result as a histogram. Toggle saving is defined as the fraction of the input clock toggles to the clock gates that are prevented from propagating, by the clock-gating cell.

Use the `-by_clock_gate` option to report the estimated toggle savings for each clock-gating cell in the design as shown in Example 9-4 on page 9-8.

Use the `-sequential` option to report the root clock toggle rate and the Q/clock toggle ratio for each register in the design as shown in Example 9-5 on page 9-8. The root clock toggle rate is defined as the clock toggle rate at the clock source. Using the root clock toggle rate and the Q/clock toggle ratio information, you can identify specific registers in your design that might benefit from additional clock gating. Registers in the fastest clock domain that have Q/clock toggle ratios significantly lower than 25% implies possible improvement of the clock-gating for these registers. The possible solutions include adding clock gates or increasing the fraction of the time that the existing clock gates prevent the propagation of the clock toggles. For more details, see the man page.

Use the `-sequential` option to report the register gating efficiency as shown in Example 9-5 on page 9-8. The register gating efficiency represents the fraction of toggles suppressed by clock gating for each register.

Note:

The root clock toggle rate and the register gating efficiency allows you to generate graphical measurements, such as the cumulative toggle savings plot. However, PrimeTime PX does not support generation of these plots.

The `-sequential` option can be used with the `-by_clock_gate` option to report the estimated toggle savings on the register clusters in the design. A register cluster is a set of registers with clock pins driven by the same clock-gating cell. This report can help you identify register clusters that could be re-partitioned, to use separate clock gates. For instance, if some registers in the cluster have different Q/clock toggle ratios than the other registers in the cluster, you can partition the registers into different clusters with a different enable condition for clock gating.

The `-hierarchical` option can be used in combination with either the `-by_clock_gate` or `-sequential` options to isolate specific parts in the design that need further investigation.

The other useful options that can be used to report design related information or display reporting formating:

- Design related options
  - `-clocks`

    Use this option to report on objects in the specified clock domains.
  - `object_list`

    Use this option to report on hierarchical blocks or individual registers or clock-gating cells in the design.
- Reporting format related options
  - `-sort_by`

    Use this option to sort the order of objects in the reports.
  - `-no_split`

    Use this option to avoid splitting the report over multiple lines.

For more details refer to the command man page.

Example 9-3 shows the default report generated by the `report_clock_gate_savings` command.

Example 9-3 Default Behavior of the report_clock_gate_savings Command

```
report_clock_gate_savings

****************************************
Report : Clock Gate Savings
         power_mode: Averaged
Design : mydesign
Version: D-2009.12
Date   : Thu Oct 29 12:08:20 2009
****************************************
--------------------------------------------------------------------
Clock: clk
 + Clock Toggle Rate: 0.392157
 + Number of Registers: 19262
 + Number of Clock Gates: 12
 + Average Clock Toggle Rate at Registers: 0.305872
 + Average Toggle Savings at Registers: 22.0%
--------------------------------------------------------------------
Toggle Savings       Number of            % of
Distribution         Registers            Registers
--------------------------------------------------------------------
100%                  0                     0.0%
80% - 100%            76                    0.4%
60% - 80%             5660                  29.4%
40% - 60%             0                     0.0%
20% - 40%             8                     0.0%
0% - 20%              0                     0.0%
0%                    13518                 70.2%
--------------------------------------------------------------------
1
```

Example 9-4 shows the report generated by the `report_clock_gate_savings` command when you use the `-by_clock_gate` option.

Example 9-4 Report Generated by the report_clock_gate_savings -by_clock_gate Command

```
report_clock_gate_savings -by_clock_gate

****************************************
Report : Clock Gate Savings
         power_mode: Averaged
          -by_clock_gate
Design : mydesign
Version: C-2009.06
Date   : Thu Sep 24 12:08:20 2009
****************************************
--------------------------------------------------------------------
Clock Gate        Toggle            IN clk           OUT clk
                  Savings           Toggle Rate      Toggle Rate
--------------------------------------------------------------------
TOP/A/U24          37.3%              0.5             0.3135
TOP/A/U25          13.3%              0.5             0.4335
--------------------------------------------------------------------
```

Example 9-5 shows the report generated by the `report_clock_gate_savings -sequential` command.

Example 9-5 Report Generated by the report_clock_gate_savings -sequential Command

```
report_clock_gate_savings -sequential

****************************************
Report : Clock Gate Savings
Design : mydesign
power_mode: Averaged
            -sequential
Version: D-2009.12
Date   : Thu Sep 24 14:08:20 2009
****************************************
-------------------------------------------------------------------------
Register      Root Clock   Gated Clock    Q          Q/Clk   Register
              Toggle       Toggle         Toggle     Toggle  Gating
              Rate         Rate           Rate       Ratio   Efficiency
-------------------------------------------------------------------------
TOP/A/U34      0.1          0.08           0.02       25%     20%
TOP/A/U35      0.1          0.05           0.01       20%     50%
-------------------------------------------------------------------------
```

Example 9-6 shows the report generated by the `report_clock_gate_savings -sequential -hierarchical` command.

Example 9-6 Report Generated by the report_clock_gate_savings -sequential -hierarchical Command

```
report_clock_gate_savings -sequential -hierarchical

****************************************
Report : Clock Gate Savings
Design : mydesign
power_mode: Averaged
            -sequential
            -hierarchical
Version: D-2009.12
Date   : Thu Sep 24 14:08:20 2009
****************************************
--------------------------------------------------------------------------
Module      Number of      Q/Clk              Register
           Registers      Toggle Ratio       Gating Efficiency
--------------------------------------------------------------------------
TOP         2250            17.5%               20%
A           1250            15.6%               30%
B            300            23.3%               33%
C           1000            15.6%               40%
--------------------------------------------------------------------------
```

Reporting Power Derate Factors

To report the power derating factors set on design objects use the `report_power_derate` command. In the following example, the report shows the power derating factor set on various objects of the design and on the various power components:

```
*************************************
Report : power derate
         -include_inherited
Design : top
Version: D-2009.12
Date   : Wed Nov 04 17:43:23 2009
*************************************

Object Name Object Type Switching Internal Leakage
--------------------------------------------------------------------------------
top         design      0.50      0.50     0.50
H1          cell (hier) 0.50      0.10     0.50
Inherited               top        --      top

H1/U1       cell (leaf) 0.50      0.20      --
Inherited               top       H1       MY_LIB/IV

MY_LIB/IV   lib_cell     --       0.30     0.30
Inherited                --        --      --
```

Power Group Reports

You can group any set of cells for reporting purposes. These groups are called power groups, and their cells are treated as a virtual block for power calculation and reporting. The cells do not have to be located at the same level of hierarchy. For example, you can place all I/O cells into a group and report I/O cell power, or you can place all cells in the same layout row into a group and report peak power for that row for reliability analysis.

Using Power Groups

To create a user-specified power group, use the following command:

```
create_power_group
      -name name
      [-default] cell_list
```

The following predefined power groups are available by default. Predefined power groups are not overlapping; if one object falls into more than one group, it goes to the group that is ordered first. When defining your own power group, use a unique name.

- io_pad

  All I/O pad cells. The `is_io_pad` attribute is placed on cells identified as I/O pad cells.

- memory

All memory cells. The `is_memory_cell` attribute is applied to these cells only if you previously defined them as part of the memory group in the library. By default, memory cells are considered black box and assigned the `is_black_box true` attribute.

- black_box

  All black boxes excluding memory cells. The `is_black_box true` attribute is placed on cells identified as black box cells.

- clock_network

  Objects in the clock network, excluding I/O pad cells. If the `power_clock_network_include_register_clock_pin_power` variable is set to `true`, the clock network value includes the internal power consumed by register clock pins. If the `power_clock_network_include_clock_gating_network` variable is set to `true`, the clock network value includes the discrete clock-gating logic network data.

- register

  The latches and flip-flops driven by the clock network, excluding memory or black box cells also driven by the clock network. If the `power_clock_network_include_register_clock_pin_power` variable is set to `true`, the register value excludes the internal power consumed by the register clock pins. If the `power_clock_network_include_clock_gating_network` variable is set to `true`, the register value excludes the latches in the discrete clock-gating logic network from the register group.

- sequential

  All other sequential cells.

- combinational

  The combinational logic from register to register, including all the objects left after clock_network, register, memory, io_pad, and black_box objects have been excluded from the design.

To remove predefined or user-specified power groups, use the following command:

```
remove_power_groups -all | power_group_list
```

To return a list of cells contained within specified predefined or user-defined power groups, use the following command:

```
get_power_group_objects power_group_list
```

To report information for specified predefined or user-defined power groups, use the following command:

```
report_power_groups [-nosplit] power_group_list
```

For more information about these commands, see the man pages.

Reporting Power Groups

Specifying `report_power -groups` *`power_group_list`* outputs a report that contains power data for objects in the specified power groups, which can be predefined or user-defined.

Sample Power Reports

The following command generates a net-based power report sorted by net switching power and filtered to display only the five nets with the highest switching power:

```
pt_shell> report_power -net_power -leaf -nworst 5
```

The `-leaf` option causes the report to also include leaf instance power dissipation. Reporting to the leaf level is useful particularly when leaf cells are scattered throughout the hierarchy or when you want to perform detailed analysis.

Example 9-7 Report generated when you use the -net_power -leaf -nworst options of the report_power command

```
****************************************
Report : Averaged Power
         -net_power
         -nworst 5
         -leaf
         -sort_by net_switching_power
        -power_greater_than       0
Design : testcase
Version: B-2008.12
Date   : Mon Nov 17 12:05:44 2008
****************************************

Attributes
----------
      a  -  Switching activity information annotated on net
      p  -  Propagated switching activity information on net
      d  -  Default switching activity used on net
      u  -  Net switching activity uninitialized
      m  -  Net is driven by multiple pins
                          Total      Static   Toggle   Switching
Net                 Vdd   Net Load   Prob.    Rate     Power      Attrs
-------------------------------------------------------------
net36               1.80    0.026    0.248    0.1985   8.397e-06  p
net42               1.80    0.026    0.248    0.1985   8.397e-06  p
net48               1.80    0.026    0.248    0.1985   8.397e-06  p
net54               1.80    0.026    0.248    0.1985   8.397e-06  p
```

```
sub/net20           1.80    0.026    0.248    0.1985   8.397e-06  p
-----------------------------------------------------------------
Total (5 nets)                                          4.199e-05 Watt

1
```

The following command generates a cell-based power report for the predefined clock_network power group:

```
pt_shell> report_power -cell_power -groups clock_network
```

Example 9-7 Report generated by the report_power -cell_power -groups command

```
****************************************
Report : Averaged Power
        -cell_power
        -sort_by cell_internal_power
        -power_greater_than       0
        -groups clock_network
Design : mac
Version: B-2008.12
Date   : Mon Nov 17 12:09:46 2008
****************************************

  Attributes
  ----------
      a  -  Annotated internal & leakage power
      b  -  Black-box (unresolved) cell
      c  -  Clock pin internal power only
      d  -  Does not include clock pin internal power
      h  -  Hierarchical cell

                Internal  Switching  Leakage    Total
Cell            Power     Power      Power      Power   (     %)  Attrs
--------------------------------------------------------------------
sub             3.626e-05 8.397e-06 8.384e-11 4.466e-05 (20.00%)  h
clk_out1_reg    1.228e-05 8.397e-06 8.384e-11 2.068e-05 ( 9.26%)  h
clk_temp1_reg   1.228e-05 8.397e-06 8.384e-11 2.068e-05 ( 9.26%)  h
temp1_reg_3_    5.995e-06    0.0000    0.0000 5.995e-06 ( 2.68%)  c
temp1_reg_0_    5.995e-06    0.0000    0.0000 5.995e-06 ( 2.68%)  c
--------------------------------------------------------------------
Totals          1.813e-04 4.199e-05 4.192e-10 2.233e-04 (100.0%)

1
```

The following command generates a hierarchy-based power report for an instance named sub:

```
pt_shell> report_power -hierarchy -hierarchy_level 2
```

Example 9-7 Report generated by the report_power -hierarchy -hierarchy_level command

```
****************************************
Report : Averaged Power
        -hierarchy
        -hierarchy_level 2
Design : mac
Version: B-2008.12
Date   : Mon Nov 17 12:14:32 2008
****************************************

                        Switch   Int      Leak      Total
Hierarchy               Power    Power    Power     Power    %
------------------------------------------------------------
mac                     1.55e-03 2.23e-03 2.59e-07  3.78e-03 100.0
 mult_21                7.28e-04 5.60e-04 1.49e-07  1.29e-03  34.1
   U1/U9720             2.12e-04 1.31e-04 1.60e-08  3.43e-04   9.1
 add_23                 3.31e-04 2.54e-04 2.36e-08  5.85e-04  15.5

1
```

Custom Report Generation

PrimeTime PX provides numerous attributes that reference power information within the design and the libraries. The following table shows each attribute and the object type to which it can be assigned. These attributes are accessible only after you specify the `update_power` command.

Class	Attribute
net	switching_power glitch_rate glitch_count toggle_rate toggle_count static_probability is_power_control_signal_net power_base_clock activity_source
pin	glitch_rate toggle_rate static_probability power_base_clock (port also)

Class	Attribute
cell	dynamic_power glitch_power internal_power leakage_power peak_power switching_power peak_power total_power x_transition_power power_states peak_power_start_time peak_power_end_time has_multi_power_rails has_multi_ground_rails has_rail_specific_power_tables intrinsic_leakage_power gate_leakage_power
design	dynamic_power glitch_power internal_power leakage_power switching_power peak_power total_power x_transition_power power_simulation_time power_states peak_power_start_time peak_power_end_time power_simulation_time intrinsic_leakage_power gate_leakage_power
lib_cell	has_multi_power_rails has_multi_ground_rails has_rail_specific_power_tables is_memory_cell is_pad_cell is_black_box

You can write your own procedures and rely on the built-in object collections to access the desired information.

The script in Example 9-8 outputs a custom report for power consumption of all registers in the design:

Example 9-8 A Script to Generate a Custom Report

```
proc report_register_power {} {
  set cells [all_registers]
if { [sizeof_collection $cells] == 0 } {
    echo "Error: cannot find any registers.\n"
  } else {
    set t_total       0.0
    set t_dynamic     0.0
    set t_leakage     0.0
    set t_switching   0.0
    set t_internal    0.0
#   print the header
    echo "*********************************************"
    echo "*          Register Power Report              *"
    echo "*********************************************\n"
    echo [format "%10s %10s %10s %10s %10s %s" \
         "Total" "Dynamic" "Leakage" "Switching" "Internal" "Register
Cell"]
    echo "-----------------------------------------------"
#   print the body
    foreach_in_collection cell $cells {
      set name    [get_object_name $cell]
      set total   [get_attribute $cell total_power]
      set dynamic [get_attribute $cell dynamic_power]
      set leakage [get_attribute $cell leakage_power  ]
      set switching [get_attribute $cell switching_power]
      set internal [get_attribute $cell internal_power]
      echo [format "%10.3e %10.3e %10.3e %10.3e %10.3e %s" \
            $total $dynamic $leakage $switching $internal $name]
      set t_total     [expr $t_total + $total]
      set t_dynamic   [expr $t_dynamic + $dynamic]
      set t_leakage   [expr $t_leakage + $leakage]
      set t_switching [expr $t_switching + $switching]
      set t_internal  [expr $t_internal + $internal]
    }
#   print the total
    echo "-----------------------------------------------------"
    echo [format "%10.3e %10.3e %10.3e %10.3e %10.3e TOTAL" \
         $t_total $t_dynamic $t_leakage $t_switching $t_internal]
  }
}
```

Index

D

E

G

I

L

M

N

O

P

R

S

T

U

V

W

X