PrimeTime® PX User Guide

Version L-2016.06, June 2016

SYNOPSYS®

Copyright Notice for the CDPL Common Module

Contents

Preface

This preface includes the following sections:

- About This Guide
- Customer Support

About This Guide

This manual describes the Synopsys 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 the PrimeTime Suite, see the documentation on the Synopsys SolvNet® online support site at the following address:

https://solvnet.synopsys.com/DocsOnWeb

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

- PrimeTime®
- Power Compiler™
- Library Compiler™

PrimeTime PX Tutorials

There are several tutorials that are currently available to assist you when using the PrimeTime PX tool. Check the tutorials directory to see if one (or more) may be pertinent for your applications.

To access the tutorials, enter the following path:

```
$INSTALL_DIR /doc/pt/tutpx/
```

Then, read the PrimeTime_PX_Tutorials.pdf file under the path. There are three tutorials on the topics—averaged power analysis, vector-free analysis, and time-based power analysis. Choose a tutorial and follow the instructions described in the document.

Release Notes

Information about new features, enhancements, changes, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the *PrimeTime Suite Release Notes* on the SolvNet site.

To see the *PrimeTime Suite Release Notes*,

1. Go to the SolvNet Download Center located at the following address:

   https://solvnet.synopsys.com/DownloadCenter

2. Select PrimeTime Suite, and then select a release in the list that appears.

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
`Courier`	Indicates syntax, such as `write_file`.
`Courier italic`	Indicates a user-defined value in syntax, such as `write_file` *`design_list`*.
`Courier bold`	Indicates user input—text you type verbatim—in examples, such as `prompt>` **`write_file top`**
[ ]	Denotes optional arguments in syntax, such as `write_file [-format` *`fmt`*`]`
...	Indicates that arguments can be repeated as many times as needed, such as *`pin1 pin2 ... pinN`*
|	Indicates a choice among alternatives, such as `low | medium | high`
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl 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

The SolvNet site includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The SolvNet site also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access the SolvNet site, go to 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 sign up for an account.

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

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 support case to your local support center online by signing in to the SolvNet site at https://solvnet.synopsys.com, clicking Support, and then clicking “Open A Support Case.”
- 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 North America.
  - Find other local support center telephone numbers at

    http://www.synopsys.com/Support/GlobalSupportCenters/Pages

1

Introduction to the PrimeTime PX Tool

The PrimeTime PX tool is an add-on feature to the PrimeTime tool 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.

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

- The PrimeTime PX Tool Features
- Power Modeling Overview

The PrimeTime PX Tool Features

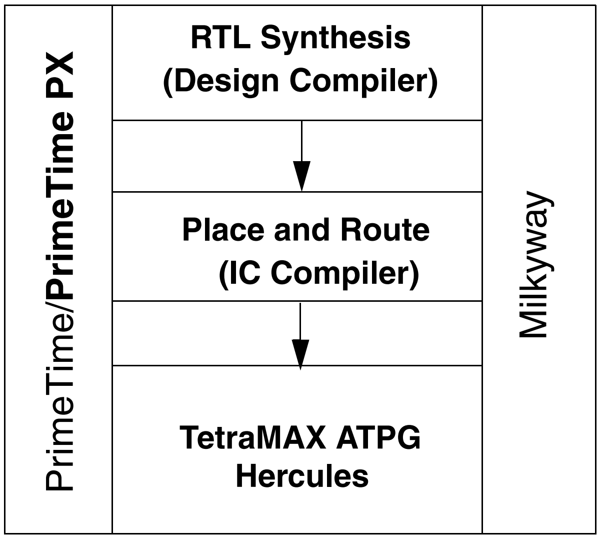
The PrimeTime PX tool provides vector-free and vector-based peak power and average power analysis. The vectors to PrimeTime PX are either RTL or gate-level simulation results in the Value Change Dump (VCD) format, Fast Signal Database (FSDB) format, or Switching Activity Interchange Format (SAIF).

The tool provides multivoltage and power domain analysis. It also has an integrated graphical user interface (GUI) for visual power debugging.

The PrimeTime PX tool 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. The library can be a nonlinear power model (NLPM) or a Composite Current Source (CCS) library. It calculates the power behavior for a circuit at the cell level and reports the power consumption at the chip, block, and cell levels.

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

Figure 1-1 The PrimeTime PX Tool Full-Chip Power Analysis


PrimeTime/PrimeTime PX
RTL Synthesis (Design Compiler)
Place and Route (IC Compiler)
TetraMAX ATPG Hercules
Milkyway

Power Analysis Techniques Performed in the PrimeTime PX Tool

The techniques for power analysis using the PrimeTime PX tool are:

- Averaged power analysis

  For purely averaged power analysis, the tool 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, the tool supports analysis based on the RTL or gate-level simulation activity over time.

  The PrimeTime PX tool uses an event-driven algorithm to calculate the power consumption for each event. The tool generates detailed time-based power waveforms to provide both average and peak power results and also reports the power results.

Power Modeling Overview

This section describes power modeling in the following topics:

- Leakage Power
- Dynamic Power, which includes internal power and switching power
- Setting the Power Derating Factor
- Generating Power Models

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

The tool calculates switching power based on the voltage, netlist capacitance, and switching of the nets.

The leakage and dynamic power calculation results are used 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 subthreshold 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.

The PrimeTime PX tool supports the analysis of the gate leakage power. The tool uses the tables in the logic library to determine the leakage power for the cells in the design. When the logic library is characterized for subthreshold leakage and gate leakage, PrimeTime PX can report these leakage components for cells.

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

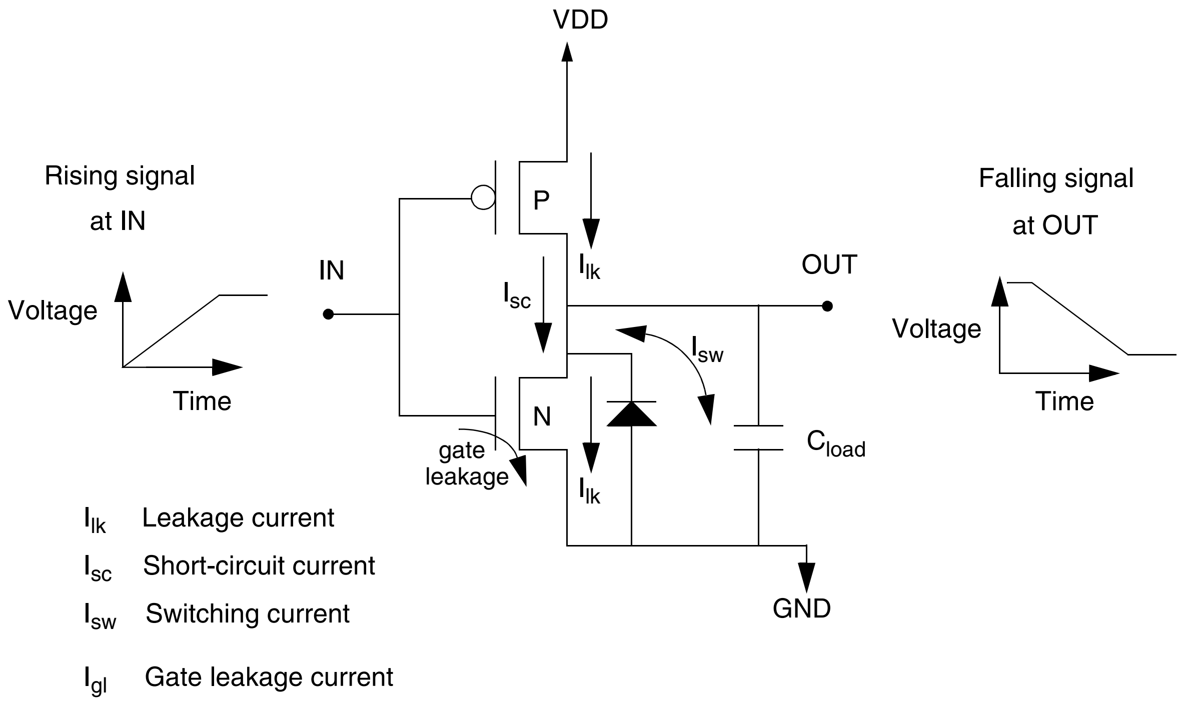
Internal Power

Internal power is the dynamic power dissipated within the boundary of a cell. It includes the power dissipation due to charging or discharging of capacitances internal to the cell during switching; and the power dissipation due to the momentary short circuit between the P and N transistors of a gate while both are turned on.

Figure 1-2 considers a simple gate to show the cause of short-circuit power. 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



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.

For simple library cells, the internal power consumption is primarily due to the short-circuit power. For complex library cells, however, the dominant source of internal power might be due to the charging and discharging of internal capacitance.

Library developers can model internal power by using the internal power library group.

For more information about modeling internal power, see the *Library Compiler Timing, Signal Integrity, and Power Modeling User Guide*.

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 is the result of the logic transitions at the output of the cell, switching power increases as the 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.

Setting the Power Derating Factor

Use the power derating feature to apply a multiplication factor to the switching power, internal power and leakage power. You can apply the power derating factor to a design, cell, library cell, hierarchical, and or leaf cells.

The power derating feature is supported in both the averaged and time-based power analysis modes. The derating factor that you specify affects the result of your power analysis.

Commands Supporting Power Derating

To set, reset and report the power derating factors, use the following commands:

- `set_power_derate`

  Set power derating factors for each rail on different power components for the current design, 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. A default of 1.0 is used by the tool, if you do not specify any power derating factor.

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

- `reset_power_derate`

  Reset all the specified rail-based power derating factors for 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 information about the `reset_power_derate` command, see the man page.

- `report_power_derate`

  List rail and PG pin power derating factors. If the derating factor is applied to a rail in the current design, the rail is listed as an object to which the derating factor is applied when set at the design level or on hierarchical cells. If the derating factor is applied to a leaf cell, the object type is the PG pin to which the rail is connected.

  For a detailed example, see “Generating Reports on Power Derating Factors” on page 9-21.

Specifying the Power Derating Factor on Design Objects

Use the `set_power_derate` command to specify the power derating factor for a rail on different objects in the current design or the library, hierarchical, and leaf cells. You can also specify derating factors on the following power components:

- Internal power
- Switching power
- Leakage power

The PrimeTime PX tool applies the power derating factors on the following objects with a precedence order from high to low:

- Design rail

  When not specified, the tool applies scaling factors to all rails in the design.

- Leaf cell and power group

  The power derating feature treats a collection of cells as a power group.

- Hierarchical cell
- Library cell
- 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.

  Use the `-derate` option to display the power derating factors used in the power calculation. This option applies only to the cell-based averaged power report and time-based power report. For multiple-voltage cells with different derating factors specified per PG pin, the derating value is calculated by dividing the sum of the original derated values.

  The `report_power` command generates:

  ❍ The derated averaged 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 the peak power report.

- `report_power_calculation`

  Uses the power derating factors to calculate and report power values.

- `reset_design`

  Resets the power derating factors while resetting the design.

- `estimate_clock_network_power`

  Applies the specified power derating factors to the buffer or the current design while estimating the clock network power. This command is supported only in the averaged power analysis mode.

- `get_attribute`

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

Generating Power Models

The PrimeTime PX tool 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 the PrimeTime tool. The power model is generated by incorporating power information into the ETM. Perform timing analysis and power analysis on your design before generating the power model. Based on the result of the analysis, use the `extract_model -power` command to generate the power models that contain both timing and power data.

When generating the power model from the gate-level design, the PrimeTime PX tool associates the dynamic power of the gate-level design with the clock pins of the generated power model. The dynamic power of the gate-level design is mapped into the internal power of the generated power model within the tool. 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 tool annotates the switching activity of internal clock pins of ETMs during power analysis of hierarchical designs. This feature is useful for analyzing the power consumption of complex macro cells, which is a function of internal clock frequencies. This is supported in both the averaged and time-based power analysis modes.

For more information, see “Specifying the Activity File” on page 6-2.

Limitations in Generating Power Models

The limitations when generating the power model are:

- Multiple modes of operation are not supported in the generated power model.
- Block scope checking for power is not supported.
- The `extract_model -power` command writes out the power model only in the .lib format. To generate the .db equivalent of the model, use the Library Compiler or Design Compiler tool.

2

Power Analysis Flow in the PrimeTime PX Tool

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

- Overview
- Power Analysis Inputs
- Power Analysis Flow
- Saving and Restoring PrimeTime PX Sessions

Overview

The PrimeTime PX tool supports two types of power analysis modes—averaged and time-based. 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, select the required power analysis mode and the specific options supported by the selected mode.

The user interface supports variables, commands, and command options for performing power analysis. Provide the switching activity information in the SAIF, VCD or FSDB file format. Irrespective of the mode you select and the options you choose, power analysis is performed when you run the `update_power` command.

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

Power Analysis Inputs

The PrimeTime PX tool requires the following data for performing power analysis:

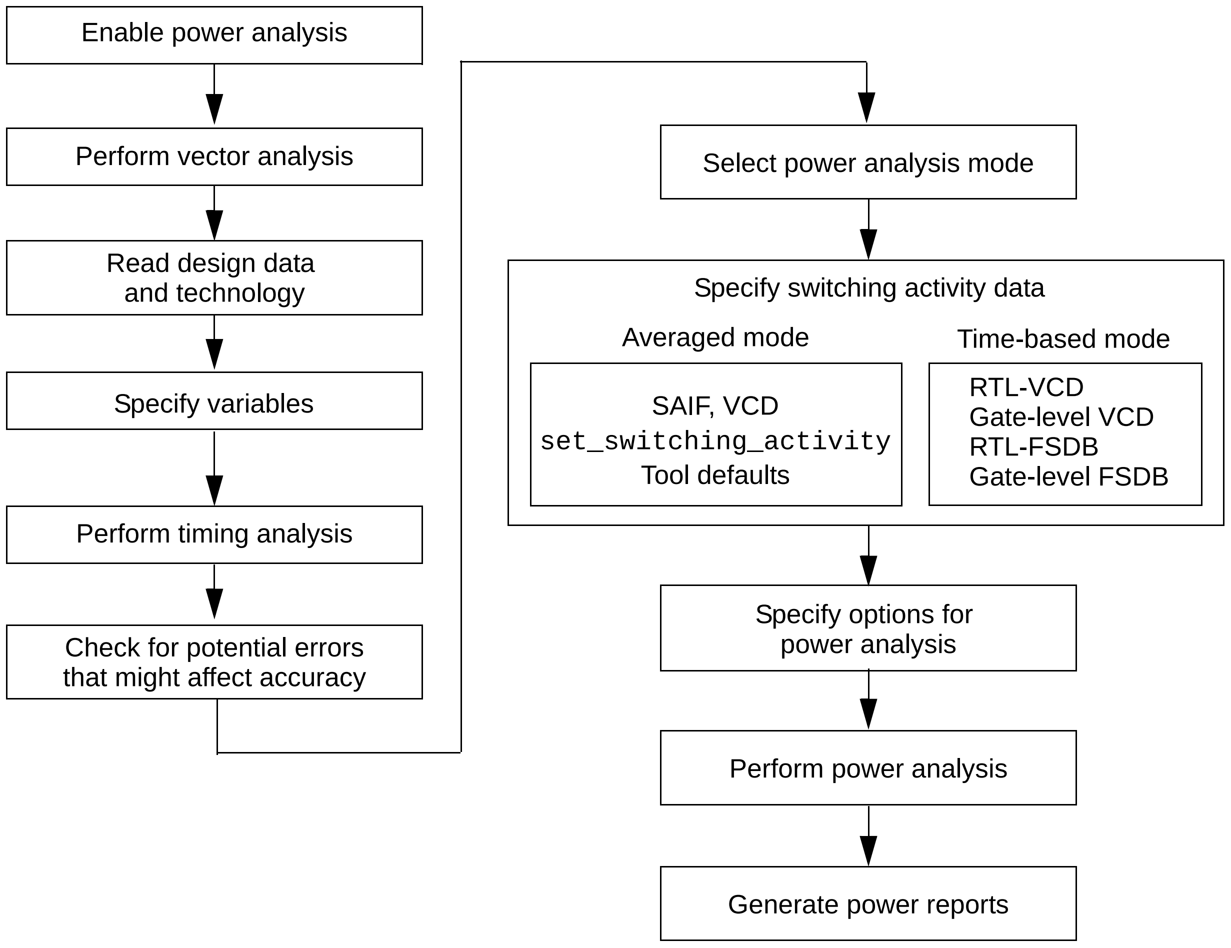
- Logic library: A cell library containing timing and power characterization information for each cell. If both Composite Current Source (CCS) and nonlinear power model (NLPM) libraries are available, use the `power_model_preference` variable to specify which library to use. The default is `nlpm`.
- Gate-level netlist: A flat or hierarchical gate-level netlist in Verilog, VHDL, or Synopsys database format, containing leaf-level instantiation of the library cells.
- Design constraints: An SDC file containing design constraints to calculate the transition time on the primary inputs and to define the clocks.
- Switching activity: The design switching activity information for averaged power analysis or accurate peak power analysis.
- Net parasitics: A parasitics file (SPEF) containing net capacitances for all the nets.

Power Analysis Flow

Figure 2-1 shows the steps for performing power analysis, as described in the following sections:

1. Enabling Power Analysis
2. Performing Vector Analysis
3. Reading the Design Data and Logic Library
4. Specifying Variables
5. Timing Analysis
6. Checking for Potential Errors That Might Affect Power Accuracy
7. Selecting the Power Analysis Mode
8. Specifying Switching Activity Data
9. Specifying Options for Power Analysis
10. Performing Power Analysis
11. Generating Power Reports

Figure 2-1 Power Analysis Flow



Enabling Power Analysis

To enable power analysis and review the power data, set the `power_enable_analysis` variable to `true`. The default is `false`. A PrimeTime PX license is required to enable this variable.

Performing Vector Analysis

If an event-based activity file (either in the VCD or FSDB format) is available, you can qualify the activity before reading the design data using the vector analysis feature. During vector analysis, the tool plots the average toggle rate per interval of the signals in the activity file so that you can identify the time windows of high activity likely to produce peak power. Use the `write_activity_waveforms` command to generate activity waveforms from the activity file.

For more information, see “Performing Vector Analysis” on page 6-6.

Reading the Design Data and Logic Library

The tool supports netlists in Verilog, VHDL, .db, .ddc, and Milkyway formats. The logic library must be in the .db (Synopsys database) format. Design data includes parasitic information, such as port and net loads, wire load models, back-annotated parasitics, and transition time information. For averaged power analysis, the design data includes the design constraints compiled with the PrimeTime tool in a Synopsys Design Constraints (SDC) file.

For more information, see the PrimeTime documentation.

CCS and NLPM Data

PrimeTime PX supports both CCS power model and NLDM data. Both the CCS and NLPM data can coexist in a cell description in the .lib file. That is, a cell description can have either NLPM data, CCS data, or both.

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

Use the `power_model_preference` variable to specify your power model preference when the logic library contains both the NLPM and CCS data. The default is `nlpm`.

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

Setting Operating Modes for Power Models

You can define operating modes, such as read and write modes, for a power model by specifying the mode definition in the library.

The following example uses the `mode_name` and `mode_value` attributes (in bold) to define conditional data modeling in the `mode_definition` group.

```
library (mylib3) {
  delay_model : table_lookup;
  cell(cell_3) {
    pg_pin(VDD) {
      voltage_name : VDD;
      pg_type : primary_power;
    }
    pg_pin(VSS) {
      voltage_name : GND;
      pg_type : primary_ground;
    }
    mode_definition(rw) {
      mode_value(read) {
      when : "A1";
      sdf_cond : "A1 == 1";
    }
      mode_value(write) {
      when : "!A1";
      sdf_cond : "A1 == 0";
    }
  }
  pin(A1) {
    direction : input;
    related_power_pin : VDD;
    related_ground_pin : VSS;
  }
  leakage_current() {
    mode(rw, read);
    pg_current(VDD) {
      value : 1.0;
    }
    pg_current(VSS) {
      value : -2.0;
    }
    gate_leakage(A1) {
      input_high_value : 0.5;
      input_low_value : -0.5;
  }
}
```

For more information, see the Library Compiler documentation.

Specifying Variables

When you specify the operating conditions for power analysis using the `set_operating_conditions` command, the PrimeTime PX tool uses the appropriate set of parameter values from the logic library. These parameter values generally include fabrication process, operating temperature, and power supply voltage as characterized by the ASIC vendor.

Power Calculation Variables

When you specify the `power_limit_extrapolation_range` variable, the tool controls some aspects of the power calculation for both event- and toggle-based analysis. The default is `false`.

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

When deriving the capacitive load used for the power analysis of designs using deep sub-micron and low-voltage technologies, set the `power_use_ccsp_pin_capacitance` variable to `true` to consider the Miller Effect. By default, the variable is set to `false` and the capacitive load is calculated by averaging the C1 and C2 values derived for timing analysis. When the variable is set to `true`, the capacitive load used for power analysis is more pessimistic because it includes the gate-to-source and gate-to-drain capacitance contributions due to the Miller Effect.

Timing Analysis

During power analysis, the PrimeTime PX tool first performs timing analysis if the timing is not updated and uses the transition time for calculating accurate power and timing window for calculating averaged power waveforms. Timing analysis includes the signal integrity effects if you enabled the signal integrity feature. When the timing data is available for power analysis, the tool uses the timing data for power calculation instead of recalculating them, thereby saving the runtime.

Before specifying the switching activity for power, perform timing analysis using the `update_timing` command. This maximizes the performance by preventing any additional timing updates triggered by the activity annotation commands.

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. 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 no option is specified. To update your list of default checks, update this variable or use the `-override_defaults` option.

The `-override_defaults`, `-include`, and `-exclude` checklist options define the power checks to be used. When you specify the `check_power` command, the tool performs out_of_table_range, missing_table, no_arrival_window, no_base_clock, and missing_function checks. By default, the PrimeTime PX tool performs the out_of_table_range and missing_table checks.

Selecting the Power Analysis Mode

Use the `power_analysis_mode` variable to select the power analysis mode. Set this variable before using any of the power commands, as follows:

```
set_app_var power_analysis_mode averaged | time_based
```

If you do not set this variable, by default, the PrimeTime PX tool performs averaged power analysis.

The tool loses the switching activity and power information if you modify the value of the `power_analysis_mode` variable after power analysis. To perform power analysis, you must reread the switching activity data.

Power Analysis Techniques

In CMOS circuits, the statistical average power analysis is performed early in the design flow for total power consumption. Accurate event-based peak power analysis for each event is performed closer to the signoff for voltage (IR) drop analysis. Both the analysis techniques require design switching activity on every node. Activity is defined as how often a net switches with respect to a specified clock period.

The PrimeTime PX tool performs both averaged power analysis and peak power analysis. The tool calculates static and dynamic power for both power analysis types.

Using Multiple Cores for Power Analysis

Using multiple cores for power analysis improves the runtime by dividing large tasks into smaller tasks for processing.

Note:
Multiple core processing for power analysis is enabled only in the averaged analysis mode.

To enable multiple core analysis, use the `set_host_options` command. For example, to enable the use of four cores to run power analysis,

```
pt_shell> set_host_options -max_cores 4
```

The maximum number of cores you can specify is 16. However, each license supports only four cores. If you specify five or more cores, the tool checks out another license.

Use the `report_host_options` command to identify the number of cores specified.

For more information about the `set_host_options` and `report_host_options` commands, see man pages. For more information about using multiple core processing, see the *PrimeTime User Guide*.

Specifying Switching Activity Data

Specify the switching activity to annotate the activity information. The switching activity source can be a gate-level VCD file, a RTL VCD file, a SAIF file, or an FSDB file. The activity can also be defined by the user-defined commands for annotation, such as the `set_switching_activity` and `set_case_analysis` commands.

If the switching activity is obtained from an RTL simulation, specify a name mapping file to match the RTL activity file names to the corresponding gate-level objects. The default name mapping performed in the Design Compiler tool is correctly matched using the built-in name mapping capability in the PrimeTime PX tool.

When the switching activity information is available, you must annotate this information appropriately on the design objects, before you can use the switching activity information for power analysis.

For more information about specifying net switching activity information for averaged power analysis and time-averaged power analysis, see “Switching Activity Annotation” on page 5-3 and “Specifying the Activity File” on page 6-2, respectively.

For more information about name mapping, see Chapter 4, “Name Mapping.”

Performing Conditional Power Analysis

The PrimeTime PX tool performs conditional power analysis in both averaged and time-based power analysis modes when you specify an event-based activity. Using this feature, you can analyze power for specific modes of operation, or for specific simulation time windows with the following options to the `read_vcd` or `read_fsdb` command:

- Use the `-time` option to identify the simulation window for which power analysis must be performed.
- Use the `-when` option to specify a Boolean condition for signals in the event file. The PrimeTime PX tool calculates power only for those time windows where the condition evaluates to `true`.

Specifying Options for Power Analysis

After selecting the power analysis mode, use the `set_power_analysis_options` command with the options to control the behavior of the power analysis mode. 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 generates an error message. When you specify multiple settings by using the command multiple times, the latest setting overwrites the previous ones.

Table 2-1 lists the commonly used options for averaged and time-averaged power analysis modes. For a complete option list, see the man pages.

Table 2-1 Commonly Used Options for Averaged and Time-Averaged Power Modes

Option	Averaged Power Mode	Time-Averaged Power Mode
`-static_leakage_only`	Yes	No
`-waveform_interval`	No	Yes
`-waveform_format`	No	Yes
`-waveform_interval`	No	Yes
`-waveform_output`	No	Yes
`-include`	No	Yes
`-include_groups`	No	Yes
`-multi_rail_waveform`	No	Yes
`-cells`	Yes	Yes

Table 2-1 Commonly Used Options for Averaged and Time-Averaged Power Modes (Continued)

Option	Averaged Power Mode	Time-Averaged Power Mode
`-sdpd_tracking`	Yes	Yes
`-sdpd_tracking_cells`	Yes	Yes

Performing Power Analysis

To perform power analysis, run the `update_power` command. Based on the mode that you specify along with the `set_power_analysis_options` command, the PrimeTime PX tool performs power analysis.

When you run the `update_power` command, power analysis is triggered consistently for various power analysis modes. After the analysis, the `update_power` command generates the power data.

In the averaged mode, the tool uses the default settings when no activity data is available. However, in the time-based mode, you must specify the activity data in the event-based activity file format before using the `update_power` command. Otherwise, the PrimeTime PX tool generates an error message.

For more information, see Chapter 5, “Averaged Power Analysis,” and Chapter 6, “Time-Based Power Analysis.”

Generating Power Reports

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

Example 2-1 shows the power report in the averaged power analysis mode:

Example 2-1 Power Report Generated During the Averaged Power Analysis Mode

```
****************************************
Report : Averaged Power
****************************************
 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
```

Example 2-2 shows the power report with peak power for time-based power analysis. You can also use the nWave waveform viewer to view the power waveforms.

Example 2-2 Peak Power Report Generated During the Time-Based Power Analysis Mode

```
****************************************
Report : Time-Based Power
...
****************************************

 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
```

You can also generate your own reports using the Tcl constructs and accessing the power attributes. To save and restore your power parameters, set the `power_enable_analysis` variable to `true`.

For more information, see Chapter 9, “Generating Reports.”

Saving and Restoring PrimeTime PX Sessions

Use the `save_session` and `restore_session` commands to save the current state of a PrimeTime PX tool session and restore the same session. To examine the results of an earlier session of power analysis, use the save and restore feature to avoid repeating the execution of the time-consuming `update_power` command. The `restore_session` command begins from the same analysis point where you last saved your session, and uses only a small fraction of the original runtime.

The tool version used to save and restore a PrimeTime PX tool session must be the same. You can locate the version used to save the session from the README file located in the session directory. You can also restore any PrimeTime or PrimeTime signal integrity timing analysis sessions and continue the power analysis.

For more information about saving and restoring a session, see the *PrimeTime User Guide* and the corresponding man pages.

3

Simulation Activity Formats

This chapter describes the types of simulation activity file formats. It contains the following sections:

Capturing Switching Activity

The PrimeTime PX tool annotates switching activity from both the RTL and gate-level simulation.

Early in the design cycle, use RTL simulation to explore your design and identify,

- Which RTL architecture consumes the least power?
- Which module consumes the most power?
- Where is power being consumed within a specified block?

Later in the design cycle, use gate-level simulation instead of RTL simulation to annotate specific nets of your design or all the elements of your design for better accuracy.

Table 3-1 summarizes the various methods of capturing switching activity:

Table 3-1 Comparing Methods of Capturing Switching Activity

Simulation	Captured	Not captured	Trade-offs
RTL	Synthesis-invariant elements	1. Internal nodes 2. Correlation of non-synthesis-invariant elements 3. Glitching 4. State and path dependencies	Fast runtime at expense of some accuracy
Zero-delay and unit-delay gate-level	1. Synthesis-invariant elements 2. Internal nodes 3. Correlation 4. State dependencies 5. Some path dependencies	1. Some path dependencies 2. Glitching	More accurate than RTL simulation, but significantly higher runtime
Full-timing gate-level	1. All elements of design 2. Correlation 3. State and path dependencies	Highest accuracy, but runtime can be very long	Correlation between primary inputs

Switching Activity Formats

Specify both the RTL and gate-level switching activity in any of the following formats:

- SAIF

  SAIF stands for Switching Activity Interface Format. The SAIF files capture signal transitions and the time spent at each logic level. The SAIF file contains the toggle counts and static probabilities for the nets in the design.

  SAIF is supported only in the averaged power analysis mode. Use the `read_saif` command to annotate the activity information in the SAIF format.

- VCD

  VCD stands for Verilog Change Dump format. It is an event-based format that contains every value change for the signals in the design and the time at which they occurred. Use the `read_vcd` command to specify the VCD file in both the averaged and time-based analysis modes.

  Use the `read_vcd` command to specify the activity files in other event-based formats within the PrimeTime PX tool. Internally, the tool converts these formats to the VCD format using the appropriate utility.

  Table 3-2 lists the formats, extensions, and conversion utilities supported by the tool:

Table 3-2 Supported File Formats, Extension, and Conversion Utilities

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 $NOVAS_HOME/bin (if $NOVAS_HOME environment variable is set to Verdi installation)
Bzip2	.bz2	bunzip2

Note:
If your file is of the unsupported format, the tool treats it as a VCD file and reads the data directly.

- FSDB

  FSDB is a binary event-based format.

  - When specified an FSDB file with the `read_vcd` command, the tool converts the FSDB data to the VCD format.
  - When specified an FSDB file with the `read_fsdb` command, the tool reads the FSDB file activity directly.

All the formats are supported in the averaged power analysis mode. In the time-based mode, specify the simulation activity in the VCD or FSDB format.

Note:
When you specify switching activity from multiple sources, the tool honors the latest one.

- Activity Files Generated from a SystemVerilog Simulation

  Use the activity files generated from a SystemVerilog simulation if the files are in VPD or .fsdb format. Specify the format as SystemVerilog using the `-format` option of the `read_vcd` or `read_fsdb` command.

SAIF File Format

For specifying switching activity data, two types of SAIF files are supported:

- RTL SAIF

  If the gate-level simulation data is not available, provide RTL SAIF files that contain the switching activity of the primary inputs, hierarchical ports, and synthesis-invariant elements, such as sequential elements.

  For the tool to perform correct switching activity annotation, you must specify the correct name mapping technique.

- Gate-Level SAIF

  If the gate-level SAIF file contains activity annotations for all the elements in the design, the PrimeTime PX tool can accurately calculate the average power consumption of the design.

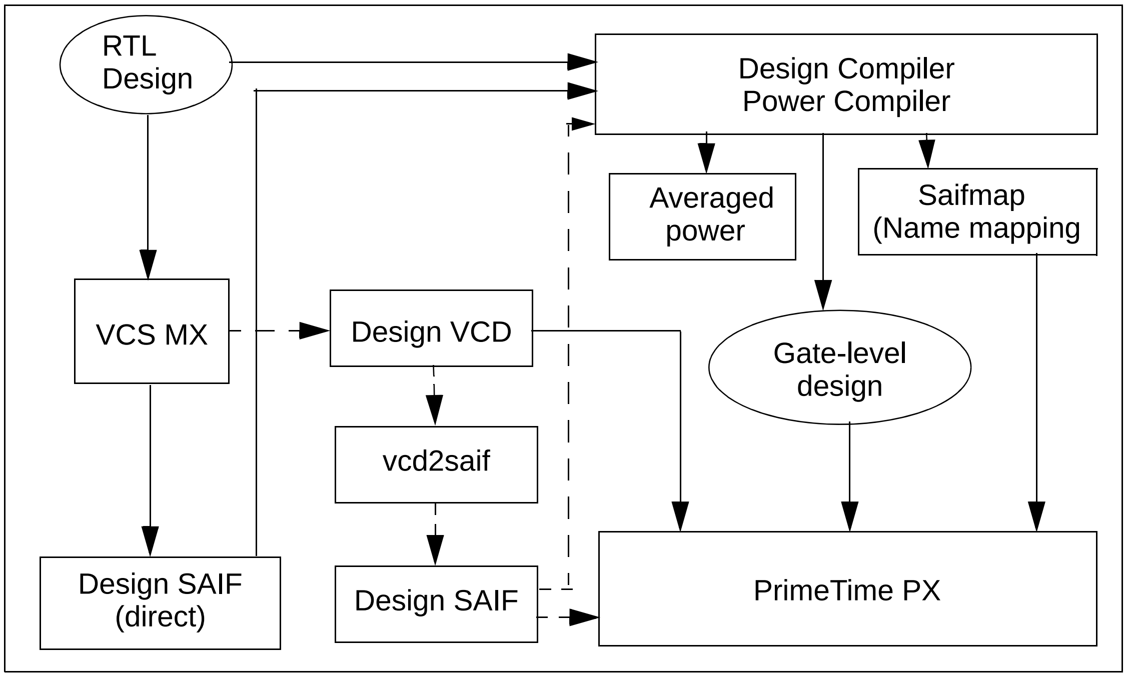
Use the `read_saif` command to specify the activity information in the SAIF file format generated from RTL or a gate-level netlist.

Generating SAIF Files

You can generate a SAIF file either from the RTL simulation or the gate-level simulation. This section discusses both RTL and gate-level simulations using Synopsys VCS. VCS supports Verilog, SystemVerilog, and VHDL formats.

Figure 3-1 shows two ways of generating a SAIF file. The solid lines indicate the suggested SAIF flow while the dotted lines indicate the alternative method of SAIF flow using various Synopsys tools.

Figure 3-1 SAIF File Generation and Its Usage With Various Synopsys Tools



When specifying RTL-based activity file, read the SAIF file into the Power Compiler tool to generate a name mapping file, which matches the RTL object names to the gate-level objects. Then, read the name mapping file in the PrimeTime PX tool before annotating the RTL-SAIF or VCD files. For more details on mapping RTL-to-gate, see Chapter 4, “Name Mapping.”

The following sections describe the ways of generating SAIF files:

- Generating SAIF Files From Simulation
- Generating SAIF Files From VCD Output Files

Generating SAIF Files From Simulation

The VCS MX tool can generate SAIF file directly from simulation. This direct SAIF file is smaller in size relative to the VCD files. Your input design for simulation can be an RTL or gate-level design. Also the design can be in Verilog, SystemVerilog, VHDL, or mixed HDL formats. When your design is in Verilog or SystemVerilog format, you must specify system tasks to VCS MX using the SAIF generation procedures.

When generating the SAIF file during simulation, use the default monitoring method. This monitoring method captures the switching activity of only the synthesis-invariant objects, such as ports, tristate cells, black box cells, flip-flops, latches, retention registers, and hierarchical cells other than clock-gating cells. Integrated clock-gating cells and latch-based isolation cells are not synthesis invariant objects and therefore not captured.

If the library forward-SAIF file contains details of state and path dependencies, the backward SAIF file generated by the tool also contains these details.

For more information about generating SAIF files using these methods, see the *Power Compiler User Guide*.

When you run the `read_saif` command, the tool annotates the simulation activity in the SAIF file to the gate-level objects. Use the `report_switching_activity` command to generate a report showing the percentage of nets annotated and the source of annotation.

Generating SAIF Files From VCD Output Files

Use the `vcd2saif` utility to convert the VCD or other event-based format activity files generated by VCS into SAIF files. To generate the SAIF file and annotate the switching activity, use the following procedure:

1. Run the simulation to generate a VCD output file.
2. Use the `vcd2saif` utility to convert the VCD output file to a SAIF file.
3. Annotate the switching activity within the SAIF file.

The `vcd2saif` utility converts the RTL or gate-level VCD file generated by the VCS tool into a SAIF file. The utility is provided with the PrimeTime PX tool installation and is located in the install_dir/bin path.

The `vcd2saif` utility supports the same event-based activity formats as the `read_vcd` command, shown in Table 3-2 on page 3-3.

The `vcd2saif` utility does not extract state-dependent and path-dependent switching activity when converting VCD to the SAIF format. For information about each option, use the `vcd2saif -help` command.

VCD File Format

During the timed-based analysis, you must provide the event-based activity for the design using only the VCD, VPD, or .fsdb formats. The PrimeTime PX tool calculates power for every event for accurate results.

For specifying the activity data, two types of VCD files are supported by the tool:

- RTL VCD

  Use the `read_vcd -rtl` command to read the activity file. With the `-rtl` option, the tool performs name mapping and event propagation.

  Use the `read_vcd -zero_delay` command if the VCD file is generated from a zero-delay simulation.

  You can also use the `set_rtl_to_gate_name` command to map the RTL and gate-level object names. As 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 causing inaccurate results. To avoid this, use the `set_rtl_to_gate_name` command.

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

- Gate-level VCD

  When you read the VCD file without specifying the `-rtl` or `-zero_delay` option with the `read_vcd` command, the tool assumes that the file is a gate-level VCD file.

Generating VCD Files

Use the VCS and other HDL simulators to generate the event-based activity files. These simulators use standard PLI procedures.

For information on how to generate VCD, VPD, and FSDB activity files, refer to the VCS manual.

Reading VCD Files

In the time-based power analysis mode, the PrimeTime PX tool does not support reading activity information from multiple VCD files. If you specify multiple VCD files, the tool uses only the latest VCD file specified.

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

FSDB File Format

Use the `read_fsdb` command to specify event-based switching activity information in the FSDB format. PrimeTime PX uses the switching activity information to annotate activities and to calculate power for each event in the design.

When the `read_fsdb` command is executed, the tool first reads the header of the activity file to determine which portion of the design is annotated in the time-based power analysis flow. The remainder of the activity file is not read until power analysis is performed.

The following are the commonly used options when specifying an event-based activity file in the FSDB format:

`-rtl`

Indicates if the input FSDB file is generated from the RTL simulation. When specified, the `read_fsdb` command uses name mapping set by the `set_rtl_to_gate_name` command. The `update_power` command also uses zero delay simulation to create events on unannotated nets.

Using an FSDB file generated from RTL simulation is recommended, so the sequential cells, primary inputs, and black box outputs are annotated, while combinational nets are not. Events on these unannotated nets must be generated via simulation.

`-zero_delay`

Indicates if the input FSDB file is generated from the zero-delay gate-level simulation. When specified, power is averaged over each clock cycle. You need to specify the Synopsys Design Constraint (SDC) file, so that the clocks and transition time at primary inputs are defined. When not specified, most events happen at clock edges, which might leads to large unrealistic peak power.

`-time` *`{start_time end_time start_time end_time ...}`*

Specifies time windows in which the activities are counted for power calculation. Define the window list with an even number list in an increasing order. For example, `-time {10 20 50 70}` specifies time windows `[10ns`, `20ns]` and `[50ns`, `70ns]`. If you do not know when the simulation time ends, use a negative number, such as `-time {10 -1}` for an activity time window from 10 ns to the end of simulation time.

Note that the specified time window is automatically adjusted by the tool based on the timescale in the FSDB file or the value specified with the `-interval` option of the `create_power_wave` command. For example, if the `-interval` option is specified as `10`, the tool changes `-time {13 33}` to `-time {10 30}`.

`-when`

Specifies a Boolean condition to determine which simulation time from the FSDB file is considered for power analysis. When specified, the time selected is chosen automatically based on the logical value in the FSDB file.

For more information, see man pages.

Examples

The following example reads the FSDB file dump.fsdb and uses the first 100 ns of the events in the file.

```
pt_shell> read_fsdb dump.fsdb -time {0 100}
```

The following example reads the FSDB file dump.fsdb and excludes time periods when the specified condition is false.

```
pt_shell> read_fsdb dump.fsdb -when {net125 && net126}
```

Debugging Problems With the Activity Files

The following errors might occur when you use the activity files for power analysis:

1. The hierarchy in the activity file does not match that of the current design.

   In this case, the total number of annotated nets and cells in the summary report generated using the `read_vcd` command report is zero. The tool generates warning messages for all the nets in the design that do not exist in the activity file.

   This hierarchy mismatch generally occurs when you incorrectly set the `-strip_path` option of the `read_vcd` and `read_fsdb` commands. Therefore, the activity file does not contain all levels of design hierarchy.

2. Syntax errors occur when parsing the activity file because of

   ❍ Unsupported file format. For example, the PrimeTime PX tool generates syntax error messages when you specify a VCD file in the unsupported format, such as an enhanced VCD (EVCD) file.

   ❍ Unmatched format or version between conversion utilities and activity files. The PrimeTime PX tool invokes the appropriate utilities to convert the switching activity data to the VCD format as applicable. The utility must be available in the search path and matches the version of the data.

3. Name mapping of the RTL to gate-level design object is incorrect.

   Specify a name mapping file that contains a list of the `set_rtl_to_gate_name` commands to avoid the error.

4

Name Mapping

This chapter describes the name mapping mechanisms used in the PrimeTime PX tool to match the RTL activity to the gate-level netlist, when annotating the RTL simulation activity files. It contains the following sections:

Overview

During synthesis, the name manipulations might cause the RTL synthesis invariants (register outputs, primary inputs, tristates, and black boxes) to be renamed. To match the RTL activity to the gate-level components, the PrimeTime PX tool supports the following mapping mechanisms in a precedence order from high to low:

1. The name mapping `set_rtl_to_gate_name` command
2. Exact name mapping
3. Default name mapping

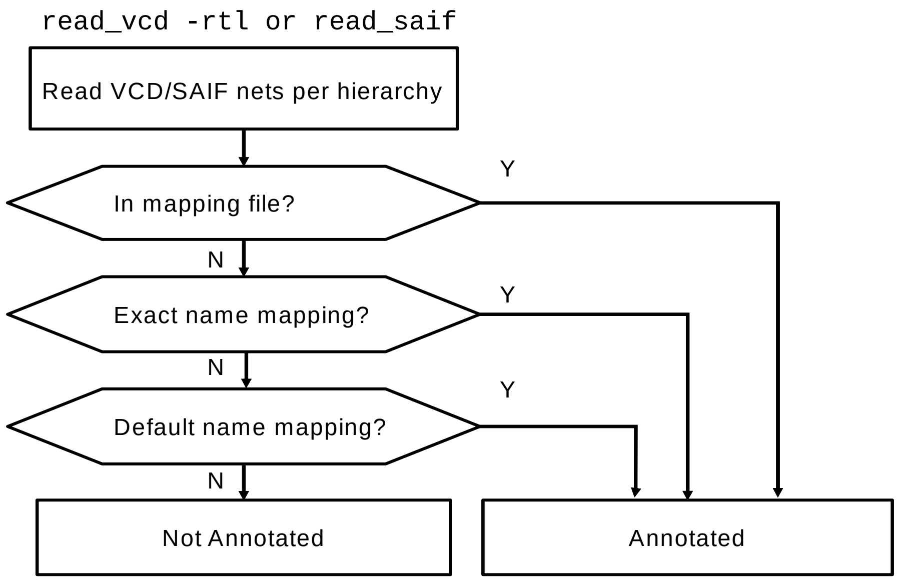
When annotating the RTL activity to the gate-level objects, the PrimeTime PX tool uses the `set_rtl_to_gate_name` command to match RTL to gate-level objects. During synthesis, using the Design Compiler tool, you can track the name changes and generate a map file that contains the `set_rtl_to_gate_name` command.

If the map file is unavailable, define the mapping using the `set_rtl_to_gate_name` command in a user-defined file. If the user-defined file is not available, the tool checks for the gate-level names if they exactly match the RTL simulation names. If not, the tool uses a built-in name mapping mechanism that matches the default name-manipulation supported by the Design Compiler tool.

If the tool does not find a corresponding gate-level object, it does not annotate the object in the RTL activity file.

Figure 4-1 indicates the name mapping flow.

Figure 4-1 Name Mapping Flow



Name Mapping Command

The PrimeTime PX tool uses the `set_rtl_to_gate_name` command to perform the name mapping of RTL to gate-level objects. Using this command, you can specify the unique name mappings and global substitutions. When you specify this command, the tool creates a name mapping database. The tool evaluates the name mapping database when reading the SAIF or VCD simulation activity files. Specify the `set_rtl_to_gate_name` command before the `read_saif` or `read_vcd -rtl` command.

The following example shows the mapping between an RTL activity file object, a, and the gate-level netlist object, a_reg.

```
pt_shell> set_rtl_to_gate_name -rtl {a} -gate{a_reg}
```

When mapping RTL registers to gate-level sequential cells, the tool applies the annotation to the Q pin, and the inverse to the QB pin.

Generating the Name Mapping File

When performing synthesis using the Design Compiler tool, use the `saif_map` command to generate a name mapping file. The command tracks the name changes throughout synthesis, and generates a mapping file for the PrimeTime PX tool. The generated map file helps to maximize the RTL annotation in the PrimeTime PX tool, and thereby the accuracy and performance of power analysis.

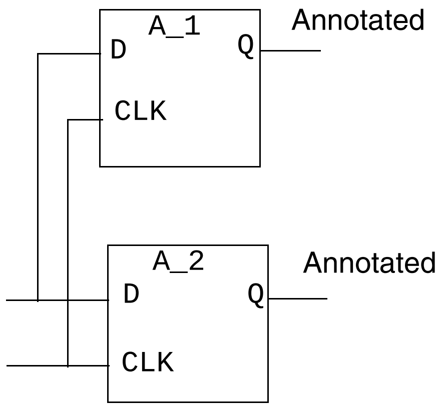
For more information about the name mapping file, see *Power Compiler User Guide*.

Mapping RTL to Multiple Gate-Level Objects

Use the `set_rtl_to_gate_name` command to map an RTL object to multiple gate-level objects. This helps to maximize the switching activity annotation where the activity of the same RTL object is applied to multiple ports, nets, and cells for accurate power analysis.

Figure 4-2 shows how an RTL object is mapped to multiple gate-level objects.

Figure 4-2 Mapping RTL-to-Multiple Gate-Level Objects



All the map file entries for the RTL object are applied due to the one-to-many mapping support. Mapping RTL-to-multiple gate-level objects improves the switching activity annotation for the following cases:

- IC Compiler clock tree synthesis buffer insertion

  When you annotate the buffers inserted by IC Compiler clock tree synthesis, the multiple mapping capability enables the annotation of the RTL clock to all the inserted buffers.

- Design Compiler manual replication

  When you annotate the replicated instances generated by Design Compiler, the multiple mapping capability supports annotation of all the replicated instances.

- Clock tree synthesis port punching

  When you annotate the additional clock ports of the same RTL clock, the multiple mapping capability supports annotation of all the clock ports.

- Hierarchical flow

  When you annotate the block-level activity and map multiple instances of a hierarchical block, the multiple mapping capability annotates all the instantiated blocks without modifying the block-level map file.

If you repeatedly map one instance of an RTL-to-gate object or map multiple RTL objects to one gate-level object, the PrimeTime PX tool honors only the first mapping definition and generates warning messages for the subsequent instances of the mapping.

Also, if you specify switching activity inversion, you can map an RTL object to some gate-level objects with switching activity inversion and to the other gate-level objects without switching activity inversion.

You can also specify annotation for a block-level RTL activity to multiple instances of the block in a design as shown in Example 4-1.

Example 4-1 Mapping One RTL Object to Two Gate-Level Objects at the Block-Level

```
set_rtl_to_gate_name -rtl u0 -gate u0_1            # 1st mapping of RTL object u0

set_rtl_to_gate_name -rtl u0 -gate u0_1            # Duplicate entry for u0->u0_1
                                                   # generates warning

set_rtl_to_gate_name -rtl u1 -gate u1_1
set_rtl_to_gate_name -rtl u2 -gate u2_1

set_rtl_to_gate_name -rtl u0 -gate u0_2 -inverted  # 2nd mapping of RTL object u0 with
                                                   # activity inversion

set_rtl_to_gate_name -rtl u1 -gate u1_2

set_rtl_to_gate_name -rtl u2 -gate u2_2

set_rtl_to_gate_name -rtl u2 -gate u0_2            # Conflicting mapping of gate
                                                   # object u0_2 with RTL u0 and u2.
                                                   # Generate a warning. Mapping
                                                   # to RTL u0 is honored.
```

As shown in the Example 4-1,

- You cannot remap RTL object u0 with the gate u0_1.
- You can also map the RTL object u0 with the switching activity inverted u0_2 gate.
- You cannot map the same gate-level object u0_2 with multiple RTL objects u2 and u0. In this case, only the first mapping is honored.

Exact Name Mapping

To annotate the RTL objects that are not specifically mapped using the `set_rtl_to_gate_name` command, the PrimeTime PX tool searches for gate-level objects with the same name in the gate-level netlist. For example, with the exact name mapping, the RTL net A is mapped to the gate-level net A.

Case Sensitivity

By default, the exact name matching is case insensitive; for example, the RTL net A is mapped to the gate-level net a. The case sensitivity is controlled by the `power_read_activity_ignore_case` variable. The default is `true`. When reading the activity files generated from VHDL simulation, set the variable to `false`, because VHDL is case-sensitive.

Disabling Exact Name Mapping of RTL Activity to Gate-Level Nets

To disable the exact name mapping of the RTL activity to gate-level nets, use the `power_disable_exact_name_match_to_net` variable.

Synthesis might introduce combinational nets with inverted logic while retaining the original name. If these nets are not listed in the name mapping file, by default the PrimeTime PX tool performs exact name mapping of the RTL activity to the gate-level netlist. This causes the non-inverted values to be annotated. To avoid inadvertent annotation of the RTL activity, which is no longer functionally equivalent to the gate-level net with the same name, set the `power_disable_exact_name_match_to_net` variable to `true`. By default, the setting is `false`. The exact name mapping is applicable to the cells and pins, but is disabled for nets.

The tool uses the exact name mapping method to annotate the nets connected to the hierarchical pins, because they have the same name as the pin, and exist at the same hierarchical level. To disable the exact name mapping to the hierarchical pins, set the `power_disable_exact_name_match_to_hier_pin` variable to `true`.

Default Name Mapping

If no exact match is found, the tool performs default name mapping to match the RTL to gate-level names. The default mapping technique considers the netlist flattening and the default name modifications performed during synthesis.

The PrimeTime PX tool performs default name mapping, when

- RTL hierarchy is mapped to the flattened gate-level objects by replacing the hierarchical separator “/” with “_” and removing the “\” in front of the flattened net names. For example, the hierarchical RTL net a/b/c is mapped to the flattened gate-level net, a_b_c.
- RTL registers are mapped to the gate-level objects appended with _reg. By default, the Design Compiler tool appends _reg to the RTL register names which are synthesized into sequential cells. For example, the RTL register, a, is mapped to the gate-level cell a_reg. As with mapping performed with the `set_rtl_to_gate_name` command, the default name mapping technique applies the annotation to the Q pin of the sequential, and its inverse to the QB pin.
- Bus dimension Verilog separators [ ] are mapped to “_”. For example, the RTL register reg a[7] is mapped to the gate-level object a_reg_7_.

Name Mapping Reporting

To verify the name mapping of the RTL activity to the gate-level objects, use the `report_name_mapping` or `report_activity_file_check` command.

The `report_name_mapping` command generates a report showing the RTL and the gate-level mapping applied using the `set_rtl_to_gate_name` command. If the gate-level objects specified using the `set_rtl_to_gate_name -gate` command do not exist in the netlist, the mapping is not stored in the database and cannot be used.

The `report_name_mapping` command reports the value of the inverted flag for each name mapping entry as shown in Example 4-2.

Example 4-2 A Block-Level Mapping Report Generated by the report_name_mapping Command

```
pt_shell> report_name_mapping
**********************************************************************
Report : report_name_mapping
**********************************************************************
Attributes
----------
         v - Inverted
----------------------------------------------------------------------
RTL Name     Gate-Level Name    Type     Attributes
----------------------------------------------------------------------
u0           u0_1               cell              # 1st mapping for u0
u1           u1_1               cell
u2           u2_1               cell
u0           u0_2               cell      v       # 2nd mapping for
                                                  # u0 with inversion
u1           u1_2               cell
u2           u2_2               cell
------------------------------------------------------------------------
```

The `report_activity_file_check` command lists all the RTL names in the activity file, and reports

- The name mapping mechanism used for annotation.
- The name mapping mechanism that is user-specified (with the `set_rtl_to_gate_name` command), exact name mapping, default name mapping, or a combination of these methods.
- Whether the logic inversion is specified using the `set_rtl_to_gate_name -inverted` command, or detected by default in the tool when the inversion attribute indicates the logic inversion between the RTL object and the gate-level object.
- The exact gate-level object to which the annotation is applied. Example 4-3 shows the report example indicating the RTL object mapping to a cell, a net, or a pin.

Example 4-3 A Block-Level Mapping Report Generated by the report_activity_file_check Command

```
pt_shell> report_activity_file_check -strip_path top/M1 -path M1 \
              myvcd.vcd -find A221_m
*************************************************************************
Report : report_activity_file_check
…
Activity File :
myvcd.vcd ...
*************************************************************************
Mapping Method
--------------
         d - Default name mapping applied
         e - Exact name matching applied
         m - User-defined RTL-to-gate name mapping applied
         s - User-defined mapping by string substitution
-----------
Attributes
----------
v - inversion
-----------------
Activity File    Activity Design    Design    Name         Attributes
File             File     Object    Object    Mapping
Object Type      Name     Type      Name      Method
--------------------------------------------------------------------------
module           u0       Hier Cell u0_1      m
module           u0       Hier Cell u0_2      m
signal           u0/x     Net       u0_1/x    e,m
signal           u0/x     Net       u0_2/x    e,m
…
```

Resetting the Name Mapping Database

To avoid any inadvertent annotation of the RTL to gate-level objects due to the exact name mapping, you can disable the exact name mapping of RTL objects to the nets and hierarchical pins. If any annotation is missing, modify the name mapping database created by the `set_rtl_to_gate_name` command. Be sure to clear the database using the `reset_rtl_to_gate_name` command before you reload a modified name mapping file and reread the RTL activity file.

5

Averaged Power Analysis

This chapter describes how to annotate switching activity and how the PrimeTime PX tool uses the annotated activity to calculate power in the averaged power analysis mode.

In the averaged power analysis mode, the tool calculates averaged power based on toggle rates. The sources of annotated activity can be the default toggle rates, user-defined switching activity, SAIF files, or VCD files generated from simulation. To calculate the power consumption for the complete design, the tool uses zero-delay simulation to propagate the switching activity to non-annotated nets.

This chapter contains the following sections:

Overview of Averaged Power Analysis

In the averaged power analysis mode, the switching activity consists of toggle rates and static probabilities. The accuracy of the power analysis results is proportional to the accuracy of the switching activity values. Therefore, the switching activity obtained from the RTL or gate-level simulation must be annotated correctly.

The tool supports the annotation of simulation-based switching activity file formats: VCD, FSDB, VPD, and SAIF. If activity files are not available, use the user-defined switching activity commands to provide realistic activity for accurate analysis results.

To calculate power in the averaged analysis mode, the tool uses the state-dependent and path-dependent leaf-cell pin toggle rates and static probabilities. If not available, the tool first checks for switching activity on all the nets in the design; It assigns the defaults for all the black box outputs and primary inputs without switching activity. For other non-annotated nets, the tool uses implied switching activity or derives the switching activity from zero-delay simulation of random vectors that match the toggle rates and static probabilities of the annotated nets.

From the net switching activity, the tool estimates the state-dependent and path-dependent leaf-cell pin toggle rates and static probabilities to calculate power.

Types of Switching Activity

Switching activity in the averaged mode is defined by the following parameters:

- Static Probability

  Static probability (SP) is the probability that a signal is at a specific logic state; it is expressed as a number between 0 and 1. SP1 is the static probability that a signal is at logic 1. Similarly SP0 is the static probability when the signal is at logic 0.

  The static probability can be calculated as a ratio of the time period, for which the signal is at a certain logic state relative to the total simulation time. For example, if SP1 = 0.70, the signal is at logic 1 state 70 percent of the time. The tool uses SP1 when modeling switching activity.

- Toggle Rate

  The toggle rate is the number of logic-0-to-logic-1 and logic-1-to-logic-0 transitions of a design object per unit of time, such as a net, a pin, or a port. The toggle rate is denoted by TR.

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 when 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, meaning the signal is high for one half the time period and low for the other half. This default value might be appropriate for data bus lines but for signals, such as `reset` or `test_enable`.

  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 toggle rates on output pins of leaf cells

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

Switching Activity Annotation

The accuracy of the power analysis is proportional to the accuracy of the switching activity. You must specify the switching activity generated from the RTL or gate-level simulation for performing power analysis. If activity files are not available, use the user-defined switching activity commands to provide realistic activity for accurate analysis results.

In the averaged power analysis mode, the PrimeTime PX tool supports the following methods for annotating switching activity:

- Switching Activity File Annotation
  - Annotation of RTL or gate-level VCD files with the `read_vcd` command

    See Annotating Switching Activity Using Event-Based Activity Files for more information.

  - Annotation of RTL or gate-level SAIF files with the `read_saif` or `merge_saif` command

    See Annotating Switching Activity Using SAIF Files for more information.

  - ❍ Annotation of RTL or gate-level FSDB files with the `read_fsdb` command

- User-Specified Activity Annotation
  - ❍ Annotation of individual design objects with the `set_switching_activity` command

    See Annotating Activity Using the set_switching_activity Command for more information.

  - ❍ Annotation of constant values on register set and reset pins with the `infer_switching_activity -apply` command

    See Inferred Switching Activity for more information.

  - ❍ Annotation of constant values with the `set_case_analysis` command

    See Annotating Constant Nets for more information.

  - ❍ Annotation of constant values via netlist logic constants
  - ❍ Clock net annotation with the `create_clock` command

    See Annotating Clocks for more information.

Following is a list of the activity annotation commands in a precedence order from high to low:

- The `read_vcd`, `read_fsdb`, `read_saif`, and `set_switching_activity` commands have equal priority. If the activity for the same object is specified with these commands, the most recent annotation overrides the earlier annotations.
- The `set_case_analysis` command.
- The `create_clock` command.

  For ETMs containing internal clock pins, the PrimeTime PX tool annotates the switching activity based on the relationship to an external clock as defined in the model. When you specify the `create_clock` command to the external clock pin, the tool annotates the internal clock pin with relative activity.

  Note:
    If you specify the `set_switching_activity -type` command, the command does not override the user-defined clock.

- Netlist logic constants.

Annotating Switching Activity Using Event-Based Activity Files

A VCD or an FSDB file stores logic value changes in a design. Use the `read_vcd` and `read_fsdb` commands to specify an RTL or gate-level activity file in either VCD or FSDB format, respectively.

In the averaged power analysis mode, when you specify an event-based activity file, the tool by default extracts the toggle rates and static probabilities from the file and applies them to the corresponding design objects. Use the `-strip_path` option to specify the path to the current design instantiated in the simulator environment. For example, the following example reads a VCD file, my.vcd, which instantiates the current design as TB/DUT:

```
pt_shell> read_vcd -strip_path TB/DUT my.vcd
```

The `read_vcd` command supports the following features during averaged power analysis:

- Supports name mapping when annotating VCD activity for both gate-level and RTL event-based activity files, either with or without the `-rtl` option.
- Supports conditional power analysis with the `-when` option. When you specify this option, the tool extracts the switching activity from the VCD file. The annotated toggle rates reflect only the activity for those time windows.

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

When state-dependent and path-dependent tracking is enabled, you can specify only one VCD file. When tracking is disabled, which is the default, you can apply activity from multiple VCD files. For more information, see “Extracting Directed State-Dependent and Path-Dependent Activity” on page 5-6.

After a successful execution, the `read_vcd` command generates 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 input pins. The following example shows a summary generated by 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%)
======================================================================
```

As shown in the example, there are 1625 nets in the design. If a net appears at different levels of the design hierarchy, the net is counted only once.

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 leaf cell pins.

Extracting Directed State-Dependent and Path-Dependent Activity

By default, the activity is extracted from an event file as net toggle rates and static probabilities. However, you can use the PrimeTime PX tool to extract state-dependent path-dependent toggle rates for large macro cells for more accurate power analysis results.

In the directed state-dependent and path-dependent analysis mode, the tool analyzes the average power using the actual state-dependent and path-dependent activity data of the specified cells. For the rest of the cells, the tool uses the estimated state-dependent and path-dependent activity data.

To use the directed state-dependent and path-dependent analysis mode, run the `set_power_analysis_options –sdpd_tracking enabled` command before running the `read_vcd` command.

Running a directed state-dependent and path-dependent power analysis consists of the following steps:

1. Identify large macro cells.

   The tool identifies a list of large macro cells to analyze. By default, the list includes memory and black box power group objects. You can also specify cells using the `set_power_analysis_options –sdpd_tracking_cells` *`list_of_cells`* command.

   You must specify cells that have state-dependent and path-dependent power tables. The total power values of the cells can induce significant differences in the averaged and time-based power analyses. For example, cells that have a large difference in the state-dependent and path-dependent energy values should be included in the list.

   To view the cells that have state-dependent and path-dependent tracking enabled, use the `report_power_analysis_options` command.

   Here is an example:

```
pt_shell> report_power_analysis_options
Option Name           Value
-----------------------------------
cells                 undefined
static_leakage_only   false
sdpd_tracking         enabled
sdpd_tracking_cells   mem1, mem2
```

2. Annotate large macro cells.

   The PrimeTime PX tool extracts the toggle rates and static probabilities for the nets in the design, and the state-dependent and path-dependent data for the ports of the large macro cells. They are annotated for averaged power analysis.

   You can extract the state-dependent and path-dependent data only when all the ports of the tracked cells are in the VCD file.

3. Analyze the averaged power of the design.

   After annotation, the PrimeTime PX tool estimates the state-dependent and path-dependent activity for the standard cells and computes the averaged power of the design.

Annotating Switching Activity Using SAIF Files

The PrimeTime PX tool uses the following ways to read and write the SAIF files:

- Reading SAIF Files Using the read_saif Command
- Reading SAIF Files Using the merge_saif Command
- Writing SAIF files Using the write_saif Command

Reading SAIF Files Using the read_saif Command

Use the `read_saif` command to apply the switching activity from both the RTL and gate-level SAIF files.

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

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

Reading SAIF Files Using the merge_saif Command

Use the `merge_saif` command to read switching activity information from multiple SAIF files. When you specify a weight for each input SAIF file, the tool annotates a weighted sum of the switching activities. Use the command for flows where different SAIF files are generated for different modes of the same design. The switching activity from all the different modes can be used for power calculations. For example, if your design has the standby, slow, and fast modes, the standby.saif, slow.saif, and fast.saif SAIF files are generated for these modes, respectively.

Based on the expected usage of the design, specify the following weight to the SAIF files:

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

Use the `merge_saif` command to read and weight the SAIF files. Use the `merge_saif -output` command to generate a SAIF file containing the weighted sum of the switching activities.

When you use the `merge_saif` command, the tool reads and propagates the switching activity for each SAIF file separately; and then applies the weights. The output SAIF file contains the resulting state-dependent and path-dependent gate-level activity.

Use the `-simple_merge` option to prevent the separate propagation of each SAIF file when merging gate-level SAIF files which contain the activity for all the nets in the design. When you specify this option, the tool weighs the activity in each SAIF file and merges before propagating activity and output in the SAIF file specified with the `-output` option.

You cannot use the SAIF files with state-dependent and path-dependent information with the `merge_saif` command.

Writing SAIF files Using the write_saif Command

When you use the `write_saif` command, the PrimeTime PX tool writes out SAIF files, containing user-annotated and propagated switching activity.

User-Specified Activity Annotation

When simulation activity files are not available, you can annotate the activity using the `set_case_analysis`, `create_clock`, `infer_switching_activity` and `set_switching_activity` commands.

For accurate results, set the values for these parameters appropriately based on your design. The guidelines for annotating switching activity are:

- Annotate a suitable default toggle rate value, for example, 0.2, to all the starting points. The value that you specify must generate a pessimistic power value compared to the gate-level VCD results.

  ```
  pt_shell> set power_default_toggle_rate 0.2
  ```

- Annotate the reset signals by applying a constant value of 0. Use this for high-fanout nets, such as reset and scan enable signals, because the default toggle rates cannot be applied.

  ```
  pt_shell> set_case_analysis 0 reset
  ```

- Annotate the toggle rates to the register outputs, based on the related clocks using the `-clock_domains` option. Use this option when the propagation through sequential elements results in the loss of activity information.

  ```
  pt_shell> set_switching_activity -type registers -toggle_rate value \
              -clock_domains {all_clocks} -hierarchy
  ```

- Annotate the clock-gating cells with a factor that is a multiple of the toggle rate of the related clock. When you specify the `-clock_domains` option, you can also use the `-clock_derate` or `-toggle_rate` option to ensure that the clock gate output toggles at the expected rate.

  ```
  pt_shell> set_switching_activity -clock_domains clk2 \
              -clock_derate value -type clock_gating_cells -hierarchy
  ```

Annotating Clocks

The PrimeTime PX tool annotates the switching activity on clock objects defined using the `create_clock` or `create_generated_clock` command.

The following example indicates the clock port `clk1` that toggles twice every 25 ns:

```
pt_shell> create_clock -period 25 clk
```

You can also define clocks on internal clock pins using the `create_clock` command.

Annotating Activity Using the set_switching_activity Command

If the toggle rate information from simulation is not available, use the `set_switching_activity` command to specify the switching activity on the design objects. Specify the toggle rate and static probability to the specific objects or object types, and apply the switching activity to the objects within a specified clock domain.

Here are the commonly used options for annotating switching activity on nets, ports, and pins. For a detailed description about how to use the command, see the man pages.

`-toggle_rate`

Set the toggle rate that represents the number of glitch-free transitions either from low to high, or high to low, during the time period specified with the `-period`, the `-base_clock` or the `-clock_domains` option. If neither the `-period` nor the `-base_clock` option is specified, the tool assumes the default library time unit as the period.

`-static_probability`

Specify the percentage of time a signal stays at logic state 1.

`-period`

Specify the time period for applying the toggle rate.

`-clock_domains`

Specify the period to which the toggle rate applies. When you specify this option, the toggle, which is relative to the period of the clock domain, is assigned to the objects within the listed clock domain.

The clock domain is determined by the value of the pin or the net attribute `power_base_clock`, representing the related clock. The tool derives the value from the SDC commands, by tracing the clock network and the fanin-to and fanout-from nets where the `power_base_clock` attribute is known. When an object has more than one related clock, by default the tool assigns the fastest clock to the `power_base_clock` attribute.

To manually assign a clock to the `power_base_clock` attribute, use the `set_power_base_clock` command. To remove the manual setting, use the `reset_power_base_clock` command.

To specify the default `power_base_clock` assigned to an object with no related clock, set the value of the `power_default_base_clock` variable to the desired clock.

For more information about the commands, see the man pages.

`-base_clock`

Indicate that the toggle rate is relative to the named clock, which can be any design clock. When you specify this option, the tool controls the toggle rate applied to the object and does not modify the value of the `power_base_clock` attribute.

Examples

In the following example, the tool applies a toggle rate of 0.1 to all the primary inputs of the design within the clk1 clock domain.

```
pt_shell> set_switching_activity -type inputs -toggle_rate 0.1 \
          -clock_domains {clk1}
```

If the period of the clock clk1 is 25ns, the tool applies the toggle rate 0.1 to all the inputs in the clk1 clock domain, which is 0.004 toggles per ns. Internally, the tool converts all toggle rates to an absolute value relative to the default library time unit and stores the value as the `toggle_rate` attribute. The toggle rate of the clock clk1 is therefore 2 toggles/25ns = 0.08 toggles per ns.

In the following example, the tool assigns a toggle rate of 0.3 to the activity of net n1, relative to the period of the clk2 clock.

```
pt_shell > set_switching_activity -toggle_rate 0.3 -base_clock clk2 n1
```

If the clk2 clock has a period of 10 ns, then the toggle rate of n1 is 0.3/10ns = 0.3.

Annotating Constant Nets

The PrimeTime PX tool annotates the nets tied to power or ground in a netlist with a toggle rate of 0 and a static probability of 1 or 0. To apply constant values to other design objects, use the `set_case_analysis` command.

In the following example, a constant value of 0 is applied to the signal reset.

```
pt_shell> set_case_analysis 0 reset
```

The toggle rate of the reset signal is 0 and the static probability is 0.

Inferred Switching Activity

When the VCD or SAIF switching activity is not provided, the tool estimates the switching activity at the primary input ports of the design and the output ports of the black boxes. It uses the default switching activity for the non-annotated pins and where the switching activity cannot be propagated.

For high-fanout nets, such as register resets, the default activity cannot be used. To avoid the default annotation on the register set and reset pins, use the `infer_switching_activity` command to apply activity to these pins so that they are disabled. The command also annotates constant values to design nets, and reports the recommended value for disabling the set and reset pins so that switching activity is propagated through the registers.

When you specify the `infer_switching_activity` command, the tool identifies the preset and clear pins on all the registers using the following pin attributes:

- `is_clear_pin true`
- `is_preset_pin false`

Note:
This mechanism might not identify all the preset and clear pins of the registers.

Use the `is_async_pin` attribute to determine if the preset and clear pins are asynchronous or synchronous. During identification, the tool traces the pins back to the ports or pins through the buffers and inverters to determine the active level of the signals. It verifies if the parity of the set and reset pins is compatible to ensure that the disabling value is the same for all the pins connected to the net.

When you specify the `infer_switching_activity` command without any options, the PrimeTime PX tool generates a report showing the current and inferred switching activity values specific to each pin, as shown in Example 5-1. Use the inferred values of the toggle rate and static probability to provide more realistic switching activity to derive more accurate power results.

Example 5-1 Report Generated by the infer_switching_activity Command With Default Settings

```
pt_shell> infer_switching_activity
------------------------------------------------------------------------
                         Current         Current      Proposed     Proposed
                         Static          Toggle       Static       Toggle
Objects   Type           Probability     Rate         Probability  Rate
------------------------------------------------------------------------
rst       Driver         0.50            0.01         1.0          0.0
------------------------------------------------------------------------
```

When you specify the `-apply` option, the PrimeTime PX tool

- Generates a report showing the current and inferred switching activity values specific to each pin (same as Example 5-1).
- Applies the inferred switching activity values using the `set_switching_activity` commands to the respective pins.

When you specify the `-verbose` option, the report shows the receiver pins and their types.

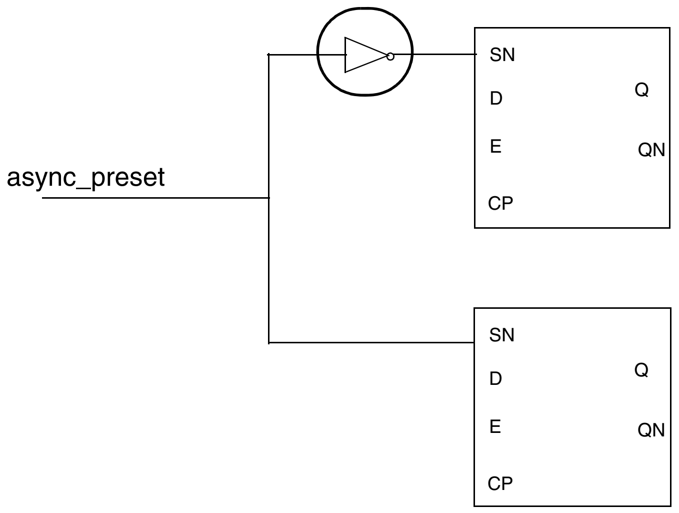
You can save the inferred switching activity values to an ASCII file using the `-output` option of the `infer_switching_activity` command. The ASCII file contains the `set_switching_activity` command issued for each pin, as shown in the following example:

```
pt_shell> set_switching_activity -static_probability 1.0 \
           -toggle_rate 0.0 rst
```

The PrimeTime PX tool does not infer the switching activity in the following cases:

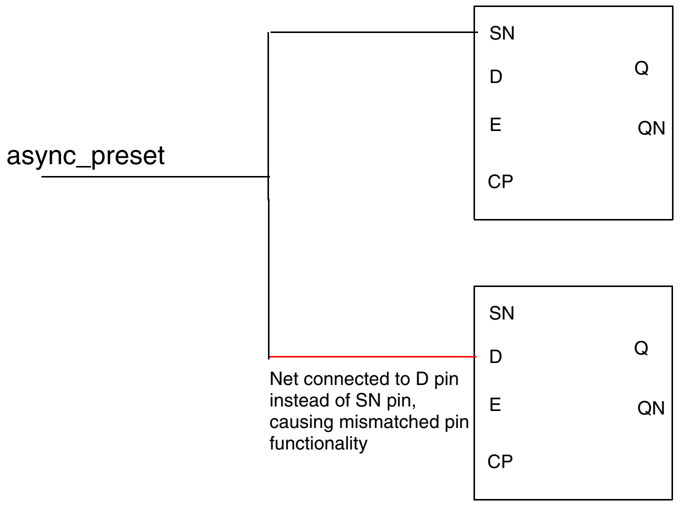
- When it finds a mismatch in the active values of the nets connected to clear and preset pins of multiple registers, as shown in Figure 5-1.

Figure 5-1 The async_preset Pin Has Mismatched Active Values on Two Paths



- When the signals connected to clear and preset pins of the registers are also connected to register pins with different functionality, as shown in Figure 5-2. You cannot use the `infer_switching_activity` command for the scan-enable pins.

Figure 5-2 The async_preset Pin is Connected to Mismatched Pin Functions of Two Registers


async_preset
SN
D
E
CP
Q
QN
SN
D
E
CP
Q
QN
Net connected to D pin instead of SN pin, causing mismatched pin functionality

Estimating Non-annotated Switching Activity

During power analysis, the PrimeTime PX tool requires switching activity information on all design nets and state-dependent and path-dependent information on all design cells and pins. The tool, by default, estimates switching activity that is not user-annotated before calculating power.

The tool estimates the switching activity in the following stages:

- The design nets where the switching activity cannot be derived through propagation are assigned with default activity.
- The switching activity is implied on the non-annotated pins of buffers and inverters or register outputs, from existing annotation. Based on the functionality of the cell, the tool derives the switching activity of non-annotated pins without simulation of the annotated pins.
- To determine the activity on all other nets in the design, the tool propagates the annotated activity with zero-delay simulation.
- The simple switching activity on all the nets (either user-specified annotation, default, implied or propagated) is used to derive the state-dependent and path-dependent switching activity for the power arcs of all the cells.

- When tracking is enabled, the state-dependent and path-dependent activity for the tracked cells is extracted from the VCD file. For the remaining cells, simple switching activity is used to estimate the state-dependent and path-dependent activity.

Annotating Design Nets With Default Switching Activity Values

For the design nets where the tool cannot accurately derive the switching activity or cannot estimate the activity using the propagation mechanism, the default switching activity is used. These include primary inputs, black box outputs and tristates. If the tool does not find any user-defined annotation, it assigns the default switching activity.

The tool uses the following variables to assign the default activity on the primary inputs and black box outputs:

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

The PrimeTime PX tool uses a value of 0.1 for the default toggle rate and 0.5 for the default static probability. The toggle rate is relative to the associated clock.

Implied Activity

After annotating the switching activity, the PrimeTime PX tool derives the annotation through the inverters and buffers, and from the output pin of one register to another. This activity is called *implied activity* because it is derived without performing zero-delay simulation of random vectors.

For example, if the Q pin of a register is annotated with a toggle rate of 0.2 and a static probability of 0.8, the activity of the QB pin is implied with the same toggle rate, and a static probability of 0.2.

For multidriven nets, the PrimeTime PX tool implies annotation based on the activity applied to the driving and loading pins.

Similarly for inverters and buffers, the activity on either input or output pin can be implied from the existing annotation on the other pin.

Precedence Rules for Implied Switching Activity Propagation on a Net

For the following types of design nets, the rules for implied switching activity propagation are:

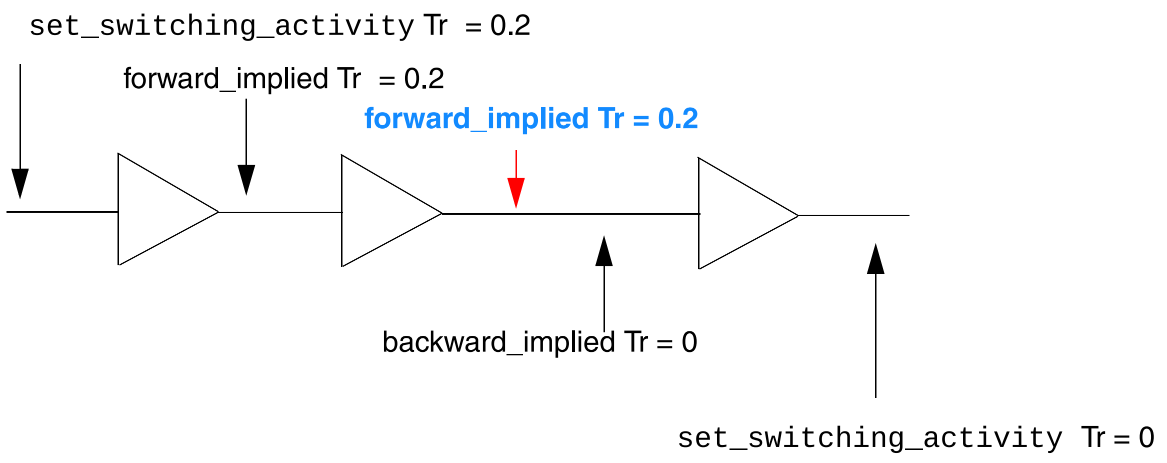
- Nets driving or driven by buffers: If the buffer input or output net is user- or default-annotated, then the non-annotated buffer output or input is default-annotated with the switching activity values on the annotated input or output.
- Nets driving or driven by inverters: If the inverter input or output net is user- or default-annotated, then the non-annotated 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.

For the nets connected to buffers and inverters, the annotation is implied in both forward and backward directions. The PrimeTime PX tool assigns priority to the forward implied activity over the backward implied activity unless the forward implied activity is derived from default annotation.

For inverter chains, the tool propagates the annotated activity to all the pins of the cells on the chain. When you specify multiple `set_switching_activity` commands, the forward and backward implied activity values might differ because the implied activity is derived in both the forward and backward directions. This causes a conflict, as shown in Figure 5-3.

Depending on the order in which you specified the `set_switching_activity` command, the resulting toggle rate (Tr) can be 0.2 or 0.0.

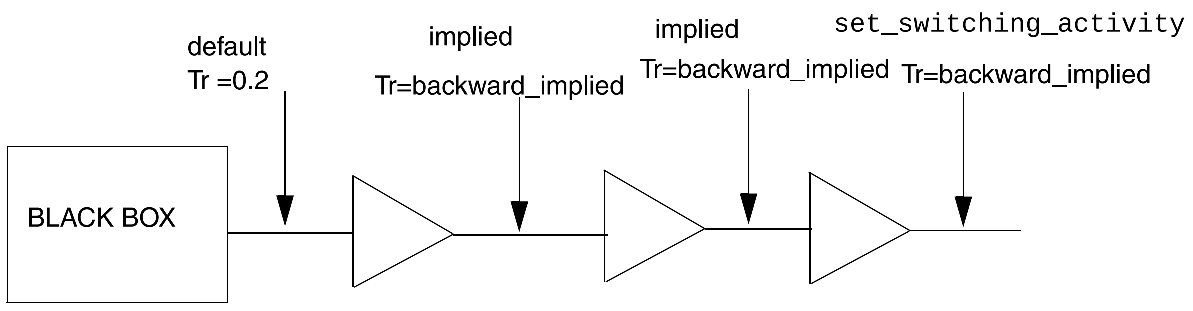
Figure 5-3 Conflict Resolution With the Precedence Rules Applied



To resolve this conflict, the PrimeTime PX tool assigns a higher precedence to the forward propagation over the backward propagation as shown in Figure 5-3. In this case, the PrimeTime PX tool uses the toggle rate of 0.2 because the forward propagation takes precedence over the backward propagation.

If the default activity is propagated forward, the backward implied activity derived from the user-defined annotation takes precedence, as shown in Figure 5-4. In this case, the PrimeTime PX tool chooses the backward_implied toggle rate because the user-defined backward propagation takes precedence over the default forward propagation.

Figure 5-4 Precedence Rules When User-Specified Activity is Applied



The `report_switching_activity` command reports both the forward and backward activities as implied in the report.

Deriving Net Switching Activity With Multiple Drivers

During power analysis, the PrimeTime PX tool derives the power characteristics of a net with multiple drivers using the following steps:

1. The toggle rate of the net is the sum of the toggle rates of the drivers driving the net.
2. The static probability of the net is the average static probability of the drivers driving the net.
3. Toggle rate and static probability value of 0 are assigned to the drivers that do not have annotated switching activity.
4. Drivers without annotated switching activity are excluded from the toggle rate and switching activity calculations.
5. Annotating switching activity on a net is equivalent to annotating switching activity on all the loads of the net. Similarly, annotating switching activity on one of the loads of a net is equivalent to annotating switching activity on the net and on all the other loads of the net.
6. When you set the switching activity multiple times on a net and its loads, the last setting overrides the previous ones.

7. When you annotate switching activity on a driver, the switching activity on the net or the load is ignored and the tool issues a warning.
8. When you do not annotate switching activity on any driver, and the switching activity is annotated on the load (or net), the toggle rate of the load (or net) is distributed evenly to the drivers. Static probability of the load (or net) is copied to all the drivers.

The tool generates a warning message if it encounters a conflict when deriving the switching activity.

If the derived switching activity value does not match the annotated value, the tool marks the source of the switching activity as implied.

For more information about implied switching activity, see "Implied Activity" on page 5-14.

Propagating Switching Activity

The PrimeTime PX tool uses the zero-delay simulation, a propagating mechanism, to derive the switching activity of design nets that are not user-annotated or default annotated. To determine the toggle rates and static probabilities of the non-annotated nets, the tool propagates random vectors by matching the annotated toggle rates and static probabilities with zero-delay simulation. The propagated values do not overwrite the user-annotated switching activity.

Timing Exceptions

When generating and propagating switching activities, the PrimeTime PX tool uses the following exceptions, as shown in Table 5-1.

Table 5-1 Exceptions Considered by the PrimeTime PX Tool to Generate Vectors

Timing command	**Attribute**
`case_analysis`	`Yes`
`multicycle_path`	`No`
`disable_timing`	`No`
`false_path`	`No`

Correlation

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

Correlation affects the accuracy with which the tool propagates the toggle rates. Because accurate analysis depends on accurate toggle rates, correlation also affects the accuracy of power analysis.

The PrimeTime PX tool uses correlation within combinational and sequential logic, resulting in more accurate analysis of switching activity for several designs. The designs with re-convergent fanouts, multipliers, and parity trees exhibit high internal correlation. However, the tool cannot access correlation information that is external to the design. If any correlation exists between the primary inputs of the design, the tool cannot recognize the correlation.

Deriving State-Dependent and Path-Dependent Switching Activity

The PrimeTime PX tool must estimate the state-dependent and path-dependent toggle rates and static probabilities based on the activity on the pins if the state-dependent and path-dependent activity is not annotated. After obtaining the simple switching activity, the tool estimates the state-dependent static probability information and the state-dependent and path-dependent toggle rate information for every cell pin. The tool extracts this information from the switching activity of each cell input and output pin. Use gate-level SAIF files with state-dependent and path-dependent information for the most accurate power results, although the estimation mechanism of the state-dependent and path-dependent activity produces nearly accurate power results.

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

For black-box cells, because the tool is not aware of the cell functionality, the state-dependent and path-dependent estimation is not as accurate as it is for standard cells. Exact state-dependent and path-dependent activity of black boxes, required for accurate power calculation, can be obtained from the VCD file by specifying the `-sdpd_tracking` option. Accurate black power can also be obtained by annotating the state-dependent and path-dependent activity or power values from time-based analysis.

For more information, see “Annotating Internal and Leakage Power on Black Box Cells” on page 5-24.

To annotate accurate state-dependent and path-dependent switching activity, read the SAIF file generated by the `write_saif` command when performing time-based analysis with state-dependent and path-dependent tracking. For more information, see "State-Dependent Path-Dependent Tracking" on page 6-9.

Reporting Switching Activity

To debug the annotation of switching activity and propagation, the PrimeTime PX tool supports several commands. These commands return average activity data that can help you identify the problematic 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.

To report or query the annotated information used for power analysis, use the following commands:

- `report_switching_activity`
- `get_switching_activity`

The commands use the switching activity generated from the `set_switching_activity` command and the SAIF or VCD files that you specify. After you specify the `read_saif`, `read_vcd`, and `set_switching_activity` commands, you can query or report the annotation on various design objects. Use the `get_switching_activity` and `report_switching_activity` commands to get the value of the annotation source indicator.

The `report_switching_activity` command reports information about the various sources of switching activity data. You can report the switching activity after you run the `read_saif` or `read_vcd` command.

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

As shown in Example 5-2, the default report shows the percentage of design objects annotated from the activity file, the `set_switching_activity` command, the `set_case_analysis` command, and the `create_clock` command. In addition, the report also shows the categorization of the nets driven by certain types of drivers.

Example 5-2 Example Report Generated by the report_switching_activity Command

```
pt_shell> report_switching_activity

report_switching_activity
****************************************
Report : Switching Activity
Design : test1
Version: <version>
Date   : <time>
****************************************

Object         File(%)     SSA    SCA Clock Default Propagated Implied Annotated Total
Type
-------------------------------------------------------------------------------------
Nets       3489(100.0%) 0(0.0%) 0(0.0%) 0(0.0%) 0(0.0%) 0(0.0%) 0(0.0%) 0(0.0%) 3489
-------------------------------------------------------------------------------------
Nets Driven by
-------------------------------------------------------------------------------------
Primary        150(100.0%) 0(0%)  0(0%)  0(0%)  0(0%)  0(0%)   0(0%)     0(0%)    150
Input
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%)  0(0%)  0(0%)  0(0%)  0(0%)   0(0%)     0(0%)   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
-------------------------------------------------------------------------------------
```

Sometimes the VCD or SAIF file is not annotated completely due to incorrect name mapping. If this is the case, use the `report_switching_activity` command to identify the non-annotated objects in the design. To debug issues with respect to name mapping, see Chapter 4, “Name Mapping.”

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

```
glitch_rate
toggle_count
toggle_rate
static_probability
```

For example, if you suspect that the tool has incorrectly propagated low-activity rates through a series of sequential elements, resulting in zero activity at the output of some sequential elements, use the following commands to verify this issue:

```
#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_low_activity \
      -include_only sequential
```

To determine average switching activity at the output of sequential elements with nonzero switching activities, use the `update_power` command. As shown in Example 5-3, 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 specify by using the `-base_clock` option.

Example 5-3 The Average Activity Report Showing Toggle Rates and Toggle Counts

```
pt_shell> report_switching_activity -average_activity -hierarchy

****************************************
Report : Switching Activity
-average_activity -hierarchy
Design : counter
...
****************************************
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
--------------------------------------------------------------------------
```

To analyze and debug the annotation, use the `get_switching_activity -toggle_rate` command. This command returns the toggle rate for the specified design object as shown in the following example:

```
pt_shell> get_switching_activity \
            -toggle_rate smas_read_data[29] \
           {“smas_read_data[29]” 0.000017}
```

Without specifying the options, the `get_switching_activity` command displays the switching activity attribute values and their sources in the following order:

- Toggle rate and activity source
- Glitch rate and activity source
- Static probability and activity source

In the following example, the `get_switching_activity` command reports the toggle rate as 0.00017, and the toggle rate annotation source as file, a 0 glitch toggle rate, annotated from file, and the static probability as 0.4998 with the annotation source as file.

```
pt_shell> get_switching_activity \
         -toggle_rate smas_read_data[29] \
           {“smas_read_data[29]” 0.000017 file 0 file 0.4998 file}
```

Use the `activity_source` attribute to identify the source of switching activity information for the net. The attribute value can be one of the following: `file`, `set_switching_activity`, `set_case_analysis`, `propagated`, `implied`, `default`, or `UNINITIALIZED`.

Table 5-2 shows the `activity_source` attribute value and switching activity source information in a descending order of priority:

Table 5-2 Switching Activity Annotation Sources Using the activity_source Attribute

Attribute value	**Annotation source**

`file`
`set_switching_activity`
(Equivalent priority)
	- Toggle rate derived from a SAIF or VCD file (`read_vcd`, `read_saif` commands)
	- Toggle rate derived using the `set_switching_activity` command

`set_case_analysis`	Toggle rate derived using the `set_case_analysis` command.
`create_clock`	Toggle rate derived using the `create_clock` command.
`implied`	Implied activity without random vector propagation from the annotated design points, Q/QB, buffers/inverters, multidriven resolution, and netlist constraints.
`default`	Toggle rate value is the tool default.
`propagated`	Toggle rate is propagated using random vectors from the annotated design points (zero-delay simulation).
`UNINITIALIZED`	Not user-annotated, implied, or default. The uninitialized nets or pins annotation is derived from cycle simulation, and the `activity_source` attribute value is updated to `propagated` after power analysis.

Removing Switching Activity Annotation

Use the `reset_switching_activity` command to remove switching activity annotation from individual design objects, as shown in the following example:

```
pt_shell> reset_switching_activity objects
```

This command removes the simple, state-dependent, and path-dependent switching activity annotation from the specified objects.

When you specify the `reset_switching_activity` command without any option, it removes all previously annotated and propagated switching activity for the complete design hierarchy. By default, this command reports all warning messages.

Use the `reset_switching_activity` command to remove the switching activity information that you already annotated, before specifying new SAIF files.

Performing Averaged Power Analysis

When switching activity annotation is complete, set the power analysis mode to `averaged` with the `power_analysis_mode` variable and then run the `update_power` command to apply the annotation on the required design objects. During `update_power`, the PrimeTime PX tool propagates the annotated and implied activity, and estimates the state- and path-dependent toggle rates to calculate the power.

For black box cells, which have unknown functionality, the estimation of the state-dependent and path-dependent activity is less accurate. Therefore, the power consumption values reported are also less accurate than that of the standard cells. For more information, see “Annotating Internal and Leakage Power on Black Box Cells” on page 5-24.

The PrimeTime PX tool supports power analysis at multiple clock frequencies by scaling the activity relative to the SDC clock. This is useful when the logic simulation activity files (VCD or SAIF) are generated at clock frequencies, other than the SDC clock. Use this feature to perform power analysis at varying clock frequencies without having to regenerate the simulation activity file. For more information, see “Power Analysis With Different Clock Frequencies” on page 5-25.

Run the `report_power` command to report the power values. Or you run the `report_power_calculation` command to display the calculation of the internal power for a pin, the leakage power for a cell, or the switching power for a net. For a detailed description about how to generate power reports, see Chapter 9, “Generating Power Reports.”

Annotating Internal and Leakage Power on Black Box Cells

Use the `set_annotated_power` command to annotate unresolved black boxes with internal and leakage power values (in watts). Although the command is used for black box cells for which the internal and leakage power cannot be estimated, you can also specify leaf cells. The tool overrides the internally estimated internal and leakage values with the annotated internal and leakage power values.

Use the `remove_annotated_power` command to remove the annotated values. If available, the tool re-associates the internally estimated values with the specified cells.

In the cell power report, cells with annotated powers are displayed together with cells where the powers are internally calculated, but they are labeled to show that their powers are annotated. The reports shows the estimated power is overridden by the annotated power.

Use the `report_annotated_power` command to view a summary report of annotated power values as shown in Example 5-4:

Example 5-4 Summary Report Generated by the report_annotated_power Command

```
pt_shell> report_annotated_power -list_annotated
***********************************************
Report: annotated power -list_annotated
Design: mac
...
***********************************************
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  |
```

Power Analysis With Different Clock Frequencies

To perform power analysis at multiple clock frequencies by scaling the activity relative to the SDC clock, first set the `power_enable_clock_scaling` variable to `true`. The default is `false`.

Then 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 activity specified using the `set_switching_activity` and `set_case_analysis` commands.

Example 5-5 shows an example script using this feature.

Example 5-5 An Example Script Using the Clock Definitions From SDC File

```
set_app_var power_enable_analysis true
set_app_var power_analysis_mode averaged
set_app_var search_path         "../src/hdl/gate ../src/lib/snps . "
set_app_var link_library        " * core_typ.db"
read_verilog            mac.vg
current_design          mac
link
read_sdc ../src/hdl/gate/abc.sdc
read_parasitics ../src/annotate/abc.spef.gz
read_saif "../sim/mac.saif" -strip_path "tb/abcinst"

set_power_clock_scaling –period 24 [get_clock clk]
set_app_var power_enable_clock_scaling true
update_power
report_power
quit
```

6

Time-Based Power Analysis

This chapter describes how to annotate the switching activity files, and how the PrimeTime PX tool uses the activity to calculate power in the time-based power analysis mode. In this mode, the tool calculates the power per event to generate power waveforms over time, as well as the averaged power. Based on the type of event-based switching activity file provided, the tool performs instantaneous peak power analysis or cycle-accurate peak power analysis.

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

This chapter contains the following sections:

Overview of Time-Based Power Analysis

In the time-based power analysis mode, the PrimeTime PX tool uses the events in the event-based switching activity file to calculate the power per event. It supports both RTL and gate-level activity files. When RTL-based switching activity files are provided, the tool propagates the RTL events per clock cycle to determine the activity on the non-annotated nets.

For both types of event files, the tool can accurately identify the related pin for each event to calculate power per event.

Specifying the Activity File

In the time-based power analysis mode, you must specify an activity file in the VCD or FSDB format with the `read_vcd` or `read_fsdb` command, respectively.

The PrimeTime PX tool reads the activity files generated from the gate-level or RTL simulation. When you run the `read_vcd` or `read_fsdb` command, the tool reads the header of the event file, and matches the design nets to the nets in the event file. Internal clock pins of the ETM models must be annotated with activity from the event file. This is because in the time-based mode, the tool does not derive the switching activity for these pins from the model definition.

Use the `-rtl` or `-zero_delay` option if the event file is generated from the RTL simulation or gate-level zero-delay simulation, respectively. When both options are specified, PrimeTime PX uses the name mapping set by the `set_rtl_to_gate_name` command to process the event file; the `update_power` command in the subsequently stage uses zero delay simulation to create events on unannotated nets.

Table 6-1 shows how the tool handles different types of event files in the time-based power analysis mode with or without specifying the `-rtl` or `-zero_delay` option.

Table 6-1 Types of VCD and FSDB Files Supported in the Time-Based Power Analysis Mode

The -rtl option	The -zero_delay option	Name mapping	Event propagation	Enforce cycle accurate peak power
Disabled	Disabled	Off	Off	No
Disabled	Enabled	Off	Off	Yes
Enabled	Enabled	On	On	Yes

After you run the `read_vcd` or `read_fsdb` command, the tool generates a summary report showing the percentage of nets with switching activity annotation from the event file, including nets with multiple drivers, specifically the tristate bus. You can use the `report_switching_activity` command to report the annotation of the switching activity.

However, when you specify the `update_power` command, the tool determines the related pins based on the conditions and events specified in the event file.

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

Mapping the Testbench Instance to the Design Module

The PrimeTime PX tool creates the activity file 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 that you specify as the current design, use the `read_vcd -strip_path` command. For example, if the testbench TB instantiates the design module TOP as U, specify the following command:

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

If the VCD hierarchy is not known, use the `report_vcd_hierarchy` command to identify the appropriate path specified with the `-strip_path` option.

Note:
Even if you do not specify the top-level design in the $dumpvars task, the VCD file still lists the complete hierarchy starting with the testbench.

Performing Conditional Power Analysis

To select specific time segments, use the `-time` and `-when` options with the `read_vcd` or `read_fsdb` command.

Use the `-time` option to analyze the selected time segments of a VCD or an FSDB file. With this, you can analyze power when the design is operating in a particular state or mode. Alternatively, you can identify the time segments of the simulation in which the testbench forced the design to be in a particular mode. Specify these time segments using the `-time` option. The unit for this option is nanoseconds.

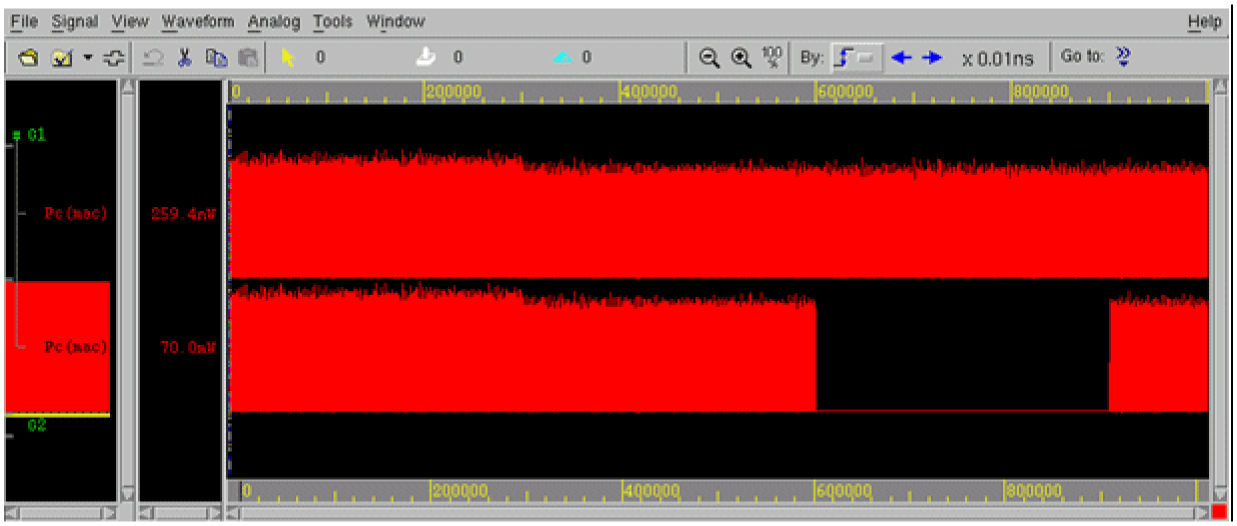
Use the `read_vcd -when` command to simplify the process of identifying the time segments. The `-when` option takes a Boolean expression as an argument, and it accepts design nets or pin names as variables in the expression. When the Boolean expression evaluates to `true`, the tool selects the time segments from the event file. The tool excludes those time segments where the logical values on nets and pins are set to `false` when performing analysis.

The following example shows the portion of the event file in which the N5 signal is `true`.

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

In Figure 6-1, the power waveform at the top is generated without the `-when` option. The power waveform at the bottom is generated with the `read_vcd -when {N5}` command. The `N5` signal is `true` from the beginning of the simulation until 6µs, `false` between 6µs to 9µs, and `true` from 9µs to the end of the simulation.

Figure 6-1 Waveforms With and Without Using the -when Option of the read_vcd Command



Using RTL-VCD Files

Use the `read_vcd -rtl` command to 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 6-9.

Handling VCD Files With Nonmodule Scopes

Use the `read_vcd -format systemverilog` command to specify the RTL-VCD files that are generated from SystemVerilog or VHDL simulation. The procedures contain scopes such as `begin` or `fork`. The VCD file parser ignores these scopes unless you specify the `-format` option.

Using Gate-Level Activity Files

The PrimeTime PX tool reads both zero-delay and SDF-delay gate-level simulation activity files in either VCD or FSDB format with the `read_vcd` and `read_fsdb` command. Use the `-zero_delay` option if the activity file is generated from zero-delay simulation. When you specify this option, the tool assumes the design is fully annotated, and does not perform name mapping. The tool generates power waveforms per cycle. To calculate the instantaneous peak power, the tool requires the SDF delay information.

When you do not specify the `-rtl` or `-zero-delay` option with the `read_vcd` or `read_fsdb` command, the tool assumes the activity file is generated from a gate-level SDF simulation. The PrimeTime PX tool generates power waveforms over time, at the time resolution of the activity file resolution, and identifies the instantaneous peak power.

When you specify the gate-level activity file using the `read_vcd` or `read_fsdb` command, the tool generates a warning message if less than 95 percent of the nets are non-annotated and the switching activity is not propagated through these nets. The non-annotated nets are assumed to have zero activity and remain at a constant 0 logic value.

Handling Large VCD Files

The gate-level VCD file size can be very large. Although the PrimeTime PX tool processes VCD files of size larger than two gigabytes on all platforms, avoid generating and storing large VCD files.

You can generate compressed formats, such as FSDB or gzipped VCD, to reduce the activity file size. Internally, the tool converts the data from the compressed files to the VCD format and inputs to the analysis engine. The converted data is not stored in the tool.

Alternatively, the activity data can also be sourced directly from the HDL simulation to the PrimeTime PX tool. 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 source the activity data directly, follow these steps:

1. Set up the HDL simulation to generate an activity file. For example, when 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 the $dumpfile task and the `read_vcd` command.

   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 run a script that executes a VCS simulation and contains the following command:

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

Use one of the following commands to invoke simulation within the PrimeTime PX tool to source the VCD data directly:

```
-pipe_exec run_vcs
```

or

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

Performing Vector Analysis

During gate-level analysis, use the vector analysis feature to identify the simulation windows with the highest activity, which are likely to produce the peak power. Vector analysis provides information about the vectors by plotting the average toggle rate per interval, over the specified time period. The PrimeTime PX tool reads the activity file and writes an activity waveform for the top-level module of the activity file. The tool partitions the total simulation time into intervals. For each interval, the average toggle rate is calculated using the following formula:

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

View the activity waveforms to determine if the testbench is simulated as expected and if the vectors have sufficiently covered the design to use them as inputs for power analysis. The coverage per interval is calculated using the following formula:

$$\text{Coverage} = \frac{\text{Number of Signals With at Least One Toggle}}{\text{Number of Signals}}$$

Use the `write_activity_waveforms` command to generate the activity waveforms from the activity file. Use this command before reading the design data into the tool. The output generated from vector analysis has the same names as the instances in the original activity file. The output for the waveforms is in the .fsdb format. 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 6-2 on page 6-8.

Use the `-peak_window` option to avoid missing the peak activity interval, when the windows of high activity fall into separate intervals. Using this option, you can specify the sliding window period to monitor the average activity per interval. The tool searches for the peak period in the activity file and reports the peak activity using a sliding window. Specifying the option affects the way the peak periods are reported in text format. It does not affect the generation of power waveforms.

Note:
Specify a value to the `-peak_window` option that is a multiple of the value that you specify to the `-interval` option.

Use the `-peak_type` option to specify either `minimum` or `maximum` values. The PrimeTime PX tool searches for an interval of maximum or minimum activity based on the argument specified. Otherwise, if you specify 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.

The `write_activity_waveforms` command also supports options to specify the hierarchy level for cells to include in the waveform, to specify signals to be excluded from the waveforms, and so on.

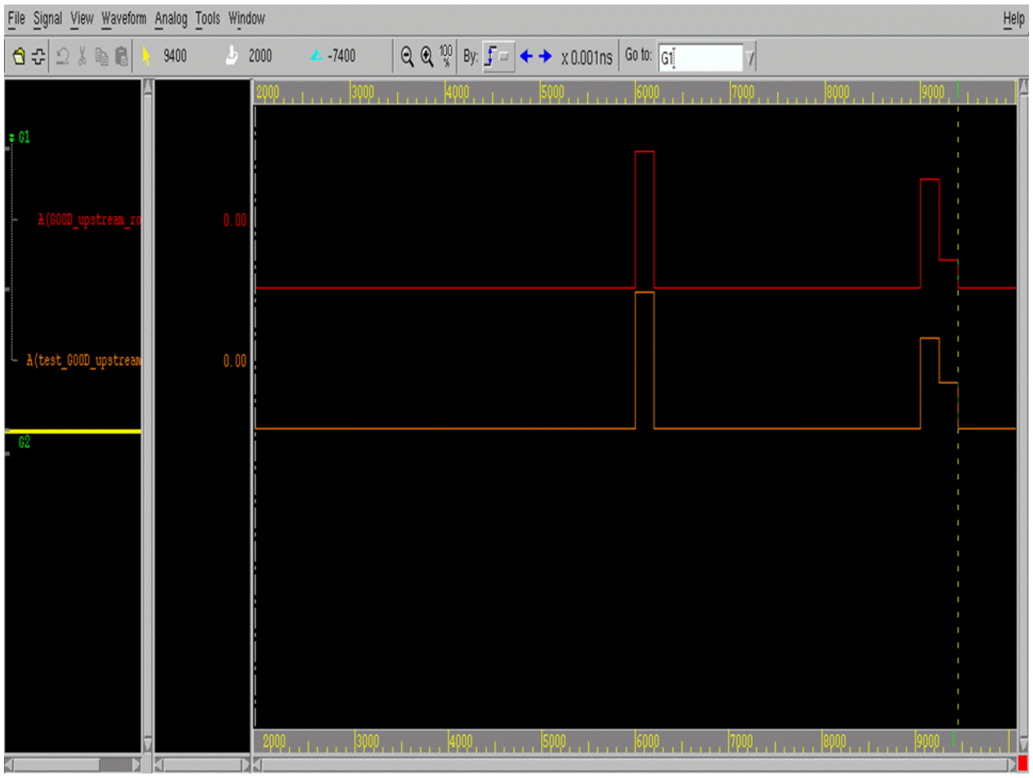
Example 6-1 shows how the `-peak_window` option affects the report generated by the `report_activity_waveforms` command. By specifying the `-peak_window` option as 5, the reported peak interval is 5 ns, although the interval for the waveform is 0.2 ns.

Example 6-1 Activity Waveform Report

```
pt_shell> write_activity_waveforms -vcd myvcd.vcd -output myvcd.out \
          -interval 0.2 -peak_window 5 -hierarchical_levels 2
pt_shell> 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
----------------------------------------------------------------------
```

Figure 6-2 shows the waveform for the report.

Figure 6-2 Time-Based Activity Waveform



To analyze power only for the high activity windows, specify the desired analysis windows using the `read_vcd -time` command.

Setting the Options for Time-Based Power Analysis

The `set_power_analysis_options` command controls the cells that are analyzed, the waveform data output, and the tracking of state-dependent path-dependent toggle rates.

Waveform Data

By default, in the time-based mode, the PrimeTime PX tool generates power waveform data for all the design hierarchies when you run the `update_power` command. To generate the waveform data for the leaf cells or top-level design, specify the `set_power_analysis_options -include` command before the `update_power` command. To prevent the generation of power waveforms, set the `waveform_format` option to `none`.

State-Dependent Path-Dependent Tracking

By default, the PrimeTime PX tool does not store the state-dependent path-dependent toggle rates in the time-based analysis mode. When the tool processes the VCD file, it computes power for each event in the file.

To store the state-dependent path-dependent toggle rates so that they can be output to a SAIF file, specify the `write_saif -sdpd_tracking` command. This causes the PrimeTime tool to store the toggle rates for black box cells, as the tool uses the accurate toggle rate information for accurate power results. Use the `-sdpd_tracking_cells` option to specify a list of cells for which you enabled the `-sdpd_tracking` option. When power analysis is complete, write out the time-based toggle rate information to a SAIF file, and use this file in the averaged power analysis mode.

Performing Time-Based Power Analysis

To analyze power in the time-based mode, run the `update_power` command. The tool processes the events in the VCD file and writes out power waveforms.

Before processing the events in the activity file, the tool calculates the input transition time and output load for every instance in the design. These values are stored to access the appropriate energy values in the library power tables for the events in the activity file.

For transitions on the output pins of the cells, the PrimeTime PX tool ascertains the related pin by searching for the input transitions that caused the output transition. After identifying the pin, the tool accesses the appropriate power arc and analyzes the energy for that transition based on the power table indexes.

Use the `report_power` command to generate a report showing the largest power value (peak power) and the time at which it occurred. For a detailed description about how to generate power reports, see Chapter 9, “Generating Power Reports.”

To prevent the generation of power waveforms, set the `waveform_format` option to `none`.

Cycle-Accurate Peak-Power Analysis

In the cycle-accurate peak-power analysis, specify the RTL activity file to compute the power per event accurately, and view power waveforms at the clock-cycle resolution.

Using the switching activity from the RTL activity file, the PrimeTime PX tool performs name mapping to match the RTL objects to the gate-level objects. If a name mapping file is available from the Design Compiler tool, you must source it before using the `read_vcd` or `read_fsdb` command. If not available, the tool depends on a built-in algorithm to perform the matching. For more details, see Chapter 4, “Name Mapping.”

When you run the `update_power` command, the PrimeTime PX tool processes the events in the activity file, and propagates the activity to the non-annotated portions of the design. The tool sums up the energy per clock-cycle for all the events, and averages the power over the cycle.

Sometimes, event propagation through edge-triggered sequential elements might lead to excess activity and therefore overestimation of power in the cycle-accurate time-based analysis mode. In this case, disable event propagation by setting the `power_capp_edge_triggered_propagation` variable to `false` before the `update_power` command. The default is `true`.

Example 6-2 is a script used to perform cycle-accurate peak power analysis using the RTL-VCD file.

Example 6-2 Cycle-Accurate Peak Power Analysis Using RTL VCD File

```
set_app_var power_enable_analysis true
set_app_var 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
set_power_analysis_options
update_power
report_power
```

Specifying Activity With the set_case_analysis Command

If the activity file does not contain switching activity for primary inputs, use the `set_case_analysis` command to force the port to a constant value. This is useful for applying activity to test enable signals, which do not exist in the RTL-based activity file, but exist in the gate-level netlist.

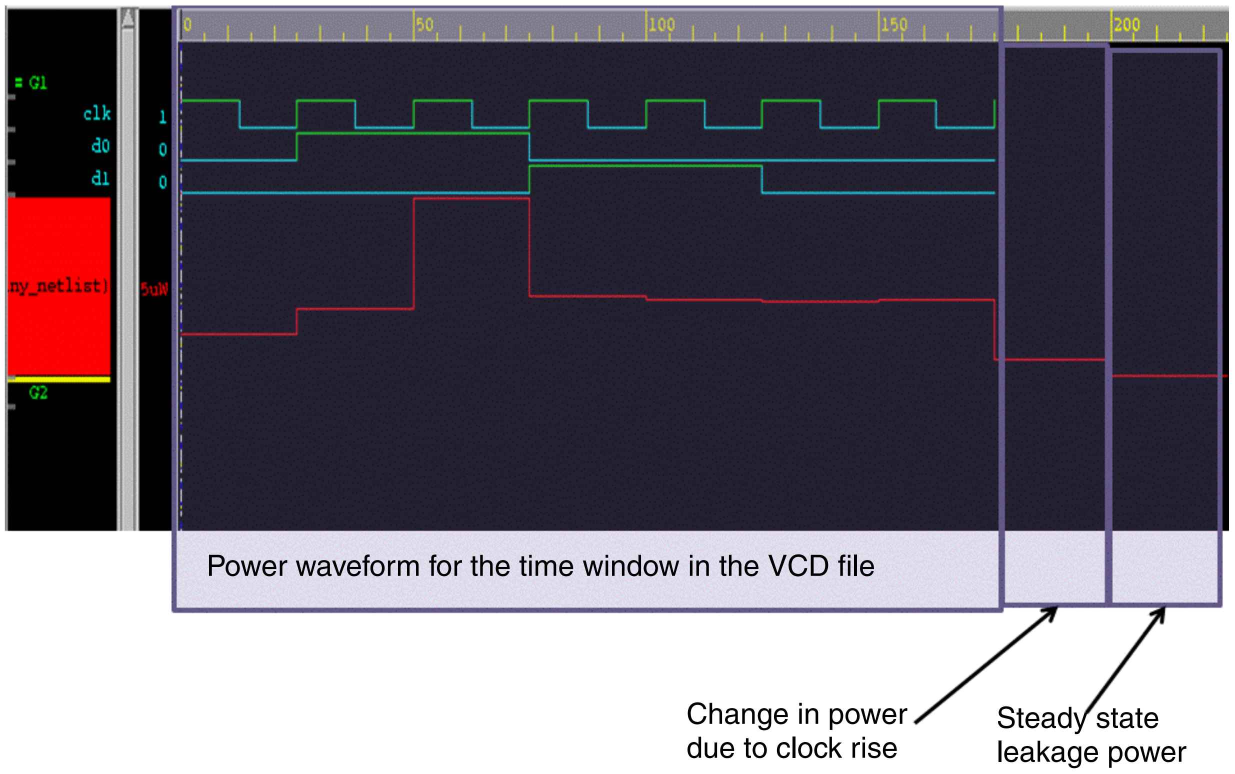
Because the annotation in the activity file takes precedence over the `set_case_analysis` command, it can only be used to annotate signals, which are not present in the activity file.

Understanding Cycle-Accurate Peak-Power Waveforms

During peak power analysis, the PrimeTime PX tool generates a time-based power waveform and reports the peak power over time. The tool reports power variation at every clock cycle and is not limited by the time window in the activity file. Sometimes the power waveform contains more cycles than those in the activity file. This is because the tool monitors the cycles until it detects no further power variations as shown in Figure 6-3.

Use the `report_power` command to report the power values that match the values shown in the power waveform.

Figure 6-3 Understanding Cycle-Accurate Peak Power Waveforms


G1
clk
d0
d1
my_netlist)
5uW
G2
0
50
100
150
200
Power waveform for the time window in the VCD file
Change in power due to clock rise
Steady state leakage power

Estimating Glitch Power

In the cycle-accurate peak-power mode, the tool considers glitches as extra transitions within a clock cycle before a signal settles to its steady state. When multiple inputs transition simultaneously, the tool treats them as separate events, resulting in multiple transitions in the output. The power consumption of these extra transitions is scaled by 0.5.

In the following equation, the power consumption of these extra transitions is scaled by 0.5:

```
glitch power = 0.5 * dynamic power for extra transitions
```

Instantaneous or Event-Based Peak Power Analysis

When you specify gate-level SDF simulation activity, the PrimeTime PX tool calculates the power for every event. To calculate peak power values, the PrimeTime PX tool

1. Analyzes every event in the event-based activity file.
2. Checks for associated power arcs and related pins.
3. Accesses the appropriate energy table in the logic 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.

From the power waveforms, you can identify the instantaneous peak power.

By default, the tool generates power waveforms at the event-based activity file resolution. To change the resolution, use the `set_power_analysis_options -waveform_interval` command to set the value to the desired interval.

Figure 6-4 shows the peak power analysis result using a gate-level VCD file with the default time interval. The tool calculates energy for every event and distributes it over the path delay.

Figure 6-4 Peak Power Analysis for Gate-Level VCD With Default Time Interval

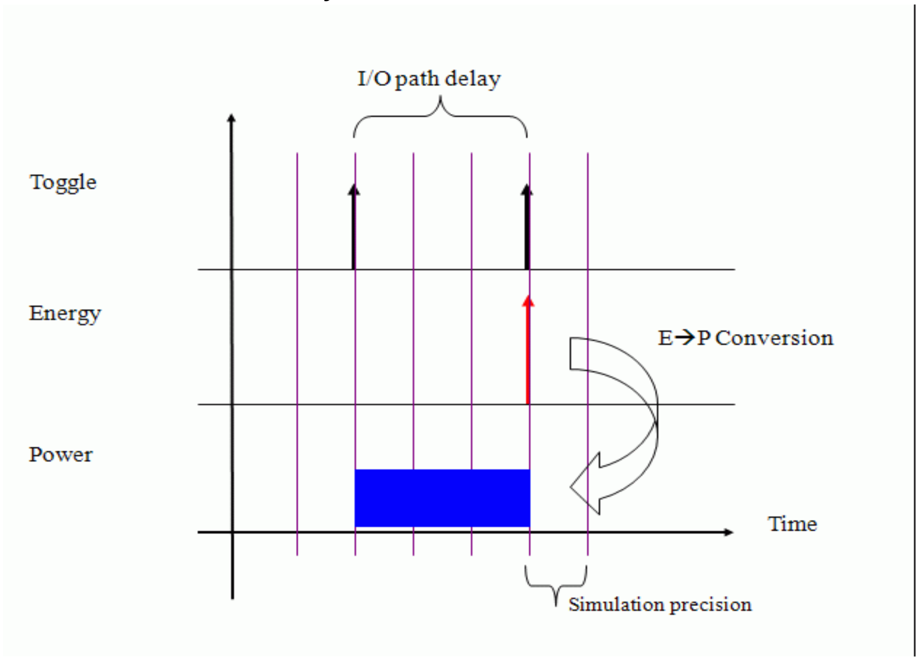
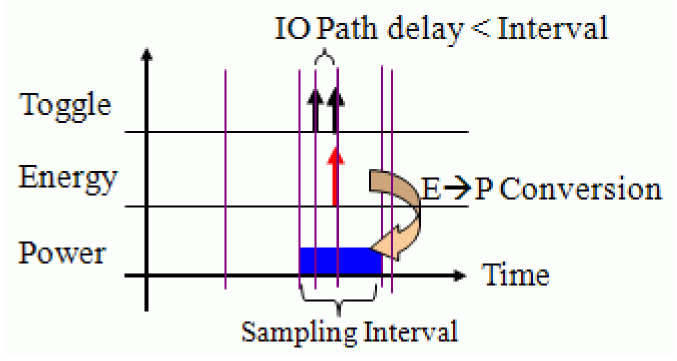


Figure 6-5 shows the peak power analysis result using gate-level VCD file with a user-specified time interval. The PrimeTime PX tool calculates the energy for every event and distributes it over the specified time interval if the time interval is greater than the I/O path delay. If the time interval is less than I/O path delay, the tool distributes the energy over the I/O path delay.

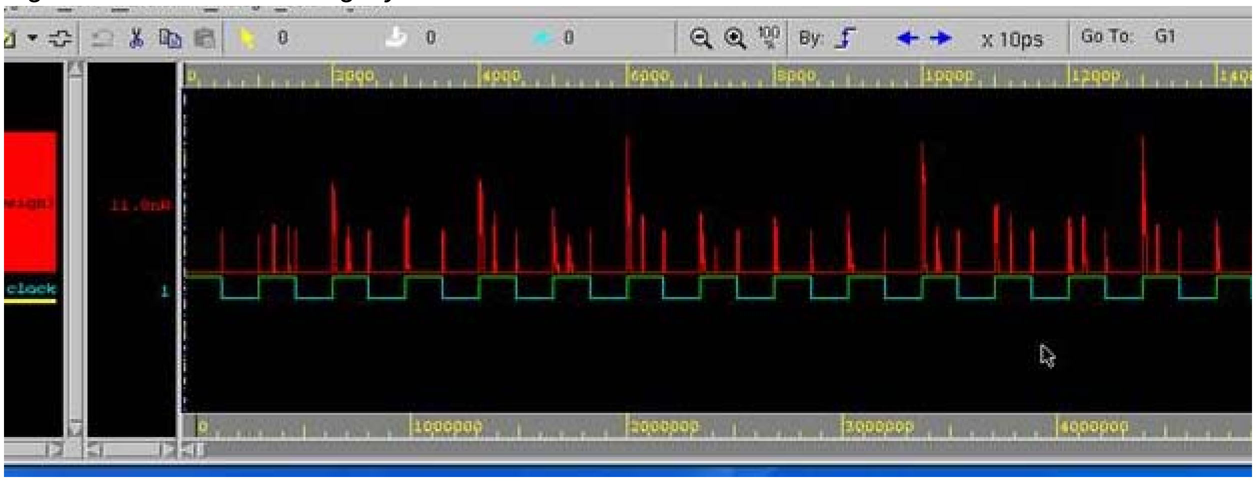
Figure 6-5 *Peak Power Analysis for Gate-Level VCD With User Specified Time Interval*



Understanding Instantaneous or Event-Based Peak Power Calculation

When calculating the power per event, the PrimeTime PX tool considers the value setting of several variables described in the following sections. Figure 6-6 shows an example of the cycle-accurate peak power waveform.

Figure 6-6 *Understanding Cycle-Accurate Peak Power Waveform*



Variables Affecting Instantaneous Peak Power Analysis

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

The PrimeTime PX tool assumes that the library power tables are 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) incurs 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 you set the variable to `false`, the tool assumes that the library internal energy tables contain only the internal energy. During characterization, the tool derives the rising and falling energy 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
```

Specifying the `power_table_include_switching_power` variable does not affect the average power because multiple events average out the difference. However, the variable setting affects the power waveforms, particularly for small designs and vector sets. For optimal peak power analysis, set this variable appropriately.

Contact the library vendor to identify the best formula used for characterization. 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.

Example 6-3 shows a time-based report indicating peak power values, including the glitch, xtransition, and peak power values.

Example 6-3 Reporting Peak Power Values: Glitch, Xtransition, Peak Power

```
**********************************************
Report: Time Based Power
Design: mac
...
**********************************************
  Attributes
      i  -  Including register clock pin internal power
      u  -  User-defined power group
Power           Internal   Switching   Leakage    Total
Group           Power      Power       Power      Power   (     %)  Attrs
-------------------------------------------------------------------------
clock_network   7.133e-04 0.0000000  0.0000000   7.133e-04(19.43%)   i
register        4.703e-04 1.095e-04  6.660e-08   5.799e-04(15.80%)
combinational   8.964e-04 1.208e-03  1.836e-07   2.104e-04(57.33%)
sequential      5.289e-05 2.200e-04  9.163e-09   2.729e-04( 7.43%)
memory          0.0000    0.0000     0.0000      0.0000(0.00%)
io_pad          0.0000    0.0000     0.0000      0.0000(0.00%)
black_box       0.0000    0.0000     0.0000      0.0000(0.00%)
-------------------------------------------------------------------------

Net Switching Power  = 1.537e-03 (41.88%)
Cell Internal Power  = 2.133e-03 (58.11%)
Cell Leakage Power   = 2.594e-07 ( 0.01%)
 -----------------------------------------
Total Power          = 3.670e-03 (100.00%)
-------------------------------------------

X Transition Power   = 2.171e-07
Glitching Power      = 2.942e-05
Peak Power           = 0.0970
Peak Time            = 1956
1
```

Modeling Special Conditions

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

- Glitches
- Z states
- X states

Glitch Power

Glitches are small pulses, which do not achieve full logic levels, because the pulse width is too small. The PrimeTime PX tool determines whether the pulse is small enough to be considered a glitch and then calculates the power using the following formula:

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

where

Ratio = $[(\text{pulse\_width} * 2) / (\text{rise\_ramp} + \text{fall\_ramp})]^2$

pulse_width < (rise_ramp + fall_ramp) / 2

You can obtain the pulse width from the simulation, and use the `report_power_calculation` command to obtain the rise and fall transition times on the desired instance.

By default, the tool uses the linearly scaled full-transition time to scale the glitch power and calculate the scaling ratio used to compute glitch power. To achieve a more accurate glitch power value, use the `power_full_transition_glitch_scaling` variable to specify a scaling factor for scaling the transition time calculated by the tool. Set the scaling factor to a value greater than 1 and less than 10. The default is 1.

Here is an example:

```
pt_shell> set power_full_transition_glitch_scaling 2
```

Note:
The `power_full_transition_glitch_scaling` variable is available only in the time-based mode.

If you set a scale factor with the `power_full_transition_glitch_scaling` variable, querying the `toggle_rate` attribute reports the number of “normal” toggles, not scaled toggles. The scaled glitches are tracked as glitch toggles, and can be queried with the `glitch_rate` attribute.

Use the `report_power` command to generate a report on the estimated glitch power.

Z State

Transitions through the Z state are assumed to consume no power. The PrimeTime PX tool traces the transitions; if a transition after the Z state differs from the original transition, it considers it 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, the tool considers it 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 of the controlling variable, `power_x_transition_derate_factor`, is `0.5`.

To change the scale factor, modify the default setting of the variable. Setting this variable to `1` results in pessimistic power reports.

The `power_match_state_for_logic_x` variable determines how the tool interprets X state in the Boolean function. The default is `0`. Therefore, the tool, by default, considers X state as logic 0.

Initial X-State Handling

When processing VCD data, the PrimeTime PX tool processes every event. Several VCD files contain X states at initialization when the tool is propagating the input values.

The `power_include_initial_x_transitions` variable determines if the tool calculates power during initialization or not.

By default, the variable is set to `false` and the tool ignores the switching and internal power consumed by the transition of nets from an unknown to a known state.

Set the `power_include_initial_x_transitions` to `true` if you are concerned with the power at initialization, so that the power peak generated during the circuit initialization is included in the power analysis.

Unmatched States

When the PrimeTime PX tool processes the events in the switching activity file, events for which there is no matching condition in a power table might occur. The PrimeTime PX tool generates 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, thereby 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, the PrimeTime PX tool provides an estimate of the power consumption based on the average value of all the available tables for that output pin.

To ignore the event, set the `power_estimate_power_for_unmatched_event` variable to `false` before the `update_power` command.

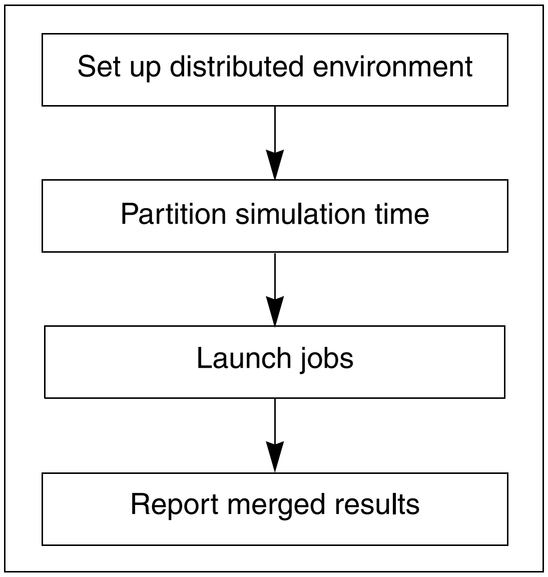
Distributed Peak Power Analysis

To speed up the peak power analysis process, the PrimeTime PX tool performs distributed peak power analysis using the distributed multi-scenario analysis infrastructure available in the PrimeTime tool.

The distributed multi-scenario analysis flow and the usage model is similar in both the PrimeTime and PrimeTime PX tools, except that, in the PrimeTime tool, the analysis is for timing, while in the PrimeTime PX tool, 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 of 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, as shown in Figure 6-7.

Figure 6-7 Distributed Peak Power Analysis Flow



To perform distributed peak power analysis,

1. Set up a distributed computational environment that can run multiple scenarios simultaneously.

This step includes specifying a working directory, the log files, the number of PrimeTime and PrimeTime PX tool licenses to be used. This step also includes setting up and starting a distributed pool of machines. The variables and commands that you use in setting up the distributed environment for peak power analysis are the same as the ones that you use in the PrimeTime tool for timing analysis.

Example 6-4 shows the commands for a distributed environment for peak power analysis using two machines in the LSF, two PrimeTime and two PrimeTime PX licenses.

Example 6-4 Setting Up Distributed Environment for Peak Power Analysis

```
set_app_var 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

set_host_options -32bit -num_processes 2 \
        -submit_command /lsf/bin/bsub

start_hosts
```

To report power when the peak power analysis process is finished, set the `power_enable_analysis` variable to `true`.

2. Partition the simulation into blocks.

   This step involves dividing the total simulation time for the peak power analysis into smaller time windows. The simulation window partitions must be the same time equivalent time as the peak power analysis. The window partitions into which the entire simulation time window must be divided depends on the number of machines used and the number of licenses available.

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

   To perform power analysis over time before processing the VCD file, you must read the design data first, apply design constraints, and then update the timing.

   To perform power analysis over time for a large VCD file, split the VCD file to run multiple time-windows in parallel, and create scenarios that correspond to each time window.

   The following steps are common to all the scenarios before reading the VCD file:

   a. Reading and linking the design.

   b. Reading and applying the constraints and parasitics.

c. Updating the timing constraints.

The steps performed to read different time-windows of the VCD file are specific to each scenario.

Example 6-5 shows the commands for creating two simulation scenarios and performing peak power analysis for the two scenarios:

Example 6-5 Creating Two Simulation Scenarios and Performing Peak Power Analysis of the Scenarios

```
set test_dir .
# The common data is used by all the scenarios. Since all the
# scenarios must read the design data and update timing.
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}

set_app_var power_enable_analysis true
set_app_var power_analysis_mode time_based
set_app_var search_path "."
set_app_var 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_parasitics $test_dir/mac.spef.gz
update_timing

# The simulation time from 0 to 10000 is divided into two blocks, 0 to
# 5000 and 5001 to 10000.
# 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"
```

3. Launch the jobs.

When you setup the distributed environment and created the scenarios, use the `current_session` command to select a set of scenarios for analysis. 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 listed. You can run this command only after you have selected the scenarios using the `current_session` command.

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

```
current_session -all

remote_execute { \
   report_power
}
```

For more information about the actual launching of the jobs and selecting the physical machines to run the jobs, see the *PrimeTime User Guide*.

4. Display the combined power report.

   Run the `report_power` command to generate a combined report which records all the power data from separate slave processes, and identifies the maximum peak power and its corresponding peak time from all the scenarios. This command is supported only in the slave processes.

   Note:
   If the `report_power` command in the slave processes is set with the `-rail` option, the tool uses the output from the last `report_power` command in the combined power report.

   Following is an example of the combined power report from distributed multi-scenario analysis:

```
****************************************
Report : Merged Power Report
Design : top
Version: <version>
Date   : <time>
****************************************
Scenario Name    Switching Power      Internal Power   Leakage Power
  Total Power    X-Transition Power   Glitch Power
  Peak Power     Peak Time            Rail Names
-------------------------------------------------------------------
S1                1.3297e-06           2.9673e-05      1.17158e-08
  3.10144e-05     0                    1.2605e-07
  0.00544426      0.101                VDD1_ADD

S2                0                    0               1.17158e-08
  1.17158e-08     0                    0
  2.34316e-08     1                    VDD1_ADD

S3                0                    0               1.17158e-08
  1.17158e-08      0                   0
  2.34316e-08         2                VDD1_ADD
Peak                                   0.00544426      0.101
```

Viewing and Scaling the Power Waveforms

You can view waveform data by using any waveform tool that supports the FSDB or .out format. The supported waveform viewers are nWave, Custom WaveView, and CosmoScope. For more information about integrating waveform viewers into the tool, see SolvNet article 036949, "Integrating Waveform Viewers into PrimeTime PX."

Setting Up the nWave Waveform Viewer

By default, the PrimeTime PX tool invokes the nWave and fsdb2vcd utility provided with the PrimeTime PX installation. To access the fsdb2vcd utility and the latest version of nWave waveform viewer, set the $NOVAS_HOME environment variable to the installation root directory of the Verdi release version. When this variable is specified, the PrimeTime PX tool invokes the waveform utilities from the $NOVAS_HOME/bin directory instead of the PrimeTime PX installation directory. This ensures version compatibility when converting FSDB to VCD format during switching activity annotation.

Invoking the nWave Waveform Viewer

To invoke the nWave waveform viewer, specify the following command on the UNIX command line:

```
%> nWave
```

Viewing Power Waveforms Using 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.

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

1. Select all the signals in the Signal column.
2. Choose Analog > Zoom Value from the nWave menu.
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

The PrimeTime PX tool analyzes power for designs with different power supply voltages for different cells. Using the multivoltage infrastructure, you can choose between two distinct methods to calculate power for designs with different power supplies. These methods are not interchangeable.

- The IEEE 1801 standard also known as Unified Power Format (UPF)
  - ❍ Use the UPF file 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.

All the methods support the following capabilities:

- Concurrent multirail power analysis.
- Accurate power calculation for cells with multiple power supplies by using internal and leakage power tables with liberty power-and-ground pin syntax.
- 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.

  The PrimeTime PX tool supports the following types of power switches:
  - ❍ Fine-grain and coarse-grain switches
  - ❍ Header and footer switches
- Voltage scaling for power calculation

Voltage and Temperature Scaling Between CCS and NLPM Libraries

Multivoltage designs that contain cells operating at different operating conditions require libraries characterized for multiple voltages. To simplify multivoltage analysis, the PrimeTime PX tool supports voltage and temperature scaling. This feature enables you to analyze the power consumption of the specified multivoltage designs that do not have libraries characterized at the desired operating condition. When libraries for the desired temperature and voltage are available, the tool uses the library data to calculate power.

However, if the libraries are not characterized for the desired operating conditions, the tool uses interpolation to derive appropriate power values. This feature reduces the need to have separate libraries for each incremental voltage and temperature when performing multivoltage analysis. This feature is supported in both the averaged and time-based power analysis modes.

Scaling is supported for multirail cells as well as single-rail cells. If libraries are not available in the scaling group, the tool performs single-rail scaling. The following guidelines apply when defining libraries that are a part of a scaling group:

- The libraries that you specify 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 present for each cell across the libraries within a scaling group. If this check fails, scaling does not occur and the tool generates a warning message.
  - The power tables are uniquely identified. If this check fails, the tool assumes that the power tables are presented in an identical order across all libraries.
- Define the scaling relationships between the libraries using the `define_scaling_lib_group` command.
- The specified libraries must be characterized at different voltage and temperature coefficients.
- List the libraries in scaling groups only one time.
- Define the default libraries using the `link_path` variable. If scaling is not performed, the PrimeTime PX tool uses the libraries specified using the `link_path` variable. If scaling is needed, link the design after you read in the scaling libraries. For example,

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

  Then set the `link_path` variable.

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

- Every library, which is part of a scaling group, is removed if one of the scaling groups is removed using the `remove_lib` command.
- If relinking the design, rerun the `define_scaling_lib_group` command to identify the libraries that are part of a scaling group.

Specify the operating conditions for analysis using the `set_operating_conditions` and `set_voltage` commands. If libraries that are characterized for the specified operating condition are available, the PrimeTime PX tool uses the data directly. If not available, the tool verifies if the operating conditions are within the range defined by scaling group. If the operating conditions are outside the range, the tool generates an error message and accesses the data from the primary linked library. If within the range, the power values are scaled. Temperature scaling of internal power is performed with linear interpolation. Both voltage scaling of leakage and internal power, and temperature scaling of leakage power are performed using the nonlinear interpolation method.

If scaling libraries are not available, use the `link_path_per_instance` variable to specify libraries to be used for different instances of the same cell type.

For more information about the `link_path_per_instance` variable, see the *PrimeTime User Guide*.

Annotating Power Per Rail

When annotating power values on multivoltage macro cells, use the `-rails` option of the `set_annotated_power` command to perform annotation of each rail or UPF supply net. If you do not specify the `-rails` option, the power values specified using `set_annotated_power` command are divided evenly between the power rails of the multivoltage macro cell. During analysis, when you run the `update_power` command, the tool uses the annotated power values for the specified macro cells when calculating the total design power for each rail.

As shown in Example 7-1, cell U1 has rail connections of VDDA, VDDB, and VDDC. The first `set_annotated_power` command annotates the internal power with 0.1 and leakage power with 0.0 for the cell U1 on rail VDDA. The second `set_annotated_power` command annotates the internal power with 0.2 and leakage power with 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 7-1 Setting the Annotated Power on Multiple Rails

```
set_annotated_power -internal_power 0.1 -leakage_power 0.0 -rails VDDA U1
set_annotated_power -internal_power 0.2 -leakage_power 0.1 \
   -rails {VDDB VDDC} U1
```

Example 7-2 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. The `report_power` command reports the annotated power of 0.1 for the cell.

Example 7-2 Setting and Reporting the Annotated Power on Power Rail

```
set_annotated_power -internal_power 0.3 -leakage_power 0.0 U1
report_power -rails VDDA -cell_power U1
```

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

Example 7-3 shows the report generated by the `report_annotated_power` command using the `-rails` and `-list_annotated` options.

Example 7-3 Report Generated by the report_annotated_power -rails Command

```
****************************************
Report : annotated_power -rails -list_annotated
Design : test
...
****************************************

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        |
```

Setting the Concurrent Multirail Power Analysis Mode

The PrimeTime PX tool supports concurrent power analysis feature on multiple power rails. During analysis, the tool supports viewing the contribution of each rail in a single run, so you can identify the rails that contribute to excessive power consumption. This helps measure the power savings and contributes to power reduction, and therefore improves the turnaround time significantly.

To enable the feature, set the `power_enable_multi_rail_analysis` variable to `true`. By default, all power rails are analyzed. When set to `false`, specify the desired rails or supply nets for analysis using the `report_power -rails` command before you run the `update_power` command. To view the power per rail, run the `update_power` command separately for each rail or supply net.

When you enable the concurrent multirail analysis mode, you can report power per rail using the `report_power -rails` command. If this mode is not enabled, set the current power rail before performing multirail analysis.

Example 7-4 shows a script example of the `current_power_rail` command to view the power consumption per rail for a design that contains two power rails, VDD1 and VDD2:

Example 7-4 Script Example Using the current_power_rail Command

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

Concurrent multirail power analysis is supported in both averaged and time-based power analysis modes and is supported in all analysis methods of multivoltage designs.

Reporting Power Per Rail in Power Domain Mode

The power reporting behavior in the power domain mode is compatible with the other power analysis modes. By default, the power for all supply nets is reported. You can use the `report_power -rails` command to report the power per rail when the `power_enable_multi_rail_analysis` variable is set to `true`.

Generating Waveforms Per Rail

Use the `–multi_rail_waveform` *`supply_net_list`* option of the `set_power_analysis_options` command to generate waveforms for the specified rails in time-based analysis mode.

Reporting Power Attributes for Each Rail

When you set the `power_enable_multi_rail_analysis` variable to `true`, the PrimeTime PX tool performs power analysis for all the supply nets or power rails in the design for all multirail analysis modes. By default, the tool reports the total power of all the supply nets or rails, as shown in Example 7-5.

Example 7-5 Report Generated With the -rails Option

```
****************************************
Report : Averaged Power
...
****************************************
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%)
```

However, you can selectively view the power for each supply net or rail by specifying the `report_power`, `report_attribute`, and `get_attribute` commands for the specified rail, without recalculating the power values, as shown in Example 7-6.

The power attributes available for each power rail are: `internal_power`, `switching_power`, `leakage_power`, `dynamic_power`, and `total_power`.

Example 7-6 Reporting Power Attributes for Each Rail

```
pt_shell> set_app_var power_enable_multi_rail_analysis true
pt_shell> update_power
pt_shell> set_current_power_net vdd1
pt_shell> get_attribute [get_cell U1] internal_power
 1.1e-07
pt_shell> set_current_power_net vdd2
pt_shell> get_attribute [get_cell U1] internal_power
 2.2e-07
```

Note:
The attributes correspond to the value for the specified rail.

The `peak_power` and `peak_time` attributes report the total power for all the rails unless you set the `power_enable_multi_rail_analysis` variable to `false`.

When you run the `report_attribute`, `get_attribute`, or `report_power` command, the report displays the power for the specified rail. The PrimeTime PX tool performs power analysis for all the rails when you run the `update_power` command but only reports the values for the specified rail. The attributes for each rail of each cell can be used in conjunction with the PrimeTime `pg_pin_info` cell attributes, which return a Tcl collection of the PG pins and their associated rails to generate custom reports. These reports include the power, supply net, and PG-pin names for all the cells in the design.

When the `power_enable_multi_rail_analysis` variable is set to `false`, use the `set_current_power_net` or `current_power_rail` command to specify the power rails in the power report. The tool analyzes and reports only the power consumption of the current power net.

Multirail power analysis can be performed for all flows and all four types of reporting: summary, cell-based, net-based, and hierarchy-based reporting.

You can extract the total power attributes for each rail, generate custom per-rail reports for hierarchical blocks, and include the power consumption of rails.

You can indicate the name of the rail to be reported by specifying `leaf`, `top`, or `through_switch` with the `-return_supply_level` option.

Example 7-7 shows how to use the `get_power_per_rail` command.

Example 7-7 Using the get_power_per_rail Command

```
pt_shell> get_power_per_rail -rails vddcx vddmx -return_supply_level leaf
{vddcx inst_blck1 0.0} {vddcx inst_blck2 0.0} {vddcx not_included 0.500}
{vddmx inst_blck1 0.0} {vddmx inst_blck2 0.0} {vddmx not_included 0.6}
{unconnected inst_blck1 0.0} {unconnected inst_blck2 0.0}
{unconnected not_included 0.0} {other u_inslt_blck1 0.3 }
{other inst_blck2 0.6} {other not_included 0.0}
```

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

Extracting Power Attributes for Each PG Pin

You can extract the power for each PG pin by using the `get_power_per_pgpin` command and use the output of this command to generate custom reports. The syntax is:

```
get_power_per_pgpin -power_type [switching | leakage | internal | total]
```

To extract the total power for each PG pin, use the `get_power_per_pgpin` command and specify a collection of cells. By default, the command reports the total power of a hierarchical net connected to the PG pins of a cell. Example 7-8 shows how to use the command. In the example, the PG pin vdd of cella is connected to the supply net vdd1, and the PG pin vdds is connected to supply net vdd2.

Example 7-8 Reporting the Power for Each PG Pin

```
    pt_shell>  get_power_per_pgpin [get_cells *cell*]
    {cella {vdd 'vdd1' 1.056e-06} {vdds 'vdd2' 6.424e-09}}
    {cellb {vdd 'vdd1' 1.865e-06} {vdds 'vdd3' 1.347e-08}}
    {cellc {vdd 'vdd1' 1.374e-06} {vdds 'vdd2' 1.237e-08}
```

You can extract the power for each PG pin in the following multivoltage analysis flows:

- UPF

  You can extract the power for each PG pin by querying the power for each power net and associating it with the PG pin, unless the power net is connected to multiple PG pins.

Using the `get_power_per_pgpin` command, you can extract the power for each PG pin even if multiple cell PG pins are connected to the same power net.

- Rail-mapping

  In rail-mapping mode, the `get_power_per_pgpin` command enables reporting of power for each PG pin without the additional scripting required to identify the supply nets connected to each PG pin. Because the `power_pin_info` attributes do not include the connected supply net per PG pin, mapping the supply net to the PG pin requiring parsing of the rail-mapping scripts is not necessary.

The output of the `get_power_per_pgpin` command lists all PG pin types except for generic ground pins (primary ground, backup ground, internal ground) whose voltage and associated power is always 0. The power for the following PG pin types are reported:

- PG power pins: primary power, backup power, and internal power
- Substrate bias pins: pwell, nwell, deeppwell, and deepnwell

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

Multivoltage Power Analysis Using UPF

In the PrimeTime PX tool, use the UPF file to specify your power intent for the analysis of your multivoltage design. The UPF file provides the ability to specify the power intent early in the design process, and is supported throughout the design flow.

When you specify the power intent for UPF designs, the PrimeTime PX tool converts and update the library power and ground (PG) pins by default using the specifications provided in a Tcl file.

For more information, see the *PrimeTime User Guide*.

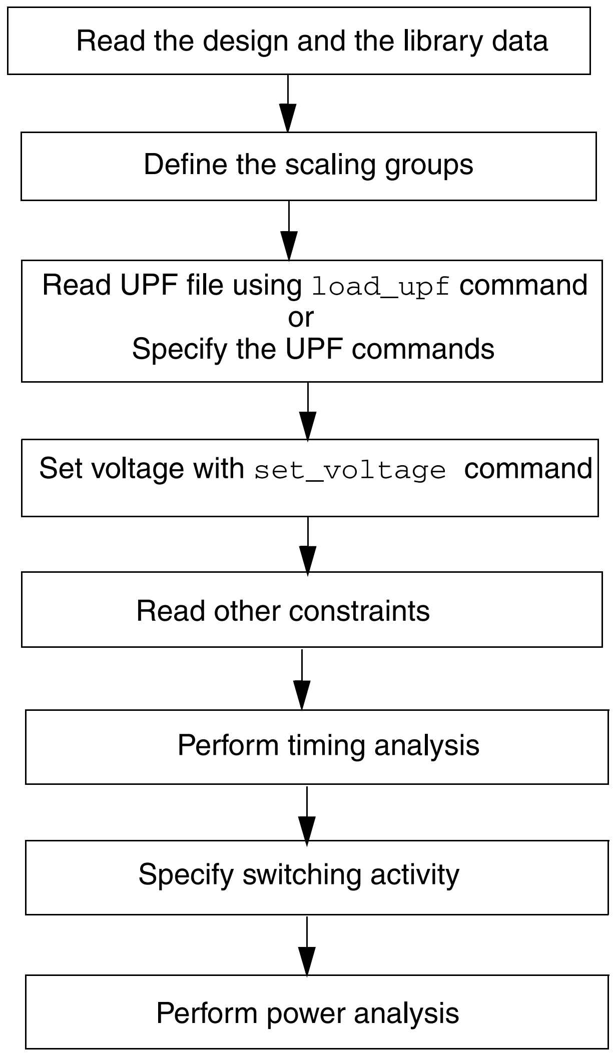
Figure 7-1 shows the UPF flow for multivoltage power analysis using PrimeTime PX.

To specify the UPF commands,

- Use the `create_power_domain` command to create power domains; use the `create_supply_port` command to create supply ports; and use the `create_supply_net` command to create supply nets.
- Use the `connect_supply_net` and `set_domain_supply_net` commands to create and define power connections.

You can specify the UPF commands directly at the tool prompt or by using the `load_upf` command if they are contained in a separate UPF file.

Figure 7-1 Multivoltage Power Analysis Flow Using UPF


Read the design and the library data
Define the scaling groups
Read UPF file using load_upf command or Specify the UPF commands
Set voltage with set_voltage command
Read other constraints
Perform timing analysis
Specify switching activity
Perform power analysis

For more information about low-power flow and various Synopsys tools that support UPF, see the *Synopsys Multivoltage Flow User Guide*.

When you invoke the PrimeTime PX tool, the power analysis using UPF flow is enabled by default.

Power Domains

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

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

The logical hierarchy where the power domain is created is called the scope of the power domain. Use the `set_scope` command to specify the scope or level of the hierarchy of the power domain.

Use the `create_power_domain` command to create a power domain. 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. Use the `-include_scope` option to specify that all the elements in the current scope share the primary supply of the power domain, but are not necessarily a part of the power domain. Use the `-scope` option to specify the logical hierarchy or the scope where the power domain is to be defined.

Use the `create_supply_net` and `create_supply_port` commands to create ports and nets respectively, for the specified power domain and `connect_supply_net` command to connect the supply net to the specified supply ports and pins.

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 ground net for an existing power domain.

Isolation Cells

For 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 functions as a buffer when the input and output of the isolation cell are both powered up. When the input of the cell is powered down, the isolation cell provides a constant output signal. The enable input controls the operating mode of the isolation cell.

Precedence Rules

To identify the supply sets or nets and the corresponding voltages for isolation, the PrimeTime PX tool uses the following precedence rules:

- Isolation supply nets

  The source or sink property of a net for isolation corresponds to all the net segments connected, including the nets that connect to the level shifters and isolation cells. The dangling isolation and level-shifter cells are treated as source or sink. The tool stops the tracing at the retention cells, standard cells, black box cells and boundary ports. It chooses the supply set or nets for isolation based on the following criteria, in the following order of precedence:

  1. Supply set or nets that have the `iso_source` or `iso_sink` attribute set by the `set_port_attributes` command.
  2. Supply set or nets that are related supply nets, defined by the `set_related_supply_net` command.
  3. Top-level supply set or nets at the boundary port.

  Note:
  When the supply set or nets are found, no further selection is made.

- Supply net voltages

  To determine the supply voltage of each PG pin for the cells in the design, the PrimeTime PX tool uses the following order of precedence, from highest to lowest. The UPF port voltage has higher precedence over the non-UPF port voltage.

  ❍ The `-driver_supply` or `-receiver_supply` option with the `-ports` option for the driver supply on the input ports and the receiver supply on the output ports.

  ❍ The `-driver_supply` or `-receiver_supply` option with the `-elements` option for the driver supply on the input ports and the receiver supply on the output ports.

  ❍ The `set_related_supply_net` command.

  ❍ The primary supply of the power domain of the top-level design.

  ❍ The `set_voltage` command on a PG pin. This requires a PrimeTime SI license.

  ❍ The `set_voltage` command on a supply net.

  ❍ The `set_operating_conditions` command specified at the design level.

  ❍ The default supply voltage in the library cell definition specified by the `voltage_map` attribute in the Liberty syntax.

Commands to Define Isolation Strategy

Use the `set_isolation`, `set_isolation_control`, `set_port_attributes`, and `set_design_attributes` UPF commands to specify isolation strategies in the PrimeTime PX UPF flow.

set_isolation

Use the `set_isolation` command to define the UPF isolation strategy for the specified power domains. An isolation strategy includes specification of the enable signal net, the clamp value, and the location; inputs, outputs, or both. This command creates an explicit connection between the isolation power and ground nets and the power and ground pins of the isolation cell.

Every isolation strategy that you define using the `set_isolation` command must have a corresponding `set_isolation_control` command, unless you have specified the `-no_isolation` option.

set_isolation_control

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

set_port_attributes

Use the `set_port_attributes` command to specify information that is relevant to the ports on the interface of the power domains. The information is used to verify the isolation and guard requirements for the port.

set_design_attributes

Use the `set_design_attributes` command to specify a collection of cells where the attributes are set for the source or sink of the power domains, when you use with the `set_isolation` command.

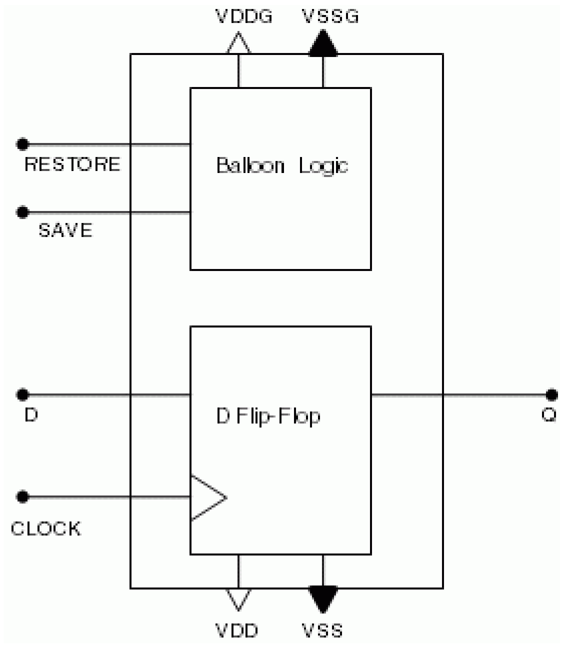
The `-elements` option specifies a collection of cells where the attributes must be set. Use the `-attribute` option to specify the *name* and *value* of the attributes to set on the cells when you specify the `-elements` option. When you do not specify `-elements` option, the `-attribute` option is set on the current top-level design. The *name* is the one set by `external_supply_map` and the *value* is the reference name of a supply set or a supply-net pair.

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 7-2, 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 7-2 Two Pin Retention Register



Retention Commands

Use the retention commands to specify the strategy for inserting retention cells inside switched (power-down) domains during synthesis. When specified, the tool verifies the syntax and power connections.

set_retention

Use the `set_retention` command to specify the registers in the domain implemented as retention registers and identifies the save and restore signals for the retention function. Specify one of the `-retention_power_net` and `-retention_ground_net` options at the least. Specifying both options indicates the supply nets are retention power and ground nets. If you specify only the retention power supply net, the tool uses the primary ground net as the retention ground supply. If you specify only the retention ground net, the tool uses the primary supply net as the retention power supply. The tool connects the retention power and ground nets to the implicit save and restore processes and shadow register.

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

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

Define a retention strategy using the `set_retention` command with a corresponding `set_retention_control` command.

set_retention_control

Use the `set_retention_control` command to specify 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.

Power Gating in UPF Mode

In the UPF mode, the PrimeTime PX tool supports power-gating-aware power calculation to measure the reduction in 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 transforms 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 a Boolean expression.

Use the `create_power_switch` command to define a power switch in a power domain. The power switch is created within the scope of the specified power domain. This command specifies 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 enabled 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`.

The PrimeTime PX tool supports single input, single output, multiple control switches. Use the `-domain` option to specify the power domain where the power switch is to be created.

Use the `-output_supply_port` option to specify the output port of the power switch and the supply net to connect this output port.

Use the `-control_port` option to specify the control port of the power switch and the net where this control port is to be connected. Because multiple control ports are supported for the power switch, use the option to specify multiple pairs of control ports and the associated nets. Use the `-on_state` option to specify the state, the associated input port and the Boolean function.

Note:

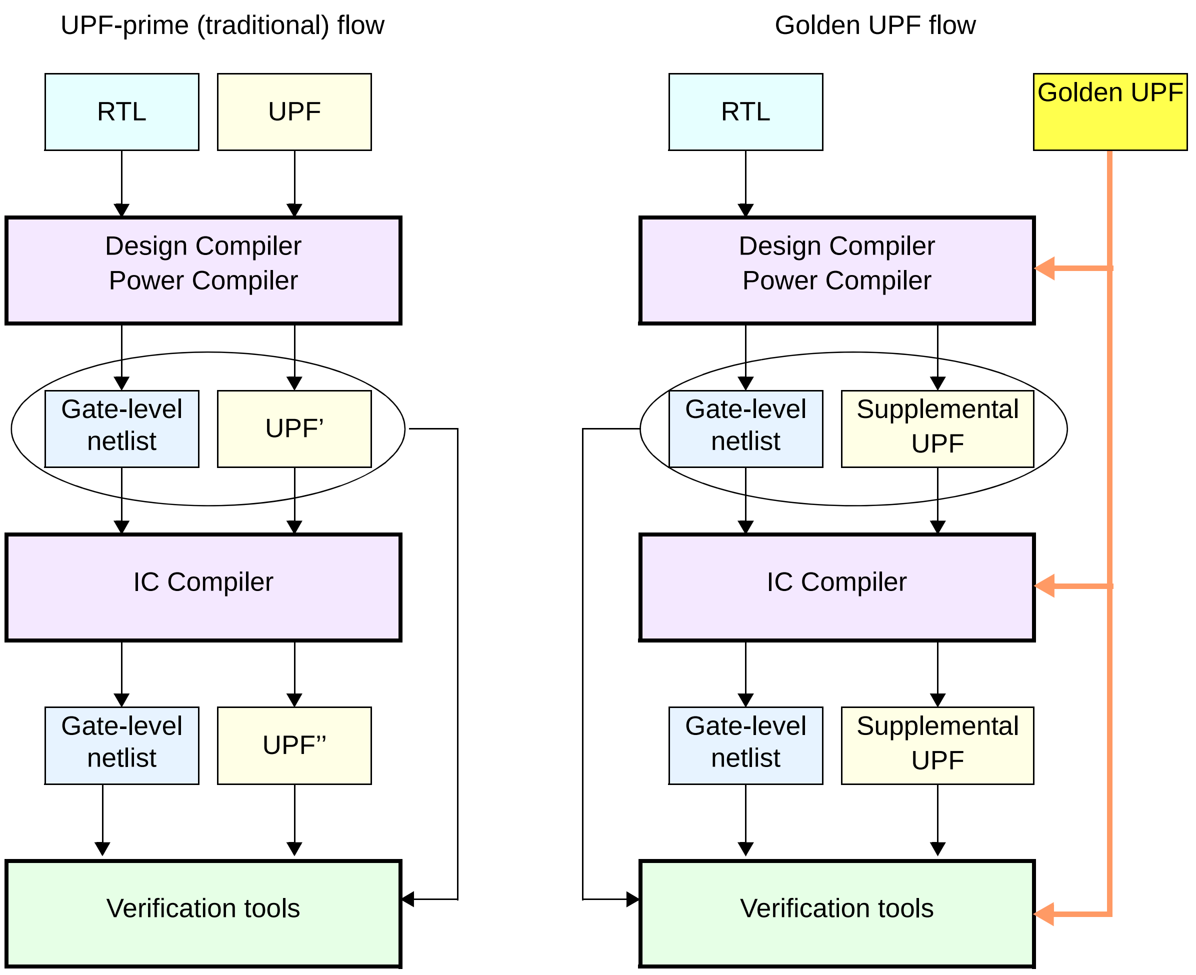
> The PrimeTime PX tool does not support additional options of the `create_power_switch` command, such as `-ack_port`, `-on_partial_state`, `-ack_delay`, `-off_state`, and -`error_state`, and ignores them without generating any warning message.
>
> The tool does not support other UPF commands related to power switches, such as `map_power_switch` and `set_power_switch`, and ignores by generating a warning message.

Use the `report_power_switch` command to report all the power switches defined in the design. This command reports details such as the names of the power switches, the power domains where the switches are defined, input, output and control port details. This command is not an IEEE 1801 compliant UPF standard command.

Golden UPF Flow

The golden UPF flow is an optional method of maintaining the UPF multivoltage power intent of the design. It uses the original “golden” UPF file throughout the synthesis, physical implementation, and verification steps, along with supplemental UPF files generated by the Design Compiler and IC Compiler tools. Figure 7-3 compares the traditional UPF flow with the golden UPF flow.

Figure 7-3 UPF-Prime (Traditional) and Golden UPF Flows



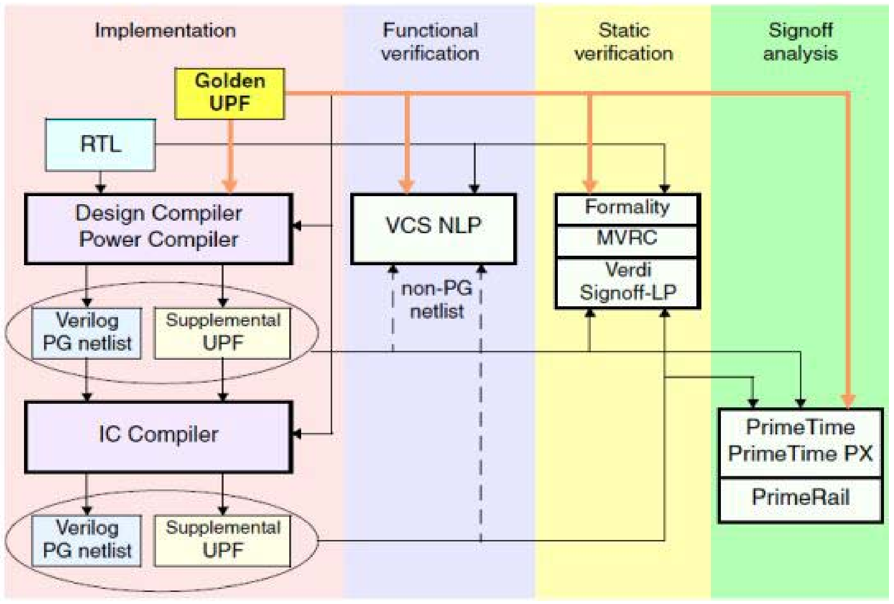
The golden UPF flow maintains and uses the same, original “golden” UPF file throughout the flow. The Design Compiler and IC Compiler tools write power intent changes into a separate “supplemental” UPF file. Downstream tools and verification tools use a combination of the golden UPF file and the supplemental UPF file, instead of a single UPF’ or UPF” file.

The golden UPF flow offers the following advantages:

- The golden UPF file remains unchanged throughout the flow, which keeps the form, structure, comment lines, and wildcard naming used in the UPF file as originally written.
- You can use tool-specific conditional statements to perform different tasks in different tools. Such statements are lost in the traditional UPF-prime flow.
- Changes to the power intent are easily tracked in the supplemental UPF file.
- You can optionally use the Verilog netlist to store all PG connectivity information, making `connect_supply_net` commands unnecessary in the UPF files. This can significantly simplify and reduce the overall size of the UPF files.

  Figure 7-4 shows how the golden UPF file is used by the PrimeTime PX and other Galaxy verification tools throughout the flow, in combination with a supplemental UPF file generated by the Design Compiler or IC Compiler tool.

Figure 7-4 Verification Tools in the Golden UPF Flow



To use the golden UPF flow, you must enable it by setting a variable:

```
pt_shell> set_app_var enable_golden_upf true
```

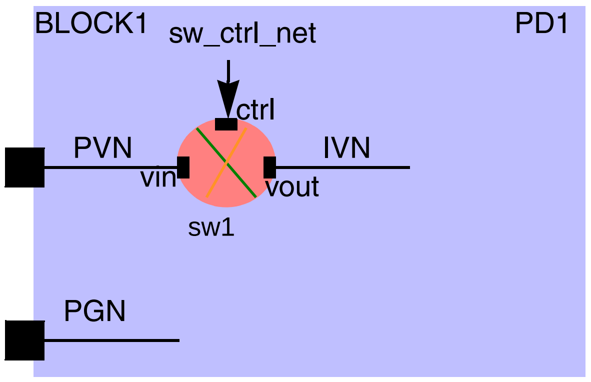
For more information about using the golden UPF flow, see SolvNet article 1412864, “Golden UPF Flow Application Note.”

Support for Switch Cells for Power Analysis in UPF Mode

Figure 7-5 shows a power switch for a block. The block, Block1, has the 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 Block1 are powered down by the switch, sw1, based on the Boolean function that you specify with the `create_power_switch -on_state` command.

Figure 7-5 Power Switch



Example 7-9 shows a typical script used for power analysis.

Example 7-9   Script Example for Power Analysis

```
# Read the target libraries and input designs
set_app_var power_enable_analysis true
set_app_var power_analysis_mode time_based

set_app_var 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

# Perform timing analysis
update_timing

# 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
```

Support of UPF Footer Switches

The PrimeTime PX tool 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. The tool supports the single and multiple ground cells controlled by UPF footer switches in the following ways:

- Single-ground cells

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

- Multiple ground cells

  For this 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, the target library must comply with the PG pin Liberty library syntax. Both the PrimeTime and PrimeTime PX tools 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 information, see the *PrimeTime User Guide*.

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 and for both CCS and NLPM power library formats. The library cells are characterized at different operating conditions and present in separate libraries. Because of the support of voltage and temperature scaling, during power analysis, the data can be interpolated from these separate libraries for the desired operating condition.

Use the `set_voltage` command 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 the PrimeTime PX tool in the UPF mode.

Note:
The operating voltage that you define on the 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.

Multivoltage Power Analysis Using Power Rails

To report power by rails, 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, the power of G1, G2, and G3 is associated with the V1 design rail; B1 and B2 are tied to the V2 design rail, and R1 to the V3 design rail.

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

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

The report generated by the `report_power` command contains the power consumption only for the last specified rail. The rails for which power is calculated are defined in the Current Voltage Rail heading of the report, as shown in Example 7-10.

Example 7-10 Report Example Showing Power Consumption for the Specified Rail

```
****************************************
Report : power
Design : d
...
****************************************
Sampling Interval: 1 ns
Current Voltage Rail: VDD2

Library(s) Used:

    pg_pin_lib (File: power_level.db)

Operating Conditions: WORST   Library: power_level
Wire Load Model Mode: top

<no wire load model is set>
```

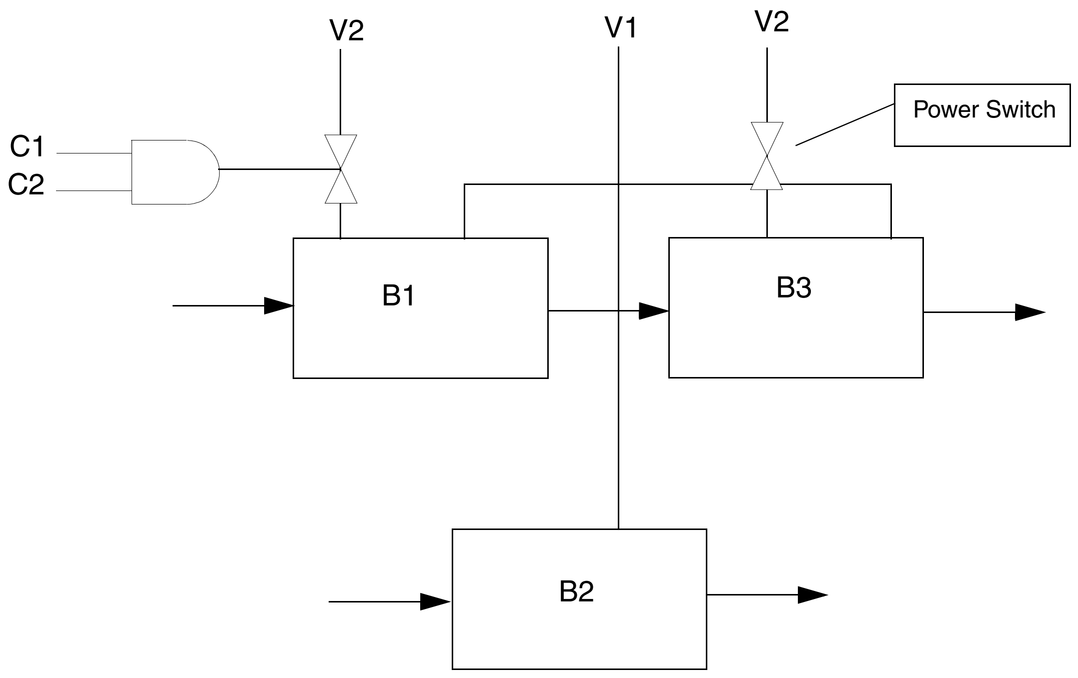
```
Power-specific unit information :
    Voltage Units = 1 V
    Capacitance Units = 1 pf
    Time Units = 1 ns
    Dynamic Power Units = 1 W
    Leakage Power Units = 1 W

  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, map the power rails defined in the libraries to the physical power rails existing in the design using the `create_power_rail_mapping` command. Use the `-lib_rail_name` option to map the power rails in each library associated with the library cells. For example,

```
pt_shell> create_power_rail_mapping V3 -lib_rail_name Vdd0 -cells B1
```

Figure 7-6 shows a design example that has multiple power rails. The V2 rail, which feeds Block-3 and portions of Block-1, can be cut off when the blocks are not used for a certain duration.

Figure 7-6 Design With Multiple Power Rails



As shown in Figure 7-6, the power rail V2 is power-gated. The second power rail mapping command shown in Example 7-11 is mandatory (see texts in bold); it defines the power-off condition and the power rail controlled by power gating. After you run the command, you can use the `update_power` and `report_power` commands. You can check the rail mapping as well. Example 7-11 is the script example:

Example 7-11 Script Example

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

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

create_power_rail_mapping V1 -lib_rail_name Vdd1 -cells "B1 B2 B3"
create_power_rail_mapping V2 -lib_rail_name Vdd2 \
     -off_condition "C1 & C2" -cells R1

report_power_rail_mapping

current_power_rail V1
report_power
```

Power-Gating With Power Rails

Power gating is an effective leakage power saving mechanism; it saves power by dynamically disabling the power supply to certain parts of the design. Both dynamic and leakage power consumptions take place inside the block. If a block is not used for a certain duration, its power supply can be cut off for saving leakage power.

To analyze the power of designs with power gating, use the `create_power_rail_mapping -off_condition` command to specify the conditions when the power rail is disabled. For example,

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

When you specify VCD activity, the PrimeTime PX tool monitors the activity over time, and accurately analyzes power only during the time windows when the power-off condition is `false`. When the condition is `true`, the power consumption is zero.

If you provide SAIF or user-defined activity, the PrimeTime PX tool uses the static probability of the power-off condition to scale the leakage power only; the tool assumes that the activity captures the toggles when the power is on. If the activity is obtained from simulation, which assumes that the power is always on, set the `power_scale_dynamic_power_at_power_off` variable to `true`. The dynamic power as well as the leakage power will be scaled.

Voltage Scaling With Power Rails

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

For libraries with power and ground pins, the PrimeTime PX tool 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.

You can set the values of single and multirail cells and modify the default rail voltages of single-rail or multirail cells. The tool uses the specified values to calculate the correct power consumption.

Use the `set_rail_voltage` command for a particular cell 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 drop on individual cells to ascertain the effects.

Reporting Power Rail Mapping

For a design with multiple VDD cells, use the `report_power_rail_mapping` command to identify the power rails in the library before creating the power rail mapping between the rails in the design and libraries. After linking the design and the libraries, this command reports the libraries and their power and ground rails.

By default, the tool 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, the tool needs to know the power rails in the design as well as the rails in the library.

Non-UPF Multivoltage Power Analysis Using Power Domains

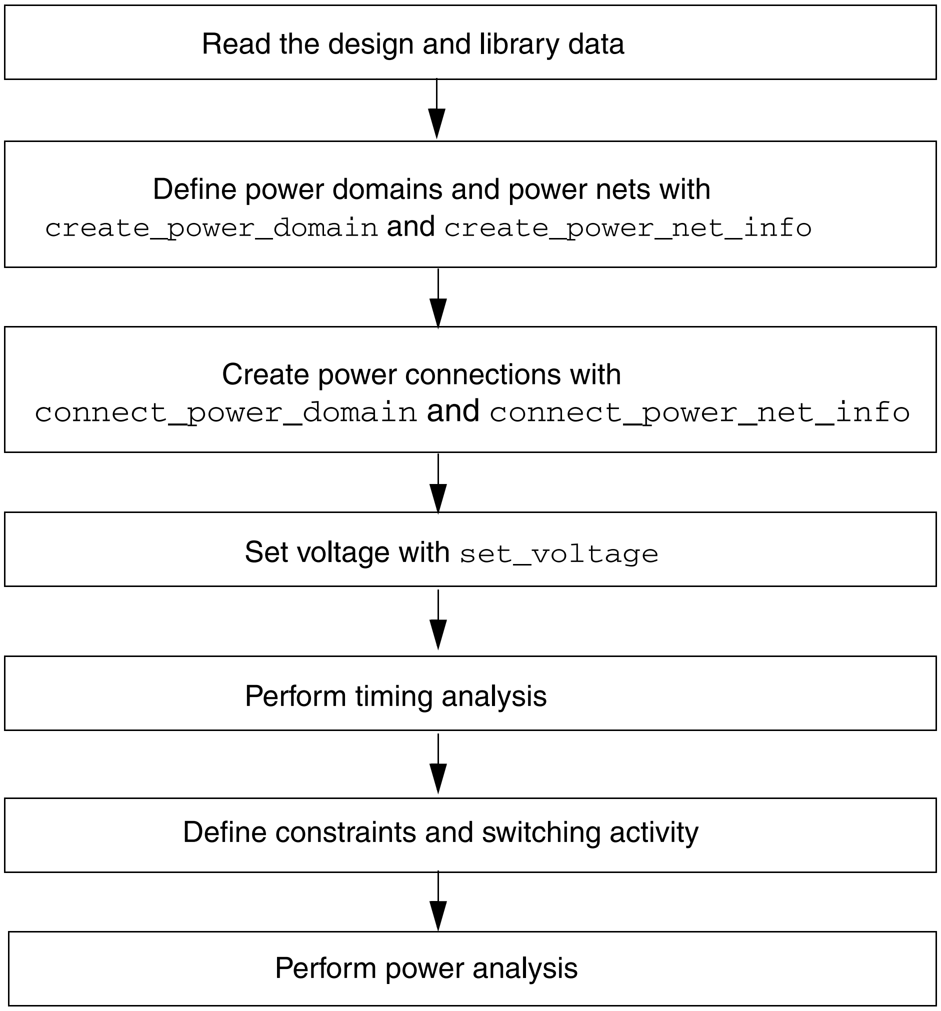
To invoke PrimeTime PX tool in the non-UPF mode, set the `power_domains_compatibility` variable to `true`. This variable reverts back to power domains and disables UPF mode for power analysis. The default is `false`. To get a list of commands supported in this mode,

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

Figure 7-7 shows the flow for multivoltage power analysis when you use power domains.

For more information about domains, see the *PrimeTime User Guide*.

Figure 7-7 Multivoltage Power Analysis Flow Using Power Domains


Read the design and library data
Define power domains and power nets with create_power_domain and create_power_net_info
Create power connections with connect_power_domain and connect_power_net_info
Set voltage with set_voltage
Perform timing analysis
Define constraints and switching activity
Perform power analysis

The following commands support multivoltage power analysis using domains:

- `create_power_domain`

  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 do not specify a list. There can be only one top-level domain.

- `connect_power_net_info`

  Defines 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.

- `connect_power_domain`

  Creates logical power connections for a specified power domain. You can specify 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. Linking the design again causes the tool to discard power domain information.

Power Gating Using Power Domains

To perform power gating, use the `create_power_domain` command with the following options for the power down specifications:

- `-power_down` to indicate that the specified domain can be powered down.
- `-power_down_ctrl` *`net_list`* to specify the nets that control the power to the specified domain. These nets can power down the domain.
- `-power_down_ctrl_sense 0 | 1` to specify 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. The PrimeTime PX tool 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:

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

Use the `create_power_net_info` command to specify the name of a power supply net or ground net in the design. For power supplies, you can also specify the voltage and the condition to disable the power supply.

Voltage Scaling Using Power Domains

Power domains can contain multiple power nets, where the type and default voltage values are defined using the `create_power_net_info` command. The voltage of the power nets can be overwritten using the `set_voltage` command. The PrimeTime PX tool scales the power consumption per power net, based on the specified voltage.

Use the `set_voltage` command to define the operating voltage on the power nets created using 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 specify this command, the tool uses the existing operating condition settings. To report the operating voltages of power nets, use the `report_power_net_info` command.

Example 7-12 shows how to perform multivoltage power analysis using power domains.

Example 7-12 Script to Perform Multivoltage Analysis Using Domains

```
# Read libraries, design, enable power analysis
# and link design
set_app_var power_enable_analysis true
set_app_var 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_info 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 SDC and other timing or power assertions
```

```
set_input_transition 0.0395 [all_inputs]
set_load 1.0 [all outputs]
# Perform timing analysis
update_timing
# Read switching activity
set_switching_activity...
set_switching_activity...
...
report_power
```

8

Clock Network Power Analysis

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:

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 the `report_power -include_estimated_clock_network` option to report the clock network power estimated by the `report_power` command.

For more information, see Chapter 9, “Generating Reports.”

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 designs, 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 as a combination. However you cannot combine any of these three options with the `-total_power` option.

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

Estimating Clock Network Power Consumption

You can estimate the additional power incurred by a clock network before 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, 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 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.

Example 8-1shows the output of the `estimate_clock_network_power` command. It includes the power consumption as well as detailed information about the pseudo-clock network, driven registers, and timing propagation.

Example 8-1 Report Generated by the estimate_clock_network_power Command

```
****************************************
Report : estimated clock network power
Design : reg16
...
****************************************

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
```

To view the total power of the design with the estimated clock network, specify the `report_power -include_estimated_clock_network` command. The clock network power is added to the total design power.

When you specify the `-include_registers` option of the `estimate_clock_network_power` command, the register clock-pin power is recalculated using the estimated clock network transition times. Because of the change in the register clock pin power, the total power is different from the results generated without this option.

9

Generating Reports

The PrimeTime PX tool can generate a wide range of reports that provide information about the power consumption, as described in the following sections:

- Generating Power Reports
- Generating Reports on Intrinsic and Gate Leakage Power
- Generating Reports on Threshold Voltage Groups
- Generating Reports on Power Derating Factors
- Generating Custom Reports with Attributes
- Generating Power Calculation Reports

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

Generating Power Reports

By default, the `report_power` command outputs the top-level power consumption. You can generate four types of reports:

- Group-based (the default) when the `-cell_power`, `-net_power`, and `-hierarchy` options are not specified (see Example 9-1 on page 9-4).
- Cell-based, with the `-cell_power` option.

  Use the `-sort_by` option to sort the report by `name`, `cell_internal_power` (the default), `cell_leakage_power`, or `dynamic_power` (see Example 9-4 on page 9-8).
- Net-based, with the `-net_power` option.

  Use the `-sort_by` option to sort the report by `name`, `net_static_probability`, `net_switching_power` (the default), `net_toggle_rate`, or `total_net_load` (see Example 9-5 on page 9-9).
- Hierarchy-based, with the `-hierarchy` option (see Example 9-6 on page 9-10).

The `-groups`, `-rail`, 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 rail, power group, and hierarchy.

Power Group Report

When none of the `-cell_power`, `-net_power`, or `-hierarchy` option is specified, the `report_power` command classifies the power consumption into categories, referred to as power groups, based on the cell type. There are seven predefined power groups, based on the rules shown in Table 9-1:

Table 9-1 Predefined Power Groups

Power group	**Cell type**	**Identification**
`io_pad`	I/O pad cells	attribute: `is_pad_cell`: `true`
memory	Memory cells, which are identified in multiple ways	attribute: `is_memory_cell`: `true`
`black_box`	Cells with no functionality, excluding `io_pad` and memory cells	attribute: `is_black_box`: `true`

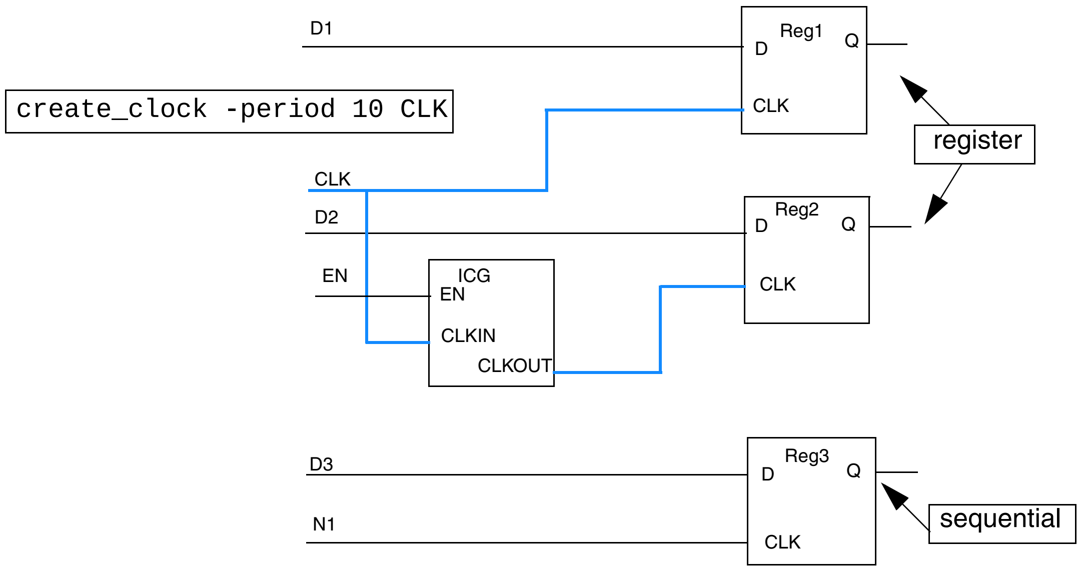
Table 9-1 Predefined Power Groups (Continued)

Power group	Cell type	Identification
`clock_network`	Cells in the clock tree network, excluding I/O pad cells	The `create_clock` command is used to identify the clock. Cells in the clock network are buffers, inverters and clock-gating logic through which the clock is traced.
register	Latches and flip-flops, excluding clock-gating logic, driven by a clock	Register clock pin power is placed in the `clock_network` group, or the register group, based on the value of the `power_clock_network_include_register_clock_pin_power` variable. The default setting is `true`. The register clock pin power is reported as `clock_network` power, while the remaining register power is reported as register power. When set to `false`, the clock pin power is included in the register group power.
sequential	Latches and flip-flops excluding clock-gating logic and not driven by a clock	The clock pin of the sequential cell is not connected to the clock network.
combinational	Combinational logic	Any remaining cells not identified as belonging to the other power groups.

When you specify a `group_list` value with the `-groups` option of the `report_power` command, the tool generates a power report for the cells in the specified `power_group`. By default, all the instances are placed in only one power group. Therefore, the summation of the power of the predefined groups equals the total power consumption of the design.

Figure 9-1 highlights the difference between register and sequential power groups. Reg1 and Reg2 are in the register power group because their clock pins are tied to nets which are part of the clock network as defined by the `create_clock` command. Reg3 is in the sequential power group because its clock pin is not part of the clock network.

Figure 9-1 Difference Between Registers and Sequential Power Groups



Example 9-1 shows the power group report without the `-cell_power`, `-net_power`, and `-hierarchy` options specified:

Example 9-1 Power Group Report Without Options Specified

```
pt_shell> report_power
**********************************************
Report: Averaged Power
Design: mac
...
**********************************************
  Attributes
      i  -  Including register clock pin internal power
      u  -  User-defined power group
Power           Internal   Switching   Leakage    Total
Group           Power      Power       Power      Power    (     %)  Attrs
--------------------------------------------------------------------------
clock_network   8.529e-04 1.599e-05  4.398e-04   8.689e-04(40.56%)   i
register        1.882e-04 3.482e-05  6.666e-08   2.230e-04(10.41%)
combinational   4.068e-04 5.646e-04  1.836e-07   9.716e-04(45.36%)
sequential      2.327e-05 5.522e-05  9.163e-09   7.851e-05( 3.67%)
memory          0.0000    0.0000     0.0000      0.0000(0.00%)
io_pad          0.0000    0.0000     0.0000      0.0000(0.00%)
black_box       0.0000    0.0000     0.0000      0.0000(0.00%)
--------------------------------------------------------------------------
Net Switching Power  = 6.707e-04 (31.31%)
Cell Internal Power  = 1.471e-03 (68.68%)
Cell Leakage Power   = 2.594e-07 ( 0.01%)
------------------------------------------
```

```
Total Power          = 2.142e-03 (100.00%)
-----------------------------------------
1
```

You can specify the `-include_estimated_clock_network` option only when generating the power group reports and provided you have already specified the `estimate_clock_network_power` command.

Creating Power Groups

You can create your own power groups to classify cells into groups using the `create_power_group` command. Cells in each group need not be mutually exclusive. When defining your own power group, use a unique name.

For more information about the predefined power groups, see the `create_power_group` command man page.

Using Power Groups

To remove predefined or user-specified power groups, use the `remove_power_groups` command.

To return a list of cells contained within specified predefined or user-defined power groups, use the `get_power_group_objects` command.

To list the predefined and user-defined power groups, use the `report_power_groups` command.

To report the power information for objects in power groups, use the `report_power -groups` *`power_group_list`* command. This command generates a report that contains power data for objects in the predefined or user-defined power groups.

For more information about these commands, see the man pages.

Overriding the Default Power Group Classification

To override the default power group classification, use the `power_cell_type` attribute to classify cells into different default power groups. You can use the `power_cell_type` attribute in both the averaged and time-based power analysis modes.

You can set the value of this attribute to one of the following default power groups: `clock_network`, `register`, `combinational`, `sequential`, `memory`, `io_pad`, and `black_box`. You cannot use the `power_cell_type` attribute to assign a cell to a user-defined power group.

The `power_cell_type` attribute is a cell-level and a library-cell level attribute. To define this attribute, use the `set_user_attribute` command. For example,

- To assign a cell to the register power group,

  ```
  set_user_attribute [get_cell U0/U1] power_cell_type register
  ```

- To assign a cell instance from a library to the register power group,

  ```
  set_user_attribute [get_lib_cells lib1/REF1] power_cell_type register
  ```

The cell-level attribute definition overrides the library-cell level attribute definition.

The `power_cell_type` attribute value takes precedence over the default group classification. For example, a cell in the black_box power group, can be reclassified as a register, by setting the `power_cell_type` attribute to register. As part of the register group, the PrimeTime PX tool associates any clock pin power of the cell to the clock_network power group.

The resulting power report includes the cell with the `power_cell_type` attribute set to register, as part of the register power instead of black box power.

The following example shows the results of the `report_power` command using the default power group classification, followed by the results of the `report_power` command after the `power_cell_type` attribute is used to move the cells from the black box power group to the register power group.

Example 9-2 shows a power report generated by the `report_power` command when the `power_cell_type` attribute is not set.

Example 9-2 Power Report Before Setting the power_cell_type Attribute

```
pt_shell> report_power

***********************************************************************
Report: Averaged Power
...
***********************************************************************
Attributes:
---------------------------------------------------
     i - Including register clock pin internal power
     u - User-defined power group

Power Group     Internal   Switching   Leakage    Total
Attrs
                Power      Power       Power      Power(  %)
-----------------------------------------------------------------------
----
clock_network   6.144e-06  1.750e-07   8.774e-08  6.406e-06(70.32%)
register        4.295e-07  4.614e-07   1.152e-06  2.043e-06(22.42%)
combinational   0.0000     0.0000      0.0000     0.0000
sequential      0.0000     0.0000      0.0000     0.0000
```

```
memory          0.0000     0.0000      0.0000     0.0000
io_pad          0.0000     0.0000      0.0000     0.0000
black_box       4.017e-07  2.326e-08   2.370e-07  6.620e-07( 7.27%)
-----------------------------------------------------------------------
---

     Net Switching Power = 6.597e-07 ( 7.24%)
     Cell Internal Power = 6.975e-06 (76.56%)
     Cell Leakage Power  = 1.476e-06 (16.20%)
---------------------------------------------------
     Total Power         = 9.111e-06 (100.00%)
---------------------------------------------------
1
```

After you have set the attribute, generate a power report, as shown in Example 9-3:

Example 9-3 Power Report After Setting the power_cell_type Attribute

```
pt_shell> set_user_attribute [get_lib_cells lib1/REF1] power_cell_type register
pt_shell> report_power
************************************************************************
Report: Averaged Power
...
************************************************************************
Attributes
---------------------------------------------------
         i - Including register clock pin internal power
         u - User-defined power group
---------------------------------------------------
Power Group      Internal   Switching    Leakage    Total           Attrs
                 Power      Power        Power      Power(  %)
------------------------------------------------------------------------
clock_network    6.144e-06  1.750e-07    8.774e-08  6.406e-06(70.32%)
register         8.312e-07  4.847e-07    1.389e-06  2.704e-06(29.68%)
combinational    0.0000     0.0000       0.0000     0.0000  (0.00%)
sequential       0.0000     0.0000       0.0000     0.0000  (0.00%)
memory           0.0000     0.0000       0.0000     0.0000  (0.00%)
io_pad           0.0000     0.0000       0.0000     0.0000  (0.00%)
black_box        0.0000     0.0000       0.0000     0.0000  (0.00%)
------------------------------------------------------------------------

         Net Switching Power = 6.597e-07 ( 7.24%)
         Cell Internal Power = 6.975e-06 (76.56%)
         Cell Leakage Power  = 1.476e-06 (16.20%)
----------------------------------------------------
         Total Power         = 9.111e-06 (100.00%)
----------------------------------------------------
1
```

The highlighted text in the report example shows the power values shifted from the `black_box` power group to the register power group after setting the `power_cell_type` attribute.

Cell-Based Report

To generate a cell-based report, specify the `-cell_power` option with the `report_power` command. The cell-based report lists the internal, switching, leakage, and total power per cell. By default, only the cells (including hierarchical cells) in the current hierarchy are displayed.

You can specify a cell list, or the `report_power` filter to report only cells in the specified power group, clock domain, or power rail.

Example 9-4 shows a cell-based power report for all of the cells in the clock network power group:

Example 9-4 Cell-Based Power Report for Predefined Clock Network Power Group

```
pt_shell> report_power -cell_power -groups clock_network
****************************************
Report : Averaged Power
        -cell_power
        -sort_by cell_internal_power
        -power_greater_than       0
        -groups clock_network
Design : test
...
****************************************
  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
```

By default, the cells are sorted based on their internal power values. Use the `-sort_by` option to sort cells based on the leakage, switching, or total power.

Net-Based Report

To generate a net-based report, specify the `-net_power` option with the `report_power` command. The net-based report lists the power rail, total net load, static probability, toggle rate, and switching power per net. By default, only nets in the current hierarchy are displayed, sorted by the net switching power.

Example 9-5 shows a net-based report generated using the `-net_power`, `-leaf`, and `-nworst` options of the `report_power` command. The generated report is sorted by net switching power and filtered to display only the five nets with the highest switching power.

Example 9-5 Report Generated by the report_power Command With -net_power, -leaf, and -nworst Options

```
pt_shell> report_power -net_power -leaf -nworst 5
****************************************
Report : Averaged Power
        -net_power
        -nworst 5
        -leaf
        -sort_by net_switching_power
       -power_greater_than      0
Design : test
...
****************************************
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
```

Specifying the `-leaf` option includes leaf instance power dissipation in the report. Reporting to the leaf level is useful specifically when the leaf cells are scattered throughout the hierarchy or when you want to perform detailed analysis.

Hierarchy-Based Report

To generate a hierarchical report that displays the power consumption per hierarchy, specify the `-hierarchy` option of the `report_power` command. By default, only the first level is shown. Additional levels can be displayed with the `-levels` option.

Example 9-6 shows a hierarchy-based power report for an instance named `mac`:

Example 9-6 Hierarchy-Based Power Report

```
pt_shell> report_power -hierarchy -levels 2
****************************************
Report : Averaged Power
        -hierarchy
        -levels 2
Design : mac
...
****************************************
                       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
```

Generating Reports on Intrinsic and Gate Leakage Power

When CCS libraries are provided, the cells might contain separate intrinsic and gate leakage values. By default, the tool reports the total leakage of a cell as a single value that represents the sum of the intrinsic and gate leakage.

To report the cell leakage power as the sum of gate and intrinsic (power-to-ground) leakage power,

- Ensure that the logic library has gate leakage and intrinsic leakage characterized for the library cells.
- Ensure that the `power_report_leakage_breakdowns` variable is set to `true`. The default is `false`.

The following example shows the gate leakage and intrinsic leakage information in the power report:

```
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%)
-------------------------------------------
```

Generating Reports on Threshold Voltage Groups

In the PrimeTime PX tool, you can report information about threshold voltage groups using the `report_threshold_voltage_group` command and `report_power -threshold_voltage_group` commands.

The `report_threshold_voltage_group` command reports the number and percentage of the cells in each threshold voltage group. When you specify the `-threshold_voltage_group` option with the `report_power` command, the tool reports the leakage power per threshold voltage group. Use the reports to compare the leakage power of your design for various threshold distributions.

Identifying the Threshold Voltage Group

By default, the `report_threshold_voltage_group` command identifies the threshold voltage groups to which the cells can be categorized using the following attributes at the library and cell level:

- Library-level attribute: `default_threshold_voltage_group`
- Cell-level attribute: `threshold_voltage_group`

When the PrimeTime PX tool does not find these attributes, you can set the attributes in the library using the `set_user_attribute` command.

The following example shows the standard and low threshold voltage attribute set on a library.

```
pt_shell> set_user_attribute [get_libs agl90g_od_svt_ss] \
               default_threshold_voltage_group SVT

pt_shell> set_user_attribute [get_libs agl90g_od_lvt_ss] \
               default_threshold_voltage_group LVT
```

The following example shows the standard and low threshold voltage attribute set on a cell:

```
pt_shell> set_user_attribute [get_lib_cells -of_object[get_cells \
               data0_reg[3]]] threshold_voltage_group HVT

pt_shell> set_user_attribute [get_lib_cells -of_object[get_cells U25]] \
               threshold_voltage_group SVT
```

When you run the `report_threshold_voltage_group -pattern_priority` command, the threshold voltage groups are identified based on user-defined string patterns. This method is compatible with the PrimeTime tool for ECO cell swapping for leakage recovery.

For more information about ECO cell swapping, see the *PrimeTime User Guide*.

By default, user-defined pattern strings correspond to strings in the cell name. For example, a library cell might be named HVT_BUF1X, where HVT represents the voltage threshold group. When you specify the `-pattern_priority` option with the `-attribute` option, the user-defined string corresponds to a string in the attribute value.

Reporting Cell Count Based on Threshold Voltage Groups

The `report_threshold_voltage_group` command reports the number and percentage of the cells in each threshold voltage group and per leakage power group.

By default, the `report_threshold_voltage_group` command reports the cells per voltage threshold group, based on the cell or library threshold voltage group attributes. With the `-lvth_groups` option, you can specify the threshold voltage groups that are considered low-threshold.

Example 9-7 shows the voltage threshold distribution report generated by the `report_threshold_voltage_group` command with the `-lvth_groups` option to identify the low threshold voltage groups. The groups are assigned the attribute (L).

Example 9-7 Report Showing Cell Distribution Based on Threshold Voltage Groups

```
pt_shell> report_threshold_voltage_group -lvth_groups {LVT SVT}
-------------------------------------------------------------
Threshold Voltage Group Report
-------------------------------------------------------------
Attributes:
-------------
          L: user-defined low Vth group
-------------------------------------------------------------
Power Group         LVT(L)       SVT         HVT(L)
Name                cell(%)      cell(%)     cell(%)     Attribute
-------------------------------------------------------------
io_pad              0(  0.00%)   0(0.00%)    0( 0.00%)
memory              0(  0.00%)   0(0.00%)    0( 0.00%)
clock_network       0(  0.00%)   0(0.00%)    0( 0.00%)
black_box           0(  0.00%)   0(0.00%)    0( 0.00%)
```

```
register            4(100.00%)    0(0.00%)    0( 0.00%)
combinational      12( 40.00%)   10(33.33%)   8(26.67%)
sequential          0(  0.00%)   4(100.00%)   0( 0.00%)
-------------------------------------------------------------------------
Total              16(42.11%)   14(36.84%)    8(21.05%)
-------------------------------------------------------------------------
SUMMARY:
          Low Vth cell (%) = 24(63.16%)
         Other Vth cell(%) = 14(36.84%)
     Undefined Vth cell(%) =  0( 0.00%)
     -----------------------------------
          Total Cell (%)   = 38(100.00%)
     -----------------------------------
-------------------------------------------------------------------------
```

When you use the `-pattern_priority` option of the `report_threshold_voltage_group` command, specify the pattern strings in the reference cell name to indicate the threshold voltage group to which the cells belong. Additionally, the `-lvth_groups` option can be used to identify the low threshold voltage groups.

Example 9-8 shows the threshold voltage group distribution based on the reference name string pattern. The `-lvth_groups` option indicates the low threshold voltage groups.

Example 9-8 Report Showing Cell Distribution Based on Threshold Voltage Groups Identified by the Cell Name Pattern Priority

```
pt_shell> report_threshold_voltage_group -pattern_priority {HVT LVT NVT SVT} \
          -lvth_groups {LVT NVT}
-------------------------------------------------------------------------
               Threshold Voltage Group Report
-------------------------------------------------------------------------
Attributes:
         u -> user-defined power group
         L -> user-defined low Vth group

Power Group          HVT        LVT(L)     NVT(L)     SVT
Name                 cell(%)    cell(%)    cell(%)    cell(%)        Attribute
--------------------------------------------------------------------------------
io_pad               0 (0.00%)  0 (0.00%)  0 (0.00%)   0 (0.00%)
memory               0 (0.00%)  0 (0.00%)  0 (0.00%)   0 (0.00%)
clock_network        0 (0.00%)  0 (0.00%)) 0 (0.00%)   0 (0.00%)
black_box            0 (0.00%)  0 (0.00%)  0 (0.00%)   0 (0.00%)
register             0 (0.00%)  0 (0.00%)  0 (0.00%)   3 (75.00%))
combinational        1 (3.33%) 13 (43.33%) 0 (0.00%)  16 (53.33%)
sequential           0 (0.00%)  4 (0.00%)  0 (0.00%)   0 (0.00%)
--------------------------------------------------------------------------------
Total                1 (2.63%) 17(44.74%)  1 (2.63%)  19 (50.00%)
--------------------------------------------------------------------------------
SUMMARY :
               Low Vth cell (%)        = 18 (47.37%)
               Other Vth cells (%)     = 20 (52.63%)
               Undefined Vth cells (%) =  0 (0.00%)
----------------------------------------------------------------------
               Total cells (%)         = 38 (100.00%)
----------------------------------------------------------------------
```

Reporting Leakage Power Based on Threshold Voltage Group

To report the leakage power per threshold voltage group, specify the `-threshold_voltage_group` option of the `report_power` command. The threshold voltage groups, by default, again are identified by the cell and library attributes. By using the `-pattern_priority` option, you can classify the threshold voltage groups based on the specified reference name strings.

You can filter the leakage power data using the `-clocks`, `-groups`, and `-rails` options. If you specify the `-lvth_groups` option, the tool reports separate data for low threshold voltage groups, as shown in Example 9-9.

Example 9-9 Report Generated by the report_power -threshold_voltage_group -lvth_groups Command

```
pt_shell> report_power -threshold_voltage_group -lvth_groups {LVT SVT}
-------------------------------------------------------------------------------------
Power Group   HVT              LVT(L)           SVT(L)              DEFAULT_VTH_GROUP
Name          leakage(%)       leakage(%)       leakage(%)          leakage(%)    Attr
-------------------------------------------------------------------------------------
memory        0.0000(0.00%)    0.0000(0.00%)    0.0000(0.00%)       0.0000(0.00%)
io_pad        0.0000(0.00%)    0.0000(0.00%)    0.0000(0.00%)       0.0000(0.00%)
black_box     0.0000(0.00%)    0.0000(0.00%)    0.0000(0.00%)       0.0000(0.00%)
clock_network 0.0000(0.00%)    0.0000(0.00%)    4.398e-11(100.00%)  0.0000(0.00%)
combinational 0.0000(0.00%)    1.084e-07(59.03%)1.913e-09(1.04%)    7.333e-08(39.93%)
register      4.084e-08(61.32%)0.0000(0.00%)    0.0000(0.00%)       2.576e-08(38.68%)
sequential    0.0000(0.00%)    0.0000(0.00%)    0.0000(0.00%)       9.163e-09(100.00%)
-------------------------------------------------------------------------------------
Total         4.084e-08(61.32%)1.084e-07(59.03%)1.957e-09(0.75%)    1.083e-07(41.72%)
-------------------------------------------------------------------------------------
SUMMARY:
          Low Vth leakage (%) = 1.104e-07(42.53%)
         Other Vth leakage(%) = 4.084e-08(15.74%)
     Undefined Vth leakage(%) = 1.083e-07(41.72%)
     -------------------------------------------
           Total Leakage (%)  = 2.594e-07(100.00%)
     -------------------------------------------
-------------------------------------------------------------------------
```

Generating Reports on Clock-Gating Efficiency

Use the `report_clock_gate_savings` command to generate reports that contain metrics for measuring the clock-gating efficiency (CGE) in the design. You can use these reports to locate inefficient clock gates and to identify sequential cells with further clock gating possibility. You can use this command for both averaged and time-based power analysis modes.

Annotate the switching activity to the design before using the `report_power` command, because the efficiency metrics are based on toggle rates. 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 activity using the `read_saif`, `read_vcd`, or `set_switching_activity` commands. If the design is not fully annotated, the `report_clock_gate_savings` command initiates power analysis to propagate switching activity.

The accuracy of the analysis reported by this command on the clock gating depends on the quality of the switching activity that you apply on the design. The switching activity must represent the typical behavior of the design, which covers all modes of operation of the design. Simulation vectors are a source of quality switching activity that can be used for power analysis.

Generating the Default Report

Clock-gate toggle-savings is the basic metric for measuring clock-gating efficiency. It calculates the ratio of input clock toggles to the clock gates that are not propagated to the clock gate output, as shown in the following equation:

$$\text{ToggleSavings} = \frac{(1 - \text{Tr}_{\text{clkout}})}{\text{Tr}_{\text{clkin}}}$$

Clock gates with high toggle savings are those that disable a large percentage of input clock toggles from propagating to the output. The default report generated by the `report_clock_gate_savings` command provides with an overview of the clock-gating efficiency of the design as shown in Example 9-10.

Example 9-10 Report Generated by the report_clock_gate_savings Command

```
pt_shell> report_clock_gate_savings
****************************************
Report : Clock Gate Savings
power_mode: Averaged
****************************************
-----------------------------------------------------------------------
Clock: CLK
+ Clock Toggle Rate: 0.494681
+ Total number of Registers: 12
     + Number of Ungated Register :0 (0%)
     + Number of Gated Register    :12 (100%)
+ Number of Clock Gates: 3
+ Max number of Clock Stages: 2
+ Average Clock Toggle Rate at Registers: 0.0735657
+ Average Register Gating Efficiency (savings with respect to root
+clock): 85.1%
-----------------------------------------------------------------------
Toggle Savings                       Number of       % of
Distribution                         Registers       Registers
-------------------------------------------------------------------------
100%                                 0               0.0%
80% - 100%                           12              100.0%
```

```
60% - 80%                             0                0.0%
40% - 60%                             0                0.0%
20% - 40%                             0                0.0%
0% - 20%                              0                0.0%
0%                                    0                0.0%
-----------------------------------------------------------------------
1
```

The report includes a histogram that shows the average clock-gate toggle savings per clock domain.

For each clock, the report also includes the percentage of gated versus ungated registers. If most registers are ungated, you can continue to perform clock gating and power savings. The maximum number of clock gate stages identifies multistage gating. The average register-gating efficiency, is a further indicator of the clock-gating efficiency.

Register efficiency is the metric that measures the ratio of the register clock toggle rate to the root clock, as shown in the following equation:

$$\mathrm{RegisterEfficiency} = \frac{(1 - \mathrm{Tr}_{\mathrm{regclk}})}{\mathrm{Tr}_{\mathrm{rootclk}}}$$

The lower the frequency of the register clock, the higher the register efficiency. The register efficiency takes into consideration the toggle savings throughout the multistage clock gating.

Generating the Clock Gating Report

To view the contribution of each block to the total toggle savings, use the `-by_clock_gate` option with the `report_clock_gate_savings` command. When you specify the option, the clock-gating stage is identified, and the clock-gating efficiency per clock gate is displayed in the generated report.

The clock-gating stage for a register or clock stage represents the number of clock gates on the clock path, starting with the register clock pin. The rules to assign the clock stage for a clock gate or a register are:

1. If there is no clock gate driving the register, the clock stage is 0.
2. If there is a clock-gate driving the register clock pin, and the clock gate does not drive additional clock gates, the clock stage is 1.

The metric for analyzing the toggle savings contribution per clock gate is the Clock Gating Efficiency metric. This metric takes into consideration the number of registers driven by the clock gate, as well as the toggle savings, to rate the clock gate efficiency per clock domain.

The equation for computing the clock-gating efficiency or CGE is:

$$\text{ClockGatingEfficiency}_G = \left(\frac{N_G}{N}\right) * TS_G * \left(\frac{IN_G}{ROOT_G}\right)$$

where,

N = Total number of registers of the clock tree

NG = Number of registers in fanout of clock gate G including transitive fanout

ING = Toggle rate at input clock of clock gate G

ROOTG = Toggle rate at the root clock of clock gate G

TSG = Toggle Savings of clock gate G = (1 - OUTG/ING)

The first term indicates the normalization over all the registers of the clock domain. The more registers a clock gate drives, the greater is its contribution to toggle savings. The second term indicates the toggle savings due to clock gate G. The third term indicates that the saving is applicable to only the portion of the clock toggles that reach the clock gate G.

The CGE for high-stage clock gates is expected to be greater than that of the lower stage clock gates, because their toggle savings contributes to reduced toggles on all the registers driven by all the clock gates they drive, as shown in Example 9-11:

Example 9-11 Report Generated by the report_clock_gate_savings -by_clock_gate Command

```
pt_shell> report_clock_gate_savings -by_clock_gate
**************************************************************************
Report: report_clock_gate_savings
        -by_clock_gate
...
**************************************************************************
--------------------------------------------------------------------------
Clock Gate Structure Summary
--------------------------------------------------------------------------
Clock       Total       CG Stage   # of Clock    # of Gated  Clock Gating
            Registers               Gates         Cells      Efficiency
--------------------------------------------------------------------------
CLK         12           0          0              0         N/A
                         1          2              8         30.61%
                         2          1              4         54.52%
--------------------------------------------------------------------------
--------------------------------------------------------------------------
```

```
Clock Gate Structure Details
-------------------------------------------------------------------------
Clock Clock  Gating     # of   # of   # of      Toggle  Clock     Receiver
      Gate   Element    Fanout Clock Registers Savings Gating     Clock
      Stage                    Gates           (%)     Efficiency Gates
-------------------------------------------------------------------------
CLK   1     clk_gate_G1 4        0         4      81.8  27.3%
      1     clk_gate_G3 4        0         4      55.2   3.4%
      2     clk_gate_G2 5        1         4      81.8  54.5%   clk_gate_G3
-------------------------------------------------------------------------
```

As shown in Example 9-11, Clock Gate Summary identifies the clock gating stages, number of clock gates per stage, number of cells driven by these clock gates, and the average CGE of each stage. The Clock Gate Structure Details reports the fanout, toggle savings, clock stage, and CGE for each clock-gate cell.

For more information about computation of clock-gating efficiency, see "Clock-Gating Efficiency Computation" in Appendix B.

Comparing Clock-Gating Efficiency of Different Clock Domains

The power consumption of each clock domain is a function of the clock toggle rate in that domain. The CGE is a normalized value with respect to the clock toggle rate and the number of registers in the domain. To compare the CGE values between two clock domains multiply both the clock frequency and the number of registers in that clock domain before comparing.

Generating the Sequential Cell Report

To identify registers that can be further gated, use the `-sequential` option with the `report_clock_gate_savings` command. The generated report includes the Q-to-clock ratio:

$$\mathrm{Q-to-clockratio} = \frac{\mathrm{Tr}_{\mathrm{Q}}}{\mathrm{Tr}_{\mathrm{clk}}}$$

Registers with a low Q-to-clock ratio might be the candidates for XOR self-gating. If the data rarely changes, there is no need to continually clock the register. For register banks with low Q-to-clock ratios, power savings might be possible by using the XOR of the data to enable the clock gate. The clock is enabled only when the data changes. Example 9-12 shows a report example:

Example 9-12 Report Generated by the report_clock_gate_savings -sequential Command

```
pt_shell> report_clock_gate_savings –sequential
****************************************
Report : Clock Gate Savings
           power_mode: Averaged
           -sequential
...
****************************************
----------------------------------------------------------------------
Register  Clock    RootClock  LocalClock Q        Q/clk        Register
          Gating    Toggle    Toggle      Toggle   Toggle        Gating
          Stage     Rate      Rate        Rate     Ratio     Efficiency
----------------------------------------------------------------------
R2_reg_1_  2     0.4947     0.09016     0.04787   53.1%         81.8%
R2_reg_3_  2     0.4947     0.09016     0.1011   112.1%         81.8%
R2_reg_1_  1     0.4947     0.09013     0.02128   23.6%         81.8%
R1_reg_3_  1     0.4947     0.09013     0.04787   53.1%         81.8%
Q_reg_2_   1     0.4947     0.0404      0.02128   52.7%         91.8%
Q_reg_1_   1     0.4947     0.0404      0.02128   52.7%         91.8%
R2_reg_0_  2     0.4947     0.09016     0.05319   59.0%         81.8%
R1_reg_0_  1     0.4947     0.09013     0.1011   112.1%         81.8%
R2_reg_2_  2     0.4947     0.09016     0.1011   112.1%         81.8%
R1_reg_2_  1     0.4947     0.09013     0.04787   53.1%         81.8%
Q_reg_3_   1     0.4947     0.0404      0.04255  105.3%         91.8%
Q_reg_0_   1     0.4947     0.0404      0           0.0%         91.8%
----------------------------------------------------------------------
```

Additional Reporting Options

The `-sequential` option can be used with the `-by_clock_gate` option to report the 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 can be repartitioned, to use separate clock gates. For instance, if some registers in the cluster have different Q-to-clock 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 report generated using the `report_clock_gate_savings` command with the `-by_clock_gate` and `-sequential` options includes the clock-gating stages of the respective clock gates, as shown in Example 9-13.

Example 9-13 Report Generated by the report_clock_gate_savings -sequential -by_clock_gate Command

```
pt_shell> report_clock_gate_savings –sequential –by_clock_gate
****************************************
Report : Clock Gate Savings
power_mode: Averaged
Design : test
****************************************
---------------------------------------------------------------------------------
Clock Gate           Clock Gating    IN clk       OUT clk          Toggle
                     Stage           Toggle Rate  Toggle Rate     Savings (%)
---------------------------------------------------------------------------------
   Register
Q             Q/Clk

Toggle        Toggle

Rate          Ratio
---------------------------------------------------------------------------------
CG_CLK2_20/latch     1                0.1          0.02472          75.3%
   d0_r_reg[0]
0.00298462    12.1%
   d0_r_reg[1]
0.00282796    11.4%
CG_CLK1_10/latch     1                0.2          0.04883          75.6%
   d0_r_reg[2]
0.00385335    7.9%
   d0_r_reg[3]
0.00357056    7.3%
   d0_r_reg[4]
0.00380249    7.8%
   d0_r_reg[5]
0.00379639    7.8%
   d0_r_reg[6]
0.00373332    7.6%
   d0_r_reg[7]
0.00374349    7.7%
**Ungated**          0                NA           NA               NA
   d1_r_reg[1]
0.00302734    3.0%
   d1_r_reg[2]
0.00285645    2.9%
   d1_r_reg[0]
0.00367228    1.8%
---------------------------------------------------------------------------------
```

To isolate specific parts in the design to analyze different parts of the design so that you can use the `-hierarchical` option in combination with the `-by_clock_gate` or `-sequential` option.

To report the design related information and control the format of the report, use the following options:

- Design-related options

  - `-clocks`: Reports the objects in the specified clock domains.
  - *`object_list`*: Reports the hierarchical blocks or individual registers or clock-gating cells in the design.
- Reporting format-related options
  - `-sort_by`: Sorts the order of objects in the report.
  - `-nosplit`: Does not cause splitting the report over multiple lines.

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

Generating Reports on Power Derating Factors

To report the power derating factors set on rails or PG pins, use the `report_power_derate` command.

If the derating value is applied to a rail, the command lists the rail as an object type to which the value is applied when set at the design level or on hierarchical cells. If the value is applied to a leaf cell, the object type is the PG pin to which the rail is connected.

Example 9-14 shows the power derating factor set on various objects and power components:

Example 9-14 Report Generated by the report_power_derate Command

```
Report : power derate
Design : top
...
**************************************************************
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                     --         --       --
--------------------------------------------------------------
```

Generating Custom Reports with Attributes

The PrimeTime PX tool provides attributes that reference power information within the design and the libraries. Table 9-2 shows each attribute and the its object type. These attributes are accessible only after you run the `update_power` command.

Table 9-2 Object Types for the Supported Attributes

Class	Attribute
net	`switching_power`
	`glitch_rate`
	`toggle_rate`
	`toggle_count`
	`static_probability`
	`is_power_control_signal_net`
	`power_base_clock`
	`activity_source`
	`internal_power_derate_factor`
	`switch_power_derate_factor`
	`leakage_power_derate_factor`
pin	`glitch_rate`
	`toggle_rate`
	`static_probability`
	`power_base_clock (port also)`
	`internal_power_derate_factor`
	`switch_power_derate_factor`
	`leakage_power_derate_factor`
lib_cell	`has_multi_power_rails`
	`has_multi_ground_rails`
	`has_rail_specific_power_tables`
	`is_memory_cell`
	`is_pad_cell`
	`is_black_box`

Table 9-2 Object Types for the Supported Attributes (Continued)

Class	Attribute

cell	`dynamic_power`
	`glitch_power`
	`internal_power`
	`leakage_power`
	`peak_power`
	`switching_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`

	`internal_power_derate_factor`
	`switch_power_derate_factor`
	`leakage_power_derate_factor`

Table 9-2 Object Types for the Supported Attributes (Continued)

Class	Attribute
design	dynamic_power glitch_power internal_power leakage_power peak_power switching_power total_power x_transition_power
	power_states peak_power_start_time peak_power_end_time power_simulation_time
	intrinsic_leakage_power gate_leakage_power
	internal_power_derate_factor switch_power_derate_factor leakage_power_derate_factor

You can write your own procedures and use the object collections to access the desired information.

The script example shown in Example 9-15 generates a custom report for power consumption of all registers in the design:

Example 9-15 Script for Generating Custom Report for Power Consumption of All Registers

```
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]
  }
}
```

Generating Power Calculation Reports

When averaged power analysis is complete, run the `report_power_calculation` command to display the calculation of the internal power for a pin, the leakage power for a cell, or the switching power for a net. You can use this information for debugging or verifying power data in a logic library.

Note:
Before running the `report_power_calculation` command, you must read the logic library file into the system with the `library_features` attribute.

Averaged Power Analysis

In averaged power mode, use the `-state_condition`, `-rise`, `-fall` and `-path_sources` options to display the path-dependent internal power and the rise and fall internal power calculation.

Power-Net-Based Power Reports

To generate a power-net-based power report for multivoltage designs, use the `set_current_power_net` command to set the power net of interest before you run the `report_power_calculation` command. Only the power dissipated on the specified power nets is calculated and reported. By default, all the power nets are included in the power report.

To check the current power net setting, use the `get_current_power_net` command.

Gate-Level Power Analysis

Because all nets are annotated during gate-level power analysis, the `report_power_calculation` command re-analyzes only the selected cells to generate debugging data without overwriting the power data. You can access the original power data for additional debugging tasks when needed.

Example

The following example calculates the path-dependent internal power from the input pin B to the output pin Y. The contributions for the rise and fall of internal power are shown separately.

```
pt_shell> report_power_calculation [get_pin U4/Y] \
             -state_condition default -path_sources B

****************************************
Report : power_calculation
 U4/Y
 -state_condition default
 -path_sources B
Design : abc
Version: <Version>
Date   : <Date>
****************************************
Library(s) Used:
example (File: /root/libraries/example.db)
Operating Conditions: typical
Library: example
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
abc                   a                  example

Global Operating Voltage = 0.9
Library Power-specific unit information :
Voltage Unit : 1 V
Capacitance Unit : 1 pF
Time Unit : 1 ns
Dynamic Power Unit : 0.001 W    (derived from V,C,T units)
```

```
Leakage Power Unit : 1e-09 W
Note: Unless specified, the numbers reported are in library units.

======================================================================
Pin Internal Power Calculation
   cell: U4
   pin: Y
   path source: B

Path-Dependent Rise Pin Internal Power = 2.35214e-05 W
Pin Internal Power = Internal Energy * Pin Toggle Rate
Rise Internal Energy Per Transition = 0.143423

library model: NLPM
table variables:
    X = total_output_net_capacitance = 0.4
    Y = input_net_transition = 0.3
    relevant portion of lookup table:
                     (X)  0.1050      (X)  0.3550
    (Y)  0.0500      (Z)  0.1310      (Z)  0.1410
    (Y)  0.4510      (Z)  0.1320      (Z)  0.1420
     Z = A + B*X + C*Y + D*X*Y
     A =  0.1267           B =  0.0400
     C =  0.0025           D =  0.0000
     Z = 0.143423
(No scaling since the library has no internal power k-factors)

Path-Dependent Rise Pin Toggle Rate = 0.164
Path-Dependent Fall Pin Internal Power = 2.35214e-05 W
Pin Internal Power = Internal Energy * Pin Toggle Rate
Fall Internal Energy Per Transition = 0.143423
```

A

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
- Analysis Flow With the GUI
- Starting and Stopping a GUI Session
- Using the GUI

Starting and Ending a Shell Session

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

To start PrimeTime shell (pt_shell) 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 a license is available, the following message is displayed. The PrimeTime PX license is checked out when the first power-related command is executed.

```
                 PrimeTime (R)
          Version H-2013.06 -- June 10, 2013
     Copyright (c) 1988-2013 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>
```

Commands are entered at the pt_shell prompt (pt_shell>).

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 `start_gui` command at the pt_shell prompt. The `start_gui` command does not work when used in a script.

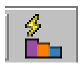
To stop the GUI while still keeping the original pt_shell session going in the terminal window, use the `stop_gui` command or choose File > Close GUI from the menu. To exit from PrimeTime PX entirely, choose File > Exit.

Using the GUI

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

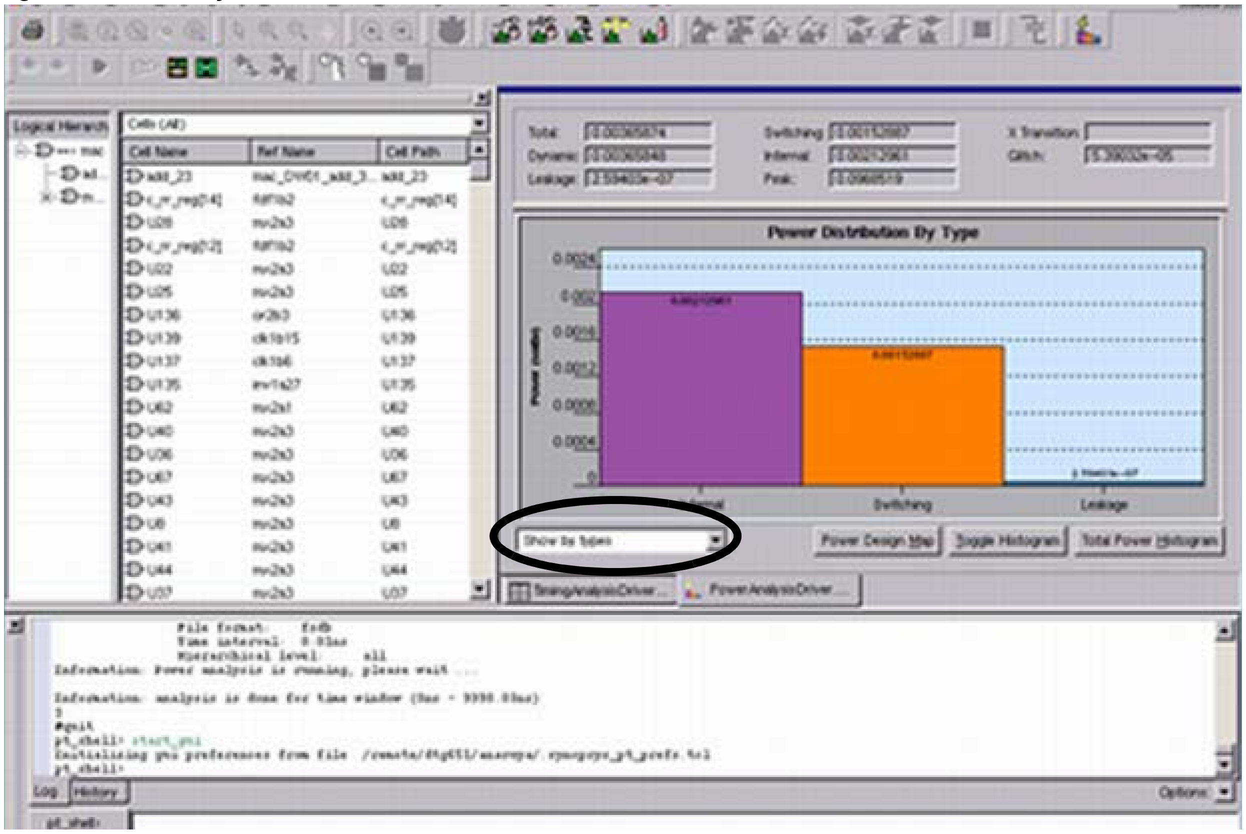
After you have read and calculated power and opened the GUI, you can view and analyze your power data by right-clicking within the main window and choosing Power > Analysis Driver or by clicking the power icon, which is as shown in Figure A-1.

Figure A-1 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 A-2, 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 A-2 Analysis Driver Window



Cell Name	Ref Name	Cell Path
add_23	mac_DW01_add_3...	add_23
c_rr_reg[14]	fdf1b2	c_rr_reg[14]
U28	mx2a3	U28
c_rr_reg[12]	fdf1b2	c_rr_reg[12]
U22	mx2a3	U22
U25	mx2a3	U25
U136	or2b3	U136
U139	clk1b15	U139
U137	clk1b6	U137
U135	inv1a27	U135
U62	mx2a1	U62
U40	mx2a3	U40
U06	mx2a3	U06
U67	mx2a3	U67
U43	mx2a3	U43
U8	mx2a3	U8
U41	mx2a3	U41
U44	mx2a3	U44
U37	mx2a3	U37

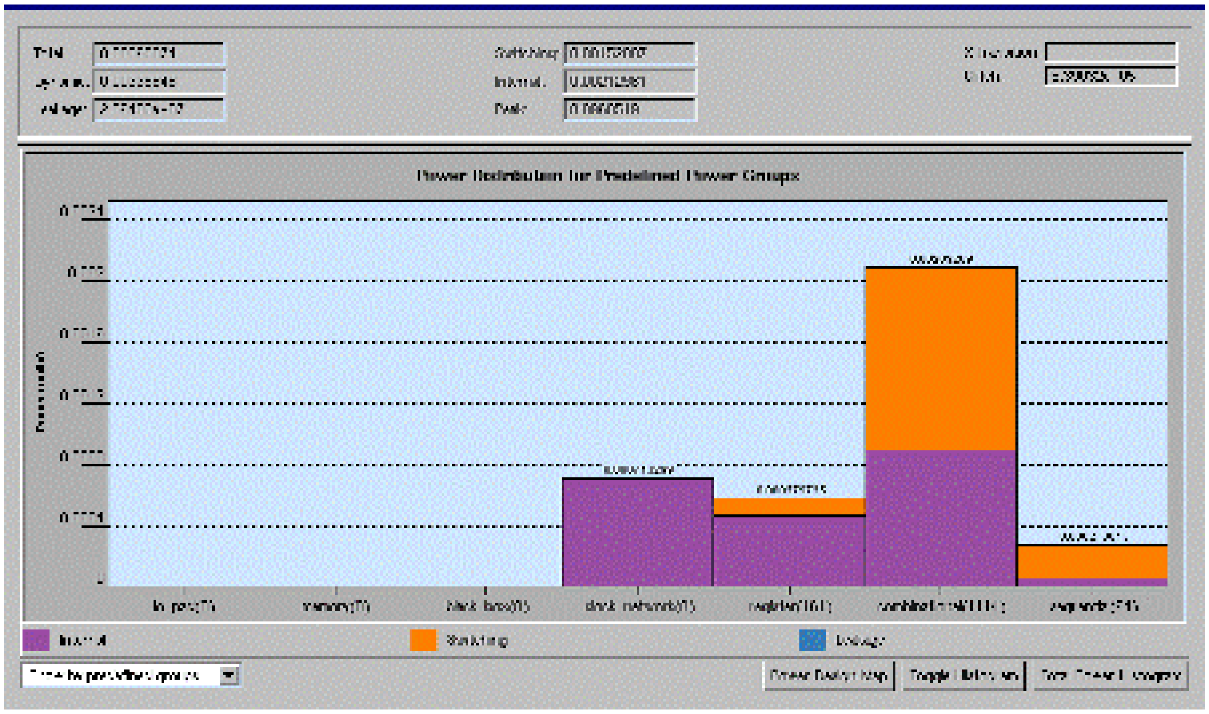
Total:	0.00365874	Switching:	0.00152687	X Transition:	
Dynamic:	0.00365848	Internal:	0.00212961	Glitch:	5.39032e-05
Leakage:	2.59403e-07	Peak:	0.0968519		

```
                 File format:     fsdb
                 Time interval:   0.01ns
                 Hierarchical level:      all
Information: Power analysis is running, please wait ...

Information: analysis is done for time window (0ns - 9999.99ns)
1
#quit
pt_shell> start_gui
Initializing gui preferences from file  /remote/[illegible]/.synopsys_pt_prefs.tcl
pt_shell>
```

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 A-3 shows the power distribution for predefined groups. The numbers inside parentheses indicate the number of cells in the associated groups.

Figure A-3 Power Distribution of Predefined Power Groups



Design Map. Click the Power Design Map button shown in Figure A-3 to view the total power density as a map of the design. The power density displays as a tree map, which is a visualization tool that can help you identify anomalies in large hierarchical data sets.

Tree maps 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.

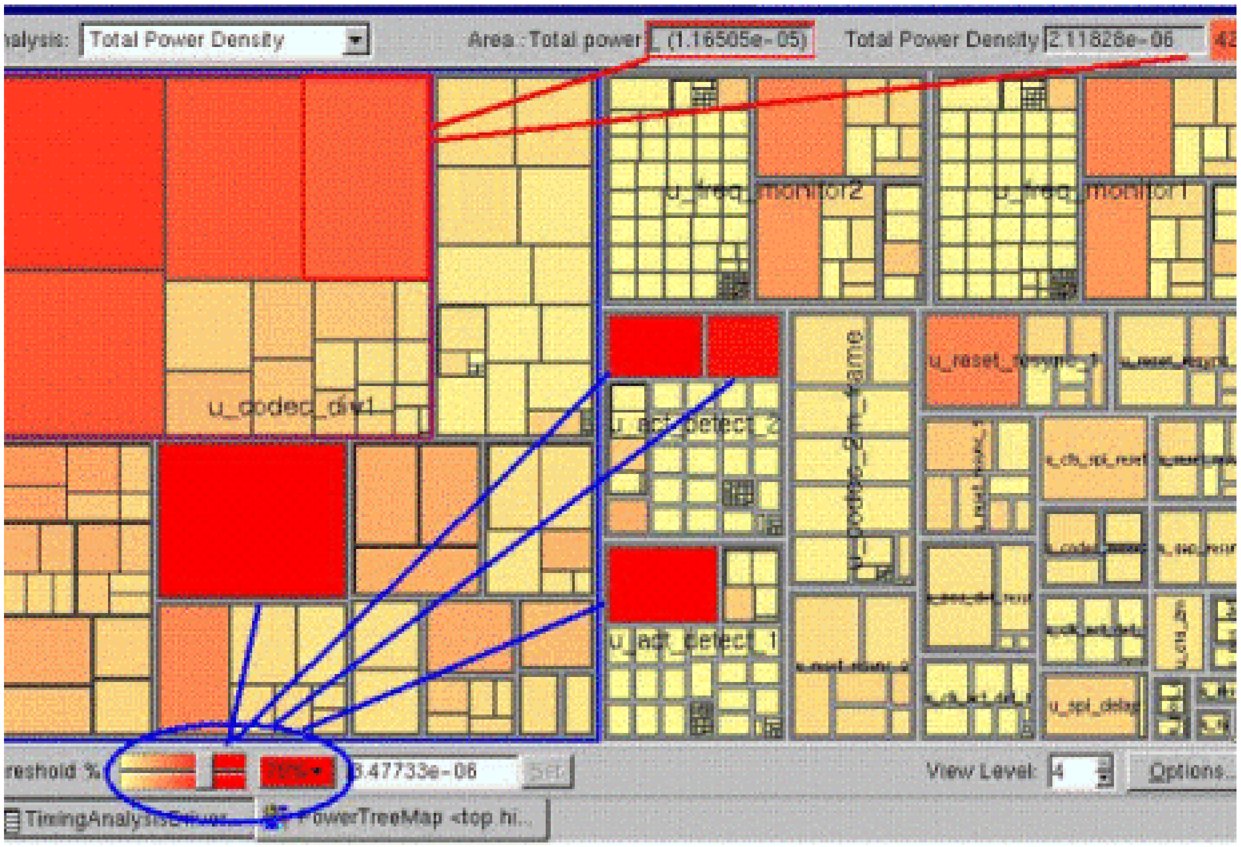
Use the Analysis box to change the parameters used for mapping how the color and node size indicators are used. Your choices are shown in Table A-1.

Table A-1 Tree Map 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 A-4 shows a tree map 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 A-4 Tree Map



Right-click within the tree map 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 A-4, 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 tree map.

Click the Customize button to view the Power Treemap Options dialog box from which you can choose to open new tree maps 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 A-2, 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 A-3, or for toggle rates, as shown in Figure A-6.

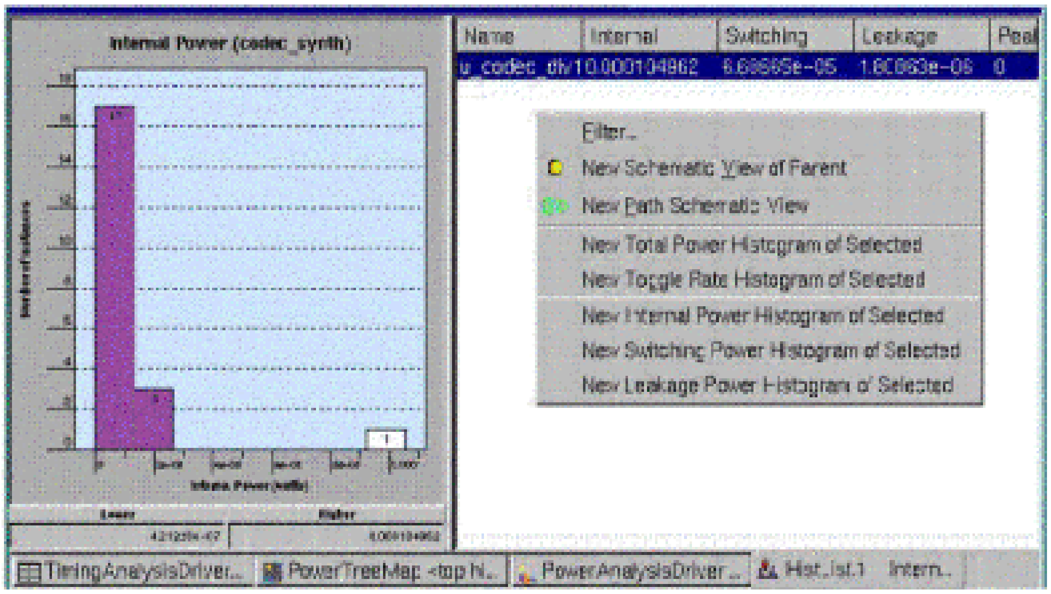
A context-sensitive menu (not shown) provides quick access to histogram data depending on which segment of the power distribution bar chart you select. For example, selecting the internal-power bar shown in Figure A-3 provides choices for viewing a histogram of the internal power as well as a total power 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 A-5, this menu allows you to generate a further series of histograms based on the selected bars.

Figure A-5 Histogram Context-Sensitive Menu

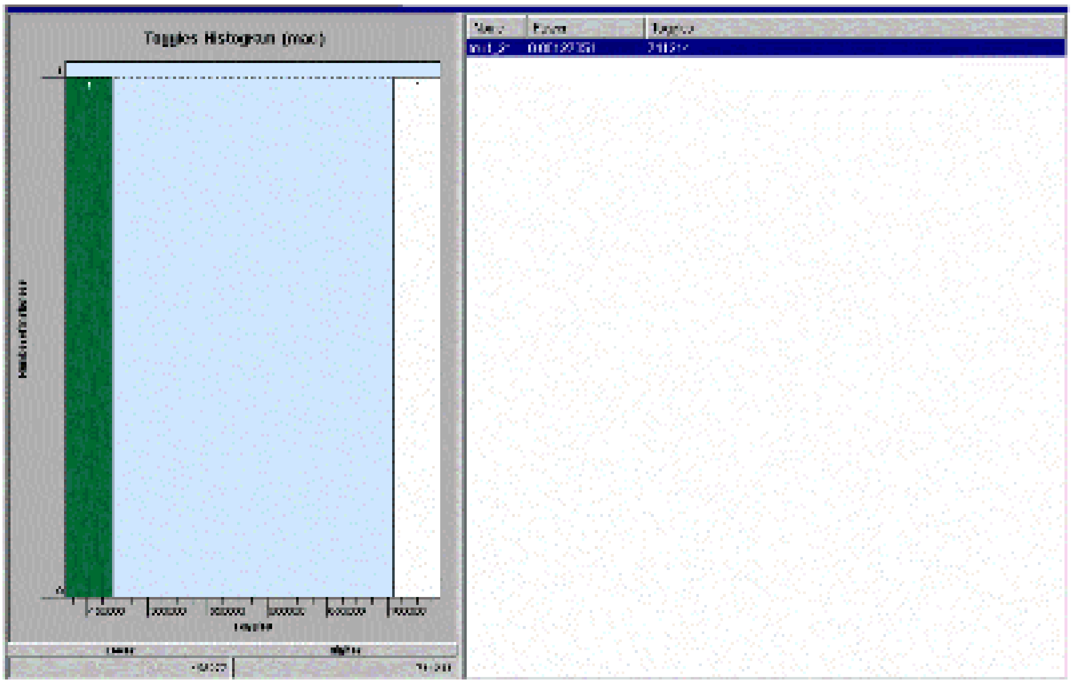

Internal Power (codec_synth)
Number of Instances
Internal Power (watts)
Lower
Higher
4.21226e-07
0.000104862
Filter...
New Schematic View of Parent
New Path Schematic View
New Total Power Histogram of Selected
New Toggle Rate Histogram of Selected
New Internal Power Histogram of Selected
New Switching Power Histogram of Selected
New Leakage Power Histogram of Selected
TimingAnalysisDriver...
PowerTreeMap <top hi...
PowerAnalysisDriver...
Hist..lst.1
Intern...

Name	Internal	Switching	Leakage	Peal
u_codec_div1	0.000104862	6.69665e-05	1.60963e-06	0

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 A-6 shows an example of a toggle rate histogram. Select multiple cells in the design hierarchy browser to view their histograms.

Figure A-6 Toggle Histogram



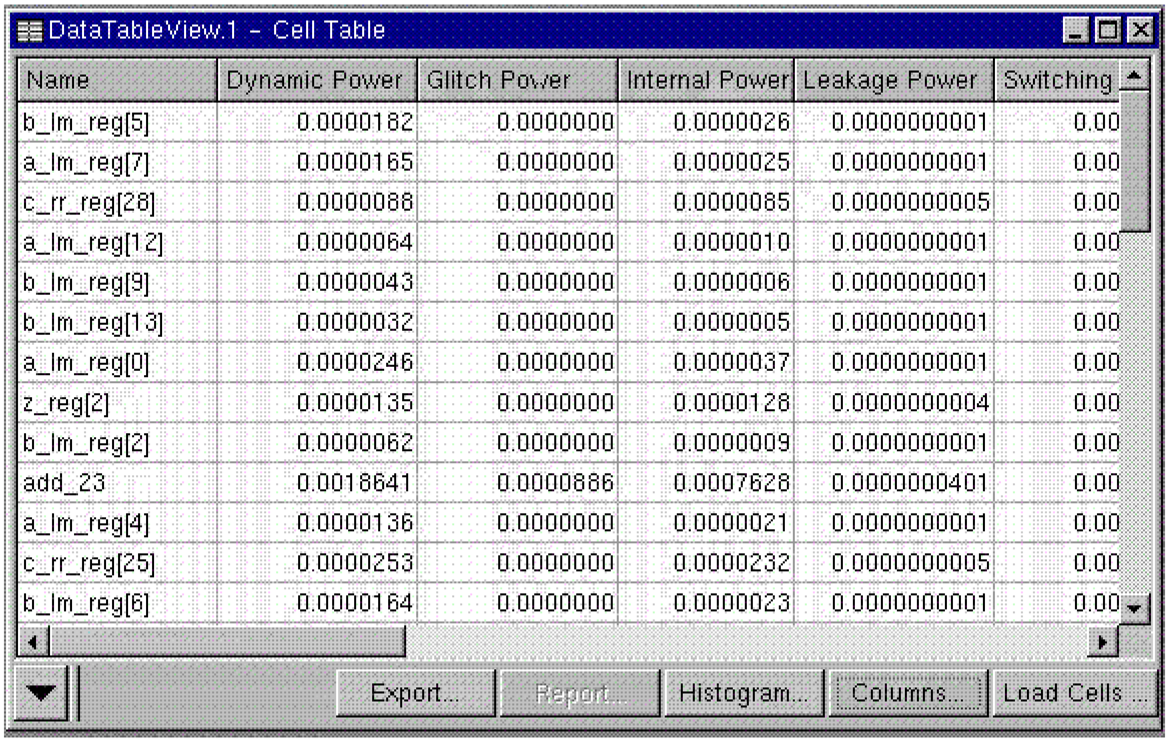
When you select a histogram bar, it turns yellow and its associated cells are displayed in the right pane. The specified multiple cells listed on the right (not shown), you can select any combination of them to view their histograms.

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 A-7 shows a cell data table.

By default, power attributes do not appear in the data tables. To view them as shown in Figure A-7, click the Columns button.

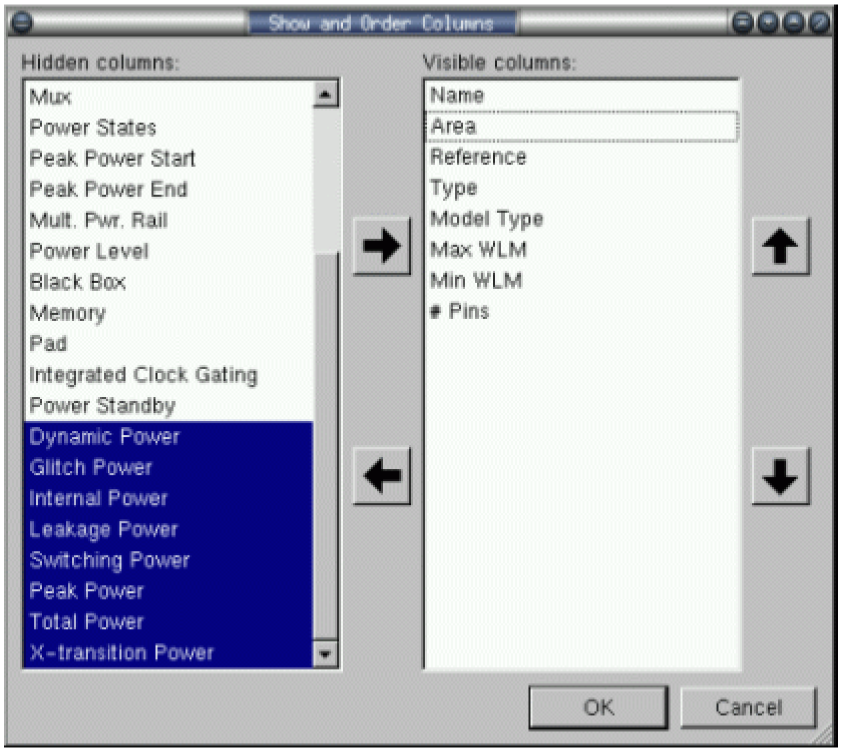
Figure A-7 Cell Data Table



Name	Dynamic Power	Glitch Power	Internal Power	Leakage Power	Switching
b_lm_reg[5]	0.0000182	0.0000000	0.0000026	0.0000000001	0.00
a_lm_reg[7]	0.0000165	0.0000000	0.0000025	0.0000000001	0.00
c_rr_reg[28]	0.0000088	0.0000000	0.0000085	0.0000000005	0.00
a_lm_reg[12]	0.0000064	0.0000000	0.0000010	0.0000000001	0.00
b_lm_reg[9]	0.0000043	0.0000000	0.0000006	0.0000000001	0.00
b_lm_reg[13]	0.0000032	0.0000000	0.0000005	0.0000000001	0.00
a_lm_reg[0]	0.0000246	0.0000000	0.0000037	0.0000000001	0.00
z_reg[2]	0.0000135	0.0000000	0.0000128	0.0000000004	0.00
b_lm_reg[2]	0.0000062	0.0000000	0.0000009	0.0000000001	0.00
add_23	0.0018641	0.0000886	0.0007628	0.0000000401	0.00
a_lm_reg[4]	0.0000136	0.0000000	0.0000021	0.0000000001	0.00
c_rr_reg[25]	0.0000253	0.0000000	0.0000232	0.0000000005	0.00
b_lm_reg[6]	0.0000164	0.0000000	0.0000023	0.0000000001	0.00

The Show and Order Columns dialog box opens, as shown in Figure A-8 where you can select the applicable power-related columns.

Figure A-8 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.

B

Clock-Gating Efficiency: Example and Computation

This appendix includes an example of multistage clock gating per clock gate stage, and the computation of clock-gating efficiency for this design, as described in the following section:

- Clock-Gating Efficiency Computation

Clock-Gating Efficiency Computation

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

The following equation shows how PrimeTime PX computes clock gating efficiency:

$$CGE_G = \left(\frac{4}{12}\right) * TS_{G3} * (1 - TS_{G2}) = 3.35\%$$

CGE represents the clock gating efficiency for a particular gating element only. To determine the CGE for the entire clock gate structure of a particular clock domain, sum up all the CGEs of that clock domain, as shown in the following formula:

For all the clock gates G in the clock domain for CLK,

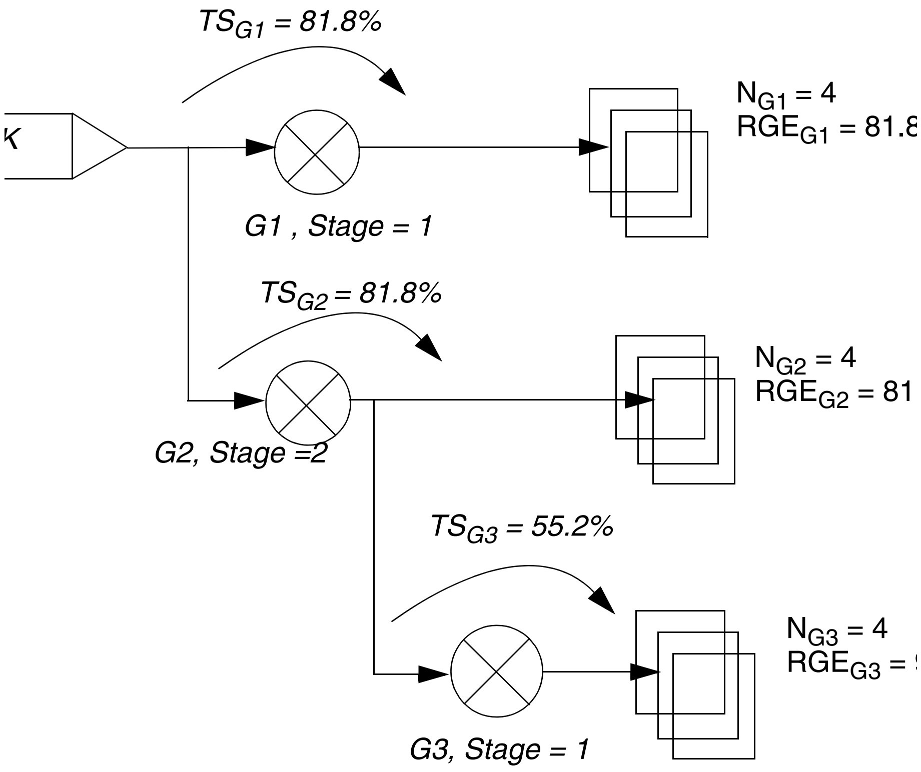
$$CGE_{CLK} = \sum CGE_G;$$

CGE for a clock gating stage S is the sum of all the CGEs of clock gates at stage S as follows:

$$CGE_S = \sum CGE_G;$$

Figure B-1 shows an example of a multistage clock gating per clock gate stage.

Figure B-1 The Clock Gating Example and How CGEs Are Computed



Continued from Figure B-1, the following examples the compute clock gating efficiency for G1, G2, and G2:

$$\mathrm{CGE}_{\mathrm{G1}} = \left(\frac{4}{12}\right) * 81.8\% = 27.3\%$$

$$\mathrm{CGE}_{\mathrm{G2}} = \left(\frac{8}{12}\right) * 81.8\% = 54.5\%$$

$$\mathrm{CGE}_{\mathrm{G3}} = \left(\frac{4}{12}\right) * 55.2\% * (1 - 81.8\%) = 3.35\%$$

Index