PrimeTime® PX

Recommended Methodology for Power Analysis

Version 1.3

For PrimeTime PX version 2006.12

SYNOPSYS®

Table of Contents

1 Introduction

As technology advances, design challenges for achieving a successfully performing chip as regards to reliability, signal integrity and power become increasingly complex.

Power management is one of the main design challenges for nanometer-scale technologies where timing, power, and signal integrity are interrelated. The total power consumption of a design is a critical parameter affecting factors such as packaging, cooling decisions, device battery life, and chip reliability. Power analysis is therefore crucial throughout the design flow (Figure 1) rather than simply at sign-off.

PrimeTime® PX (PTPX) is an accurate power analysis tool that includes timing interdependencies for power dissipation that you can use at various stages in the design flow as shown in Figure 1. It performs static timing analysis and signal integrity analysis.

Several enhancements were completed for the PTPX 2006.12 release. These cover areas such as run-time improvements, better activity annotation through name mapping, and support for power scaling between libraries. The following enhancements are available in the 2006.12 release:

- Run-time improvement up to 8X for gate-level VCD peak power analysis
- Improved RTL name mapping flow
- Clock tree estimation in total power report
- Scaling support for voltage and temperature
- Enhanced reporting for switching annotation
- Current waveform output

This white paper describes the recommended methodology for performing power analysis using PTPX.

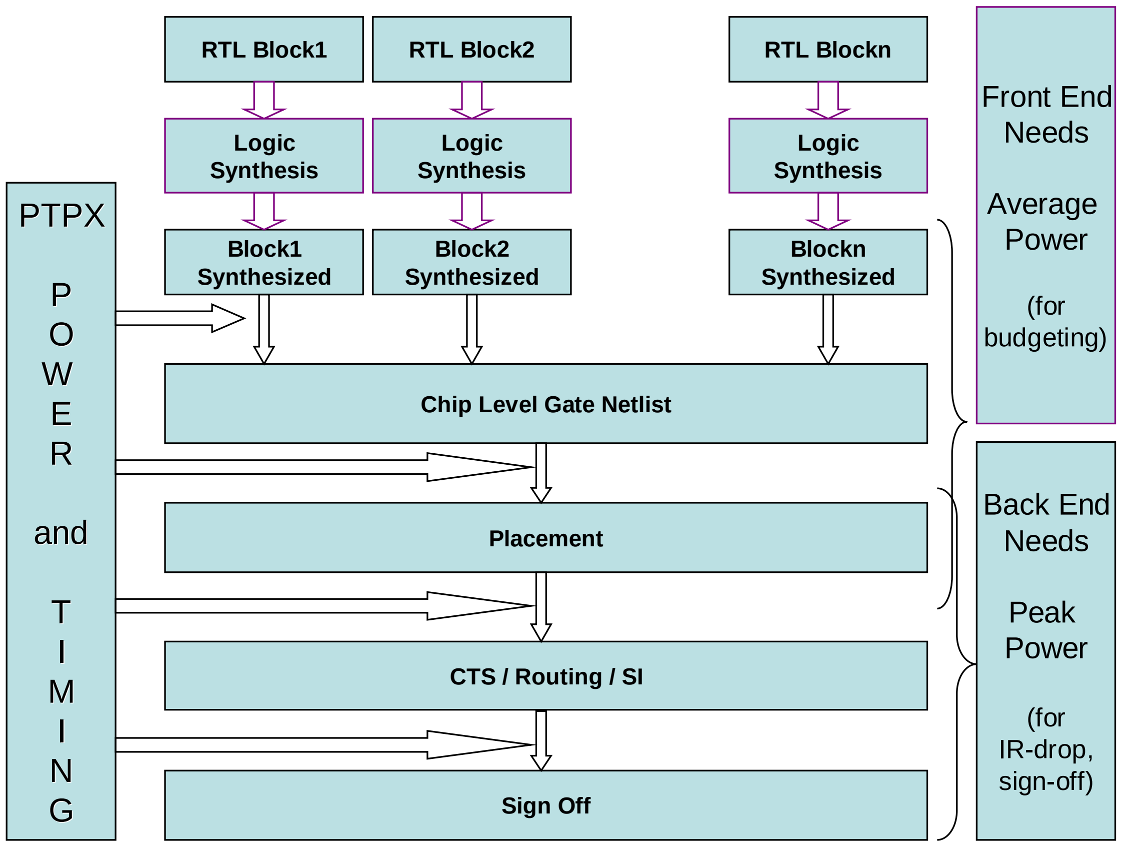


Figure 1: Power Analysis in the Design Flow

2 Power Basics

Power is the energy dissipated in a device per unit of time. The total power dissipated in a device consists of two components:

- Static or leakage power when the device is at steady state
- Dynamic power when the device is switching

$$P_{total} = P_{static} + P_{dynamic}$$

Static or leakage power is state, temperature, process and voltage dependent and is the product of the supply voltage (Vdd) and the leakage current (I_{leak}).

$$P_{static} = Vdd * I_{leak}$$

Dynamic power is comprised of internal power and switching power.

$$\mathbf{P_{dynamic} = P_{internal} + P_{switching}}$$

Energy ($E_{Ccharge}$) is consumed during charging a load capacitance (C_L) through a supply voltage (V).

$$\mathbf{E_{Ccharge} = C_L V^2}$$

No energy ($E_{Cdischarge}$) is consumed during discharging the load capacitance.

$$\mathbf{E_{Cdischarge} = 0}$$

Total energy consumed during charging and discharging the load is same as the total energy dissipated during charging and discharging. Energy (E_D) is dissipated during the charging ($E_{Dcharge}$) or discharging ($E_{Ddischarge}$) of the load capacitance.

$$\mathbf{E_{Dcharge} = \tfrac{1}{2} C_L V^2}$$

$$\mathbf{E_{Ddischarge} = \tfrac{1}{2} C_L V^2}$$

$$\begin{aligned}\textbf{Total energy dissipated} &= \mathbf{E_{Dcharge} + E_{Ddischarge}} \\ &= \mathbf{\tfrac{1}{2} C_L V^2 + \tfrac{1}{2} C_L V^2} \\ &= \mathbf{C_L V^2}\end{aligned}$$

Power is dissipated during charging and discharging of the load capacitance (C_L) at a frequency (f cycles per second) through a supply voltage (V).

$$\begin{aligned}\textbf{Power dissipated during charging and discharging} &= \mathbf{C_L V^2 f} \\ &= \mathbf{C_L V^2 (T_r / 2)}\end{aligned}$$

where toggle rate (T_r) is the rate of change of state transitions with respect to time; that is, the number of toggles per unit time.

Internal power is comprised of:

1) Short circuit or crow-bar power dissipated when the device inputs are changing from low to high or high to low when both P and N transistors are "ON" due to the momentary short circuit current (I_{sc})
2) Power dissipated due to charging and discharging of the internal load(C_{int}), which is

$$P_{internal} = P_{sc} + P_{intsw}$$

$$= (Vdd * I_{sc}) + (½ * C_{int} * Vdd^2 * T_r)$$

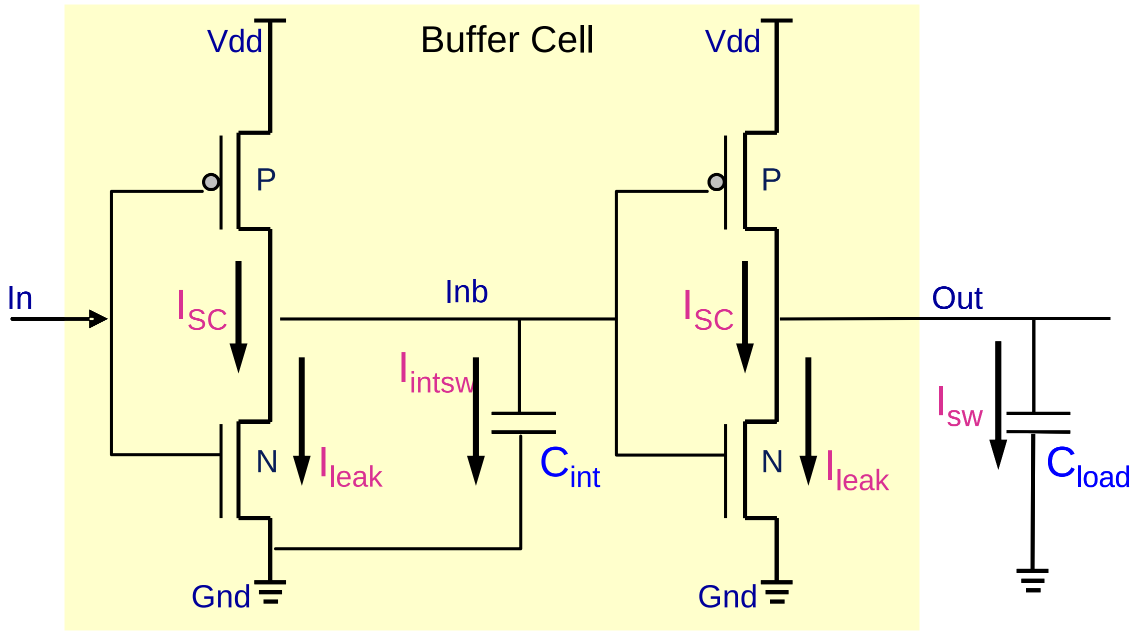
   where toggle rate (T_r) is the rate of change of state transitions with respect to time; that is, the number of toggles per unit time.

3) Switching power, which is the power dissipated by the charging and discharging of the load capacitance at the output of a cell, which is

$$P_{switching} = ½ * C_{load} * Vdd^2 * T_r$$

   where the total load capacitance (C_{load}) is the sum of the net and gate capacitances on the driving output, and T_r is the toggle rate .

Figure 2 shows the power components for a simple buffer cell.



I_{sc}	Short circuit current
I_{intsw}	Internal switching current
I_{leak}	Leakage current

Figure 2: Components of Power Dissipation

The leakage current can vary based on the states of the transistor. With the input signal in a high state, the leakage power when transistor N is on differs from that when transistor N is off. A rising signal applied at the input results in a dissipation of internal power due to currents Isc and Iintsw. Isc is seen from Vdd to Gnd due to the transition from low to high as the transistor N turns on and the transistor P turns off. Internal switching power is incurred during charging and discharging of C_{int}. Charging and discharging of C_{load} results in dissipation of switching power on Out net due to Isw. The dissipation of internal power of a cell depends on the input transition time and output load capacitance. It is pre-characterized and stored in the library for different output loads and input transition times.

3 Power Analysis flow

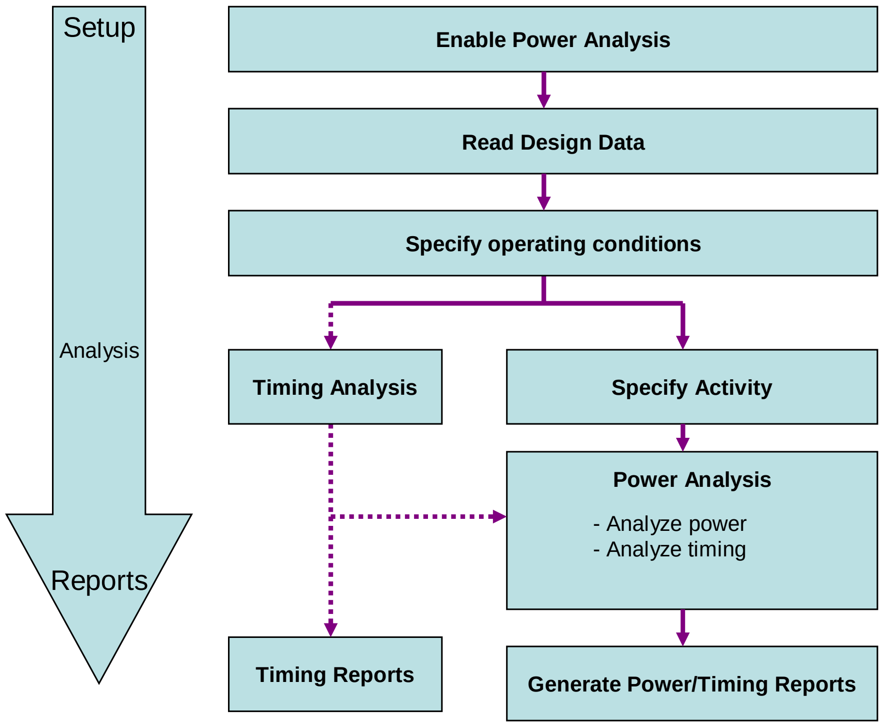


Figure 3: Power Analysis Flow in PTPX

The success of power analysis depends on adopting certain power analysis steps in a particular order. Figure 3 shows the steps adopted by PTPX.

The first step in PTPX is to enable power analysis, which is required for reading the power library and analyzing power in the flow. Next, read in the design data, including the technology library. Reading the design data also includes reading the constraints and annotating parasitics, which is required for power analysis. Afterwards, specify operating conditions such as process, temperature and voltage for power analysis, then specify the switching activity data. Finally, perform power analysis and generate reports.

During power analysis, PTPX first performs timing analysis if the timing is not updated and uses the transition time for calculating accurate power and timing window for calculating average cycle power waveforms. Timing analysis includes SI effects if SI is enabled. When the timing data is available for power analysis as shown by dotted lines in the flow diagram, PTPX uses the timing data for power calculation instead of recalculating them thereby saving the runtime.

3.1 Power Analysis Inputs

PTPX requires the following for power consumption analysis:

Technology library
Cell library containing timing and power characterization information for each cell. The quality of the library determines accuracy of the power results. By default, PTPX expects a CCS library and uses the CCS as priority when both CCS and NLPM power data are available in the library unless you specified the nlpm variable with **power_model_preference**

Gate-level netlist
Flat or hierarchical gate-level netlist in Verilog, VHDL or Synopsys database format containing leaf-level instantiation of library cells.

Design constraints
SDC file containing the design constraints to calculate the transition time on the primary inputs and to define the clocks.

Activity
Design switching activity information either for average power analysis or for accurate peak power analysis.

Net parasitics
Parasitics file (SPEF) containing net capacitance for all nets.

4 Power Analysis Techniques

In CMOS circuits, 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 sign off for IR drop analysis. Both of these power analysis techniques require design switching activity on every node. Activity is defined as how often a net switches with respect to a given clock period.

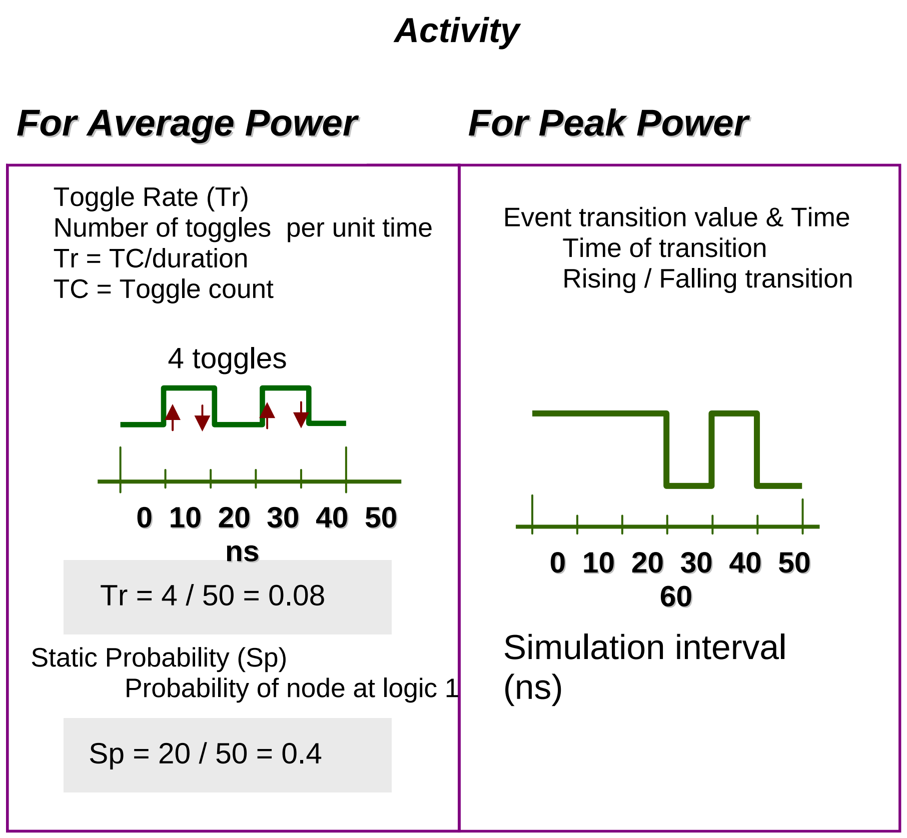


Figure 4: Activity for Average Power and Peak Power

PTPX performs both types of power analysis: average power analysis and peak power analysis. It calculates static and dynamic power for both average power analysis and peak power analysis.

5 Average Power Analysis

PTPX average power analysis supports analysis of the following activities: default switching, user-defined switching, switching from RTL or gate-level simulation in the form of VCD (Value Change Dump) or SAIF (Switching Activity Interchange Format). To perform the analysis, PTPX checks for annotation on all the nodes (Tr/Sp). If unavailable, PTPX assigns the default toggle rate and static probability to the primary inputs and black box outputs. Toggle rate and static probability are the key design activity parameters of average power analysis. Toggle rate is used in calculating the dynamic power.

PTPX also supports internal power for toggle rates on leaf cell input pins that are state-dependent (SD) and toggle rates on leaf cell output pins that are state-dependent and/or path dependent (SDPD). PTPX thus analyzes the total power consumption of each cell based on toggle rates, static probability and ultimately the total chip power, which is the sum of the entire leaf cell's power. It can generate average cycle power waveforms based on the common-base period of multiple clocks.

While PTPX supports annotation of gate-level simulation data, this is often not available. The primary flows are:

- Vector free (VF): No switching activity for quick early analysis
- RTL VCD: With switching activity from RTL simulation for more accurate average power analysis

5.1 Vector-Free Flow

You can use the vector-free flow (Figure 5) for quick power estimation early in the design flow when there is no simulation data available. In this flow, PTPX reads the design data, processes any defined switching activity and checks the activity on all ports/pins. It assigns default toggle rates and static probability on primary inputs and black-box outputs, and propagates the activity with zero-delay simulation.

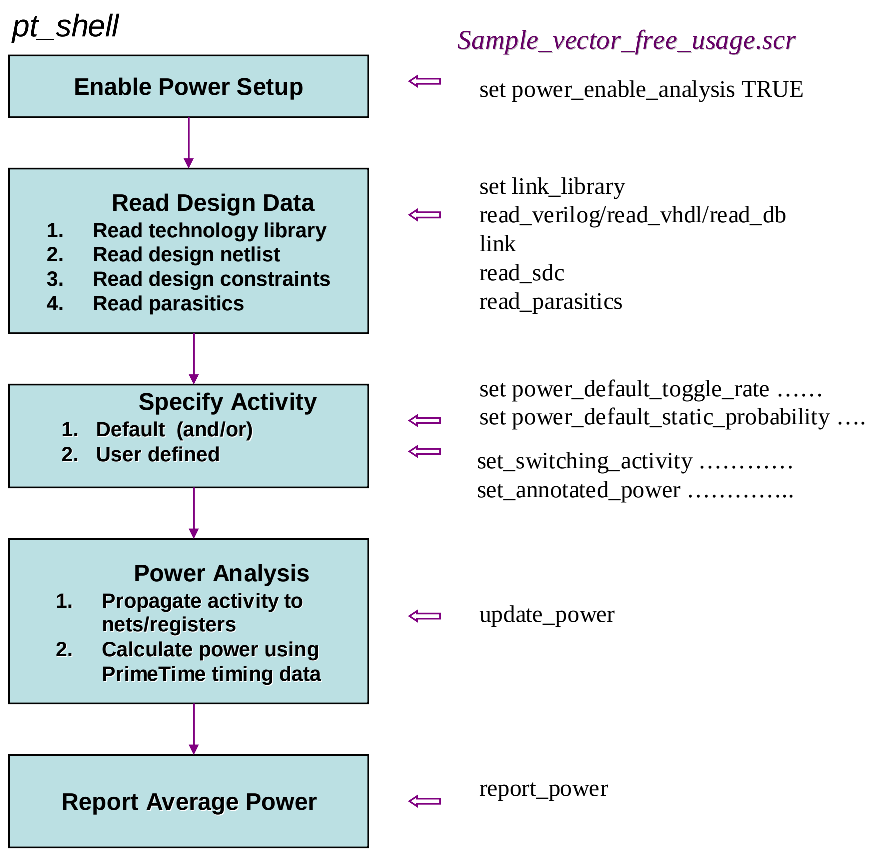

pt_shell
Sample_vector_free_usage.scr
Enable Power Setup
set power_enable_analysis TRUE
Read Design Data
1. Read technology library
2. Read design netlist
3. Read design constraints
4. Read parasitics
set link_library
read_verilog/read_vhdl/read_db
link
read_sdc
read_parasitics
Specify Activity
1. Default (and/or)
2. User defined
set power_default_toggle_rate ……
set power_default_static_probability ….
set_switching_activity …………
set_annotated_power …………..
Power Analysis
1. Propagate activity to nets/registers
2. Calculate power using PrimeTime timing data
update_power
Report Average Power
report_power

Figure 5: Vector-Free Flow for Average Power Analysis

The default toggle rate is a relative value based on the reference clock. When a design has clocks defined and if input transitions are provided relative to a clock, PTPX assumes the specified clock is the related clock. Otherwise, PTPX traces the port's fan-out until it reaches a clocked element. When more than one clock is found, PTPX uses the fastest of the related clocks. PTPX also uses the fastest design clock when no related clock is found. It is important to define the clock otherwise PTPX assumes a clock period of one library time unit. PTPX uses a default toggle rate and static probability of 0.5, which is extremely high. PTPX supports setting realistic values for these parameters for higher accuracy of results.

5.1.1 Guidelines for Setting Switching Activity

The following guidelines are recommended for annotating switching activity:

Purpose	Command
• Annotates a realistic default toggle rate value, for example, 0.3 to all the starting points. Recommended to provide a pessimistic value for power compared to the gate level VCD.	*set power_default_toggle_rate 0.3*
• Annotates reset signal by applying a constant value of 0. Recommended for special signals like reset and scan enable because default toggle rates should not be applied.	*set_case_analysis 0 reset*
• Annotates toggle rates based on the related clocks using the **-clocks** option to register outputs. Recommended because propagation through sequential elements can result in loss of activity.	*set_switching_activity –type registers –toggle_count <value> –clocks {all_clocks}*
• Annotates clock-gating cells with a factor of <*value*> times the toggle rate of the related clock. When **-clocks** is specified, the **-clock_derate** option is used as an alternative to **-toggle_count**. Guarantees that the clock-gating output is toggling at the expected rate.	*set_switching_activity –clocks clk2 –clock_derate <value> –type clock_gating_cells*

In this flow, the **update_power** command performs average power analysis while **create_power_waveforms** generates average cycle waveforms using timing windows and **report_power** reports average power results.

5.2 RTL VCD Flow

Using a RTL VCD file can provide better power results compared to the vector-free flow. This flow is depicted in Figure 6. PTPX reads the design data and by using the vcd2saif utility derives switching activity (SAIF) automatically from the VCD file. RTL VCD has partial design activity. PTPX annotates the activity from RTL VCD, and, for unannotated nets, propagates the activity with zero-delay simulation before calculating statistical average power.

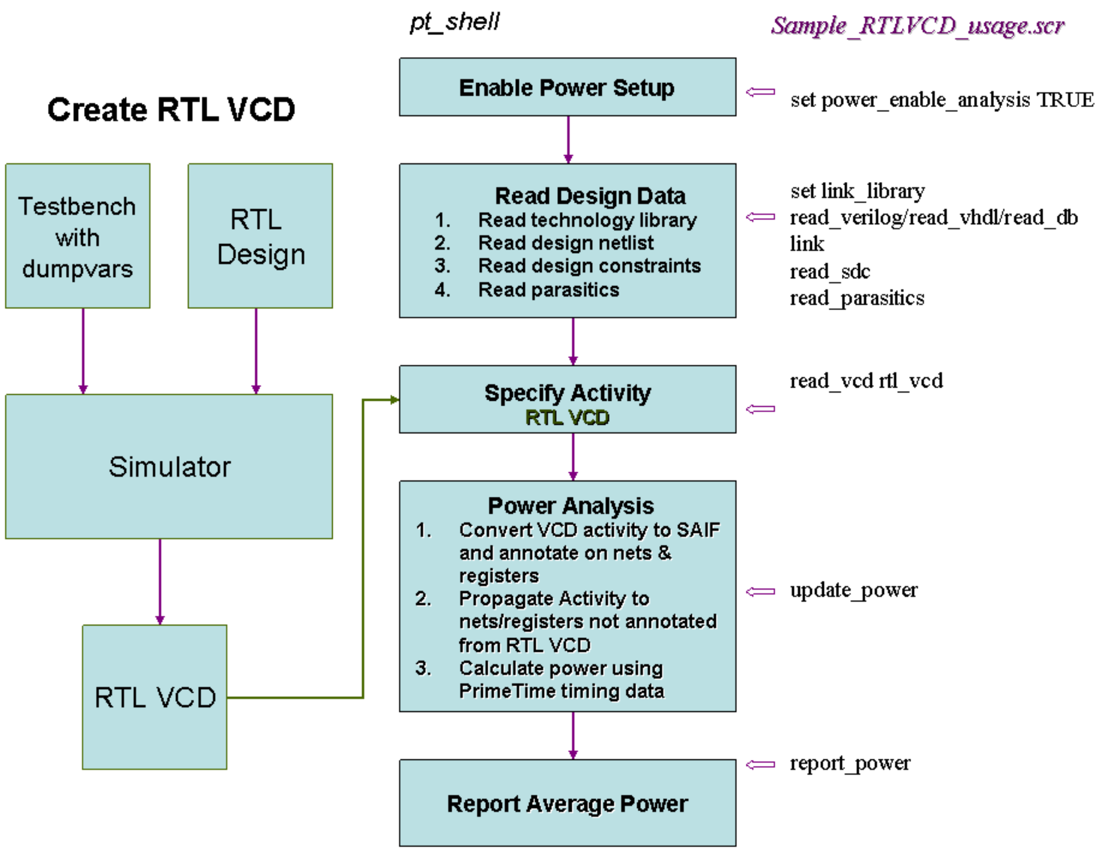


Figure 6: RTL VCD Flow for Average Power Analysis

Figure 6 shows the usage model for average power analysis with RTL VCD. In this flow, activity is specified using the **read_vcd** command.

Ex: read_vcd rtl_vcd.dump –strip_path tb/top_inst

This example specifies RTL VCD activity rtl_vcd.dump for top instance top_inst in a test bench tb.

In this flow, the **update_power** command performs average power analysis while **create_power_waveforms** generates average cycle waveforms using timing windows and **report_power** reports average power results. Unless you set **power_force_saif_flow** to true, **create_power_waveforms** can not be used.

5.2.1 Guidelines for RTL VCD flow

How to create a RTL VCD file:

Create a VCD file with $dumpvars in the testbench as shown in the following example using the Synopsys VCS-MX simulator or other HDL simulators.

Ex: Test bench command to create RTL VCD

```
begin
      $dumpfile “rtl_vcd.dump”
      $dumpvars;
 end
```

Use the simulation command:

Ex: VCS –R -I tb.v rtl.v

This example shows the simulation command for a verilog test bench tb.v and RTL verilog rtl.v

What is the minimum set of dumpvars required?

By default, dumpvars outputs the activity for all the registers, wires and variables in the netlist. It is important to capture register and IO activity for accurate analysis.

5.2.2 Support for IP and Legacy Blocks

PTPX supports commands to annotate internal and leakage power for IPs (black boxes), legacy blocks and leaf cells.

> *Ex: set_annotated_power -internal_power 0.01 -leakage_power 0.001 cell_list*

This example annotates the internal power of value 0.01 W and leakage power of value 0.001 W for the cells in the cell list.

5.3 Switching Activity Interchange Format (SAIF) Flow

PTPX also supports SAIF, which is a compact ASCII file that you can create directly from HDL simulation or by using the vcd2saif utility in UNIX provided with PTPX. SAIF provides the toggle rate. PTPX reads the design data and annotates the switching activity from SAIF. For unannotated nets, it propagates the default activity with zero-delay simulation.

In this flow activity is specified using the **read_saif** command as shown below. The **-rtl_direct** option is used to specify the RTL SAIF file.

> *Ex: read_saif rtl.saif –rtl_direct –strip_path tb/top_inst*

This example specifies RTL SAIF activity rtl.saif for top instance top_inst in a test bench tb.

The SAIF flow provides better results compared to the vector-free flow. The **update_power** performs average power analysis while **create_power_waveforms** generates average cycle waveforms and **report_power** reports average power results similar to the vector-free flow.

How do you create a SAIF file?

There are two ways to create a SAIF file as discussed below:

1. From HDL simulation, add the toggle commands as shown below to the test bench to create a SAIF file.

```
Ex:  ‘define SIM_TIME 2000
     module tb;
     top inst(a,b,c,s);
```

```
initial begin
        $set_toggle_region "tb inst"
        $set_gate_level_monitoring ("rtl_on")
        $toggle_start;
        #'SIM_TIME
        $toggle_stop;
        $toggle_report ("my_out.saif", 1.0e-9, "tb inst");
    end
    initial
      #'SIM_TIME $finish;
    endmodule // testbench
```

The **set_toggle_region** command selects instance "inst" to monitor and the **set_gate_level_monitoring** command specifies that the simulation is for RTL. This command should not be used for gate-level SAIF. Use **toggle_start** and **toggle_stop** to specify the start and end of simulation timing. "my_out.saif" is the output file specified using the **toggle_report** command.

Use a simulation command:

Ex: VCS –R -I tb.v design.v

This example shows the simulation command for a verilog test bench tb.v and a verilog design (RTL/gate level) design.v.

2. Use the vcd2saif utility provided with PTPX:

Apply the following command in UNIX:

Ex: unix> vcd2saif –I <input.vcd> -o <out.saif>

5.4 RTL to SAIF/VCD Name Mapping

PrimeTime PX allows you to backannotate activity obtained from RTL simulation. However, depending on the commands applied during synthesis, the RTL simulation objects may not have the same name as their gate-level netlist counterparts.

The following enhancements improve RTL-to-gate name mapping:

- Power Compiler 2006.06-SP3 supports mapping files that contain the **set_rtl_to_gate_name** command used by Primetime PX.

- New commands assist in debugging mapping issues.

The following example script shows how to use the Power Compiler-generated map file in PTPX.

```
set power_enable_analysis true
set search_path ...
set_link_library ...
read_verilog <mapped_design.vs>
current_design <design>
link
source ptpx_map.tcl
read_vcd -strip_path <path to instance> <VCD file>
report_name_mapping
report_switching_activity
quit
```

Refer to the appendix for instructions on generating the map file (ptpx_map.tcl) in Power Compiler.

5.4.1 Verifying Switching Activity Annotation

PrimeTime PX provides the following commands that allow you to verify the annotation:

- **report_name_mapping**

  Returns the nets that were mapped.

- **report_switching_activity**

  Returns the annotation sources. It also allows you to determine whether the annotation of sequential elements and primary inputs is high enough to ensure accurate analysis. Additional options for this command isolate low activity nets and provide the toggle rate per block. Examining the switching activity can help determine the quality of simulation vectors.

6 Peak Power Analysis

PTPX peak power analysis supports gate-level simulation activity over time in the following VCD formats:

- Standard ASCII VCD
- Compressed VCD files such as binary VPD files generated by VCS or with its PLI with vpd as suffix, binary FSDB files generated by Novas PLI with fsdb as suffix, and compressed GZ or Z files with gz or z as suffix. PTPX treats all other suffixes as ASCII VCD files.

PTPX analyzes extremely accurate power with respect to time using event transition value and transition time; and generates detailed time-based power waveforms based on the VCD resolution to provide both average and peak power results.

Power analysis with VCD is the most detailed and accurate compared to all other flows because VCD files from gate-level simulations contain detailed switching activity. The VCD file has headers, nodes, and value change sessions that provide detailed event-based accurate activity information. PTPX supports processing VCD files that are greater than 2GB on all platforms.

Figure 7 shows the usage for peak power analysis with gate-level VCD. In this flow, activity is specified using the **read_vcd** command. PTPX also supports piping the activity data directly from the HDL simulation.

> *Ex: read_vcd gate_vcd.dump –pipe_exec “exec_vcs_cmd” –strip_path tb/top_inst*

where “exec_vcs_cmd” is a file containing the simulation run script, say for example, “VCS –R –I tb.v gate.v”.

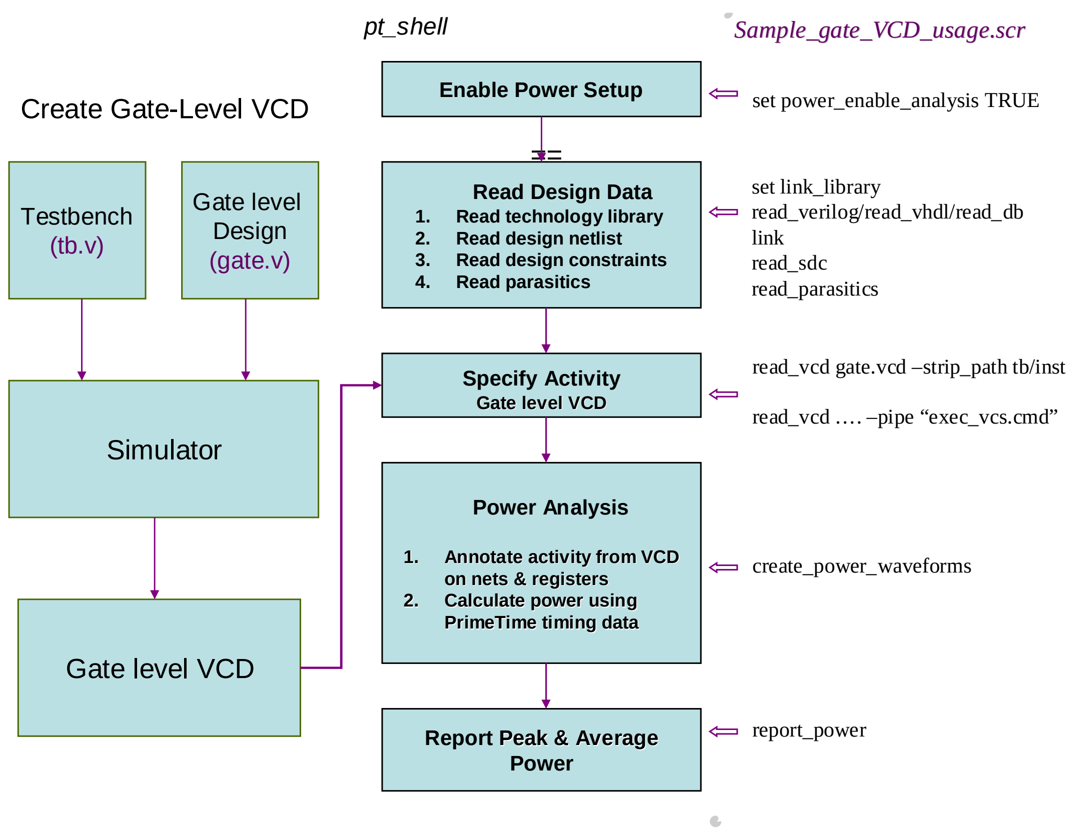

pt_shell
Sample_gate_VCD_usage.scr
Create Gate-Level VCD
Testbench (tb.v)
Gate level Design (gate.v)
Simulator
Gate level VCD
Enable Power Setup
Read Design Data
1. Read technology library
2. Read design netlist
3. Read design constraints
4. Read parasitics
Specify Activity
Gate level VCD
Power Analysis
1. Annotate activity from VCD on nets & registers
2. Calculate power using PrimeTime timing data
Report Peak & Average Power
set power_enable_analysis TRUE
set link_library
read_verilog/read_vhdl/read_db
link
read_sdc
read_parasitics
read_vcd gate.vcd –strip_path tb/inst
read_vcd …. –pipe “exec_vcs.cmd”
create_power_waveforms
report_power

Figure 7: Gate-Level VCD Flow for Peak Power Analysis

While reading the VCD file, PTPX verifies that all the nets in the design are identified in the VCD file. Peak power analysis is enabled by the **create_power_waveforms** command. It calculates transition time and load for every leaf pin, processes each single event in the simulation activity file, and distributes the energy evenly for every event over the transition time for inputs and over the path delay from related pin to output for outputs. The power waveform and peak power are calculated from the superposition of the distributed power thereby providing accurate peak power results. By default, PTPX applies the VCD resolution for accurate peak power analysis; however, if you are interested in cycle-based power, the **create_power_waveforms –interval <ns>** option allows you to view the power for the specified interval. The **report_power** command reports both peak power and average power results.

7 Summary

Accurate power analysis is dependent on accurate timing analysis. In the front end, average power analysis is required. In the back end, peak power analysis becomes more vital.

Quality of the input switching activity determines the accuracy of power analysis; therefore, the more accurate the input activity provided, the more accurate the results.

The following table summarizes the type of power analysis done by PTPX depending on the available activity.

Analyzed PTPX Power Versus Activity

Activity			Average Power	Peak Power
Method	Type	Simulation Level		
Simulation	VCD	GATE	Yes	Yes
		RTL	Yes	No
No Simulation	User-Defined Switching	-	Yes	No
	Default Switching	-	Yes	No

8 Appendix: Generating a Map File During Synthesis

As mentioned in Section 5.4, for generating a map file during synthesis, please follow the procedure outlined here. First, set the environment variable to enable map file generation from Design Compiler, as follows:

```
% setenv SNPS_SAIF_MAP_ON TRUE
```

This variable is case sensitive.

Next, adapt the following sample script in dc_shell to generate the map file (note the commands in bold font):

```
set power_preserve_rtl_hier_names true
set search_path ...
set_link_library ...
set_target_library ...
saif_map -start
read_verilog <Verilog files>
current_design <design>
link
.....
compile
saif_map -create_map -instance <design> -input <SAIF file>
        instance <path to instance>
write -hier -f Verilog -o <mapped_design.v>
saif_map -write_map -type ptpx <ptpx_map.tcl>
....
```