PrimeTime® VX User Guide

Version C-2009.06, June 2009

SYNOPSYS®

Contents

Preface

This preface includes the following sections:

What’s New in This Release

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

To see the *PrimeTime VX Release Notes,*

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

   https://solvnet.synopsys.com/DownloadCenter

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

About This User Guide

This user guide describes the features and usage flow of PrimeTime VX, a tool that adds variation-aware analysis capabilities to PrimeTime. A variation-aware analysis takes into consideration the statistical distribution (rather than simple minimum-typical-maximum values) of process parameters, thereby increasing the accuracy of the analysis.

Audience

This user guide is for engineers who use PrimeTime for static timing analysis. Prior knowledge and experience with statistics and probability will be helpful in understanding the concepts presented.

Related Publications

For additional information about PrimeTime VX, see Documentation on the Web, which is available through SolvNet at the following address:

https://solvnet.synopsys.com/DocsOnWeb

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

- PrimeTime and PrimeTime SI
- Star-RCXT
- Library Compiler
- Liberty NCX

Conventions

The following conventions are used in Synopsys documentation.

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

Customer Support

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

Accessing SolvNet

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

To access SolvNet, go to the SolvNet Web page at the following address:

https://solvnet.synopsys.com

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

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

Contacting the Synopsys Technical Support Center

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

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

1 Overview

PrimeTime VX adds variation-aware analysis capabilities to PrimeTime. A variation-aware analysis increases the accuracy of timing analysis by considering the statistical distribution of process, voltage, and temperature parameters.

This chapter contains the following sections:

Variation-Aware Timing Analysis

PrimeTime VX increases the accuracy of timing analysis in PrimeTime by considering the statistical variation and distribution of process, voltage, and temperature parameters. PrimeTime VX accurately determines the timing behavior of a circuit under varying parameters, including measured parameters such as channel length and threshold voltage, process parameters such as implant dose or gate oxidation temperatures, or parameters derived by principal component analysis from other parameters. Given a set of variation-aware cell libraries and the distribution of parameter values, PrimeTime VX analyzes path timing behavior under all combinations of those parameters.

Variation-aware analysis is more accurate than conventional minimum/maximum analysis because it takes into account the actual distributions of parameters instead of simply combining the absolute worst-case combinations of parameters. Instead of analyzing a limited number of corner cases, PrimeTime VX covers all combinations of parameters, through the full range of parameter values. The calculated slack distributions more accurately reflect the true results obtained in silicon, resulting in less pessimism in the analysis.

PrimeTime VX can handle timing analysis for any type of parameter, as long as the dependency of circuit behavior on that parameter is known and can be characterized.

Handling Variation by Considering Corner Cases

To calculate the delay of a path, PrimeTime must first calculate the delays of the cells and net parasitics in the path. To calculate the delay of a cell, PrimeTime considers the library model for the cell and the cell environment: the input slew, output load, process, voltage, and temperature. Thus, the delay is a function of these variables:

$$delay = f1(slew\_in, load, P, V, T)$$

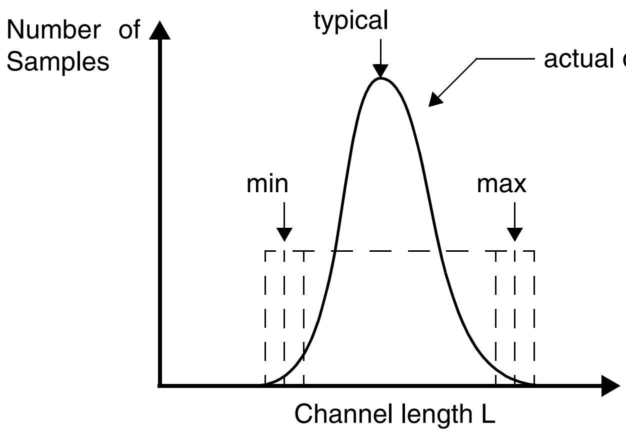
PrimeTime also calculates the output slew, which is needed for delay calculation of the next stage of the timing path:

$$slew\_out = f2(slew\_in, load, P, V, T)$$

The *P* (process) variable represents all the process-related parameters that can affect the delay, such as channel length and threshold voltage. To determine the circuit timing under varying process parameters using conventional min/max analysis, you must characterize the delays at the most extreme values for those parameters and put that information into a set of libraries, one library per operating point.

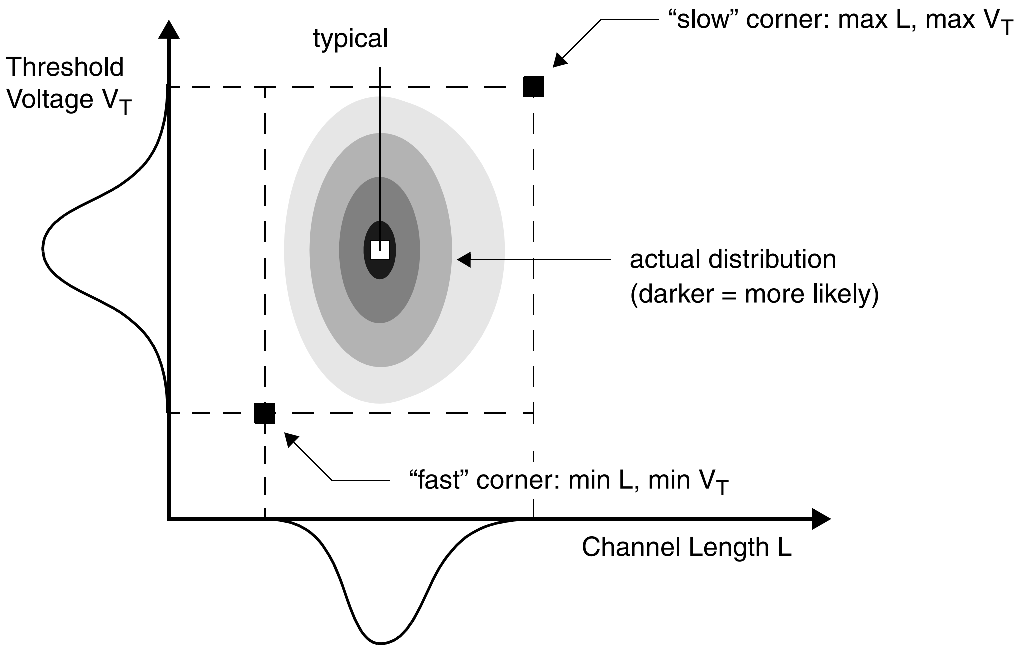
For example, suppose that the statistical distribution of channel lengths for a process technology is a curve like the one shown in Figure 1-1. To get the worst-case behavior for this distribution, you would choose the minimum and maximum ends of the range and determine the effects at those points.

Figure 1-1 Statistical Distribution of a Process Parameter



If another process parameter has significant variation effects, it must be considered as well. For example, to consider both channel length and threshold voltage, you might consider a two-dimensional plot like the one shown in Figure 1-2.

Figure 1-2 Corner Cases Using Two Parameters


"slow" corner: max L, max V_T
typical
Threshold Voltage V_T
actual distribution (darker = more likely)
"fast" corner: min L, min V_T
Channel Length L

The set of conditions at the most extreme values for two or more different parameters is called a corner case. For example, in Figure 1-2, the “fast” corner is the place where the channel length and threshold voltage are both at their minimum values. Similarly, the “slow” corner is the place where the channel length and threshold voltage are at their maximum values. This principle can be extended to any number of parameters that have significant effects.

A typical library set might consist of three libraries characterized at three points: slow, typical, and fast. However, this analysis is pessimistic because it considers the combination of the most extreme parameter values, which is very unlikely to occur in practice.

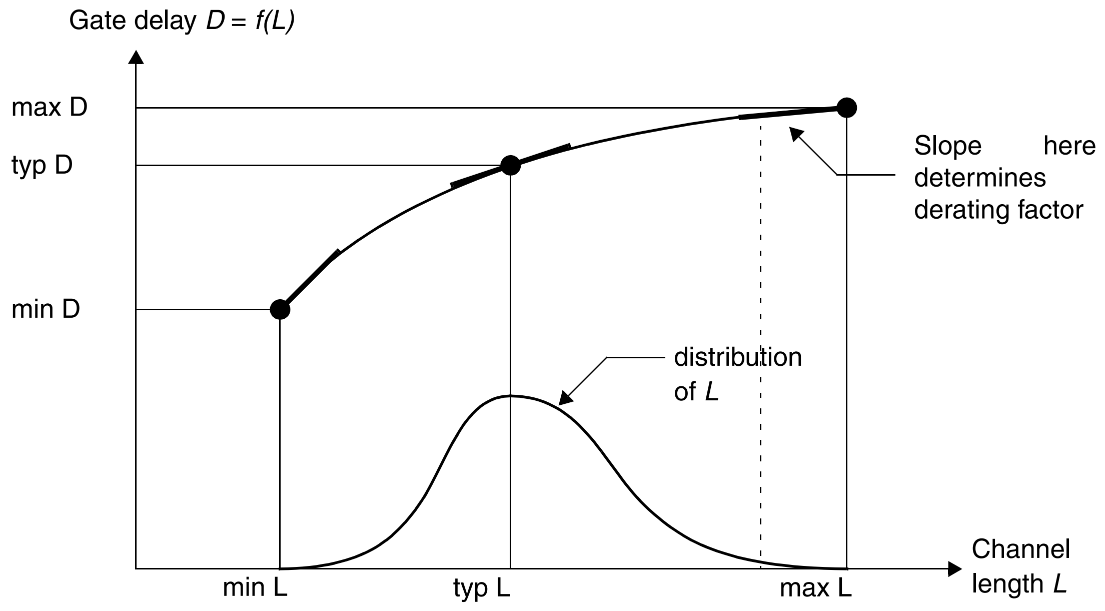
For better accuracy, all the corner cases should be considered. A parameter that affects cell delays in one way (such as fast) might affect net delays in a different way (such as slow). So you might consider four corner cases for two parameters, eight corner cases for three parameters, and so on.

In the past, using a few corner cases has often been adequate due the fairly limited amount of variation in process parameters. However, in current deep-submicron process technologies, variations in process parameters are becoming more significant, causing corner cases to become more pessimistic and making it increasingly difficult to accurately manage multiple parameter variations.

Handling Variation by Derating

One way to manage small variations in parameters is by derating the corner cases, also known as “scaling” or “multiplying by K factors.” Figure 1-3 demonstrates how derating can be used to analyze circuit behavior with known variations in parameters.

Figure 1-3 Derating Delays for Different Channel Lengths



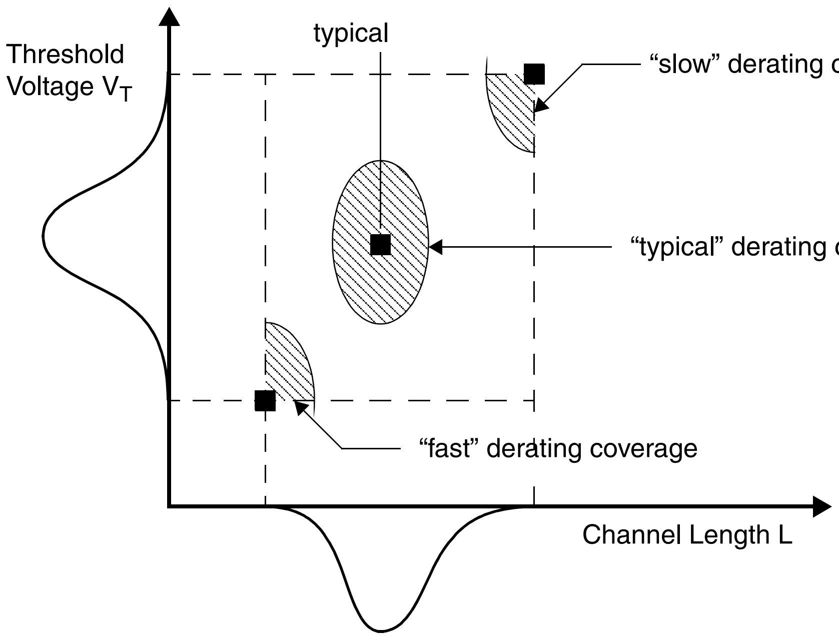
In this example, the variation in gate delay is a function of transistor channel length. For simplicity, variations in other parameters that might affect gate delay are ignored.

Characterization of the technology finds that the gate delay varies with channel length as shown by the curve at the top of Figure 1-3. You create three libraries called min, typ, and max that have the gate delays min D, typ D and max D. These libraries can be used in PrimeTime to determine the circuit behavior at the minimum, typical, and maximum channel lengths.

You want to determine the circuit timing when the channel length is a little less than max L, as represented by the dotted line in the figure. You can get fairly good results without generating a new library by specifying a derating factor with the max L library. The slope of the curve at the maximum operating point gives a good approximation of delay as a function of L, as long as the value of L is not too far away from max L. Thus, you use the max L library and apply the derating factor in PrimeTime with the `set_timing_derate` command. A separate analysis run is necessary for each derating factor chosen for analysis.

Similarly, you can apply different derating factors to the typ and min libraries to get good results for values of L near typ L and min L. Expanding derating to two or more parameters increases the scope of the analysis to areas surrounding the typical case and corner cases, as illustrated in Figure 1-4.

Figure 1-4 Corner Cases Using Two Parameters



Because of the nonlinear dependence of D on L, this technique works well only when the actual value of L is near the minimum, typical, or maximum value of L.

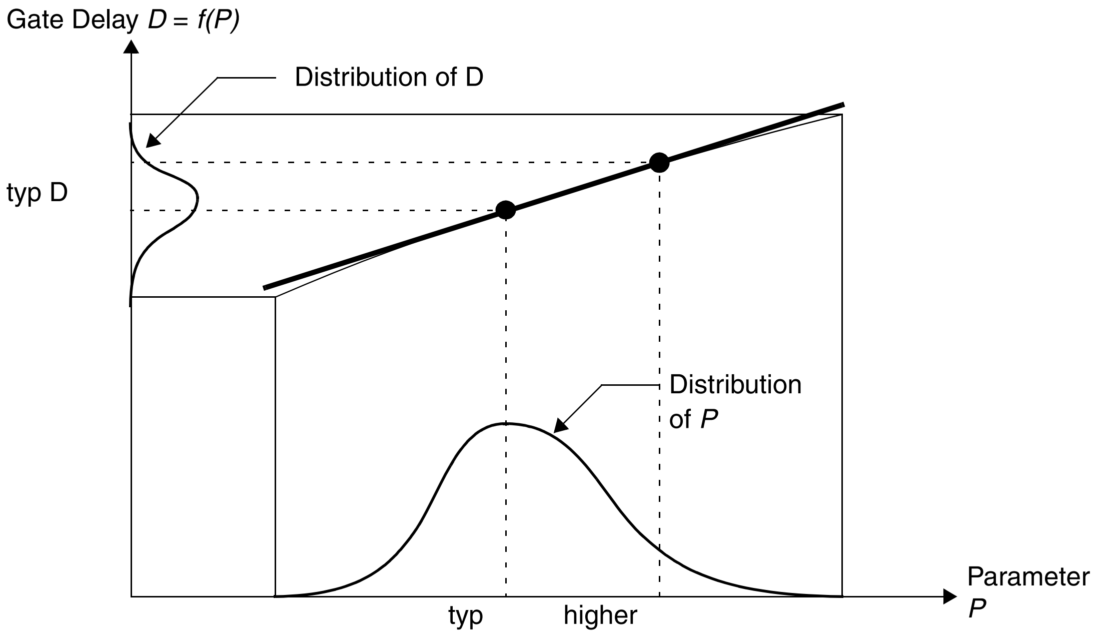
Handling Variation With Distribution Data

In a variation-aware analysis, PrimeTime VX considers the actual distribution of parameter values, rather than just the minimum and maximum extremes. It considers the functional dependence of delay, slew, and other circuit behavior on the parameter, rather than just the few characterization points contained in individual libraries.

For example, in an analysis that considers the effects of channel length on gate delays, you provide information on the distribution of channel lengths in PrimeTime VX. Then PrimeTime VX considers the functional dependence of delay on channel lengths and calculates the distribution of gate delays and slews, and from that information, calculates the distribution of required times, arrival times, and slacks in the circuit.

To specify the functional dependence of delay or slew on a parameter, the cell libraries need to characterize the behavior at the typical value and one or two nearby values. For variations that have highly linear delay and slew response, just the typical data point and one additional data point, either above or below the typical point, provide sufficient accuracy. An example is shown in Figure 1-5.

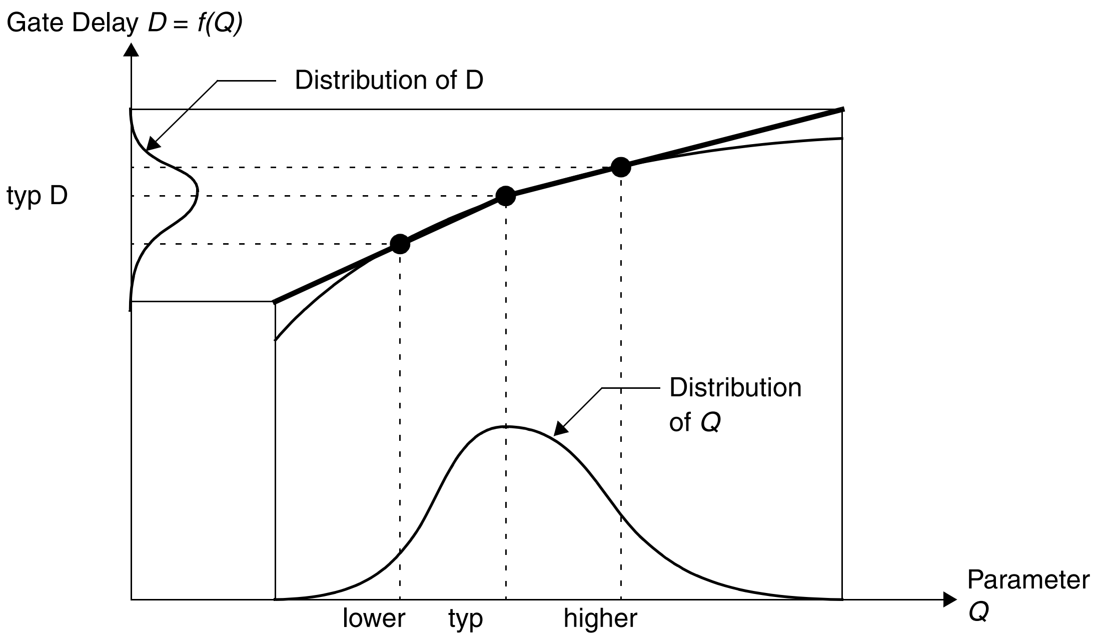
Figure 1-5 Delay as a Function of Parameter P



Given the distribution of the parameter *P*, PrimeTime VX calculates the distribution of delays continuously throughout the range of values, using linear interpolation and extrapolation of the library-defined functional operating points. In this example, interpolation and extrapolation between the typical data point and a single data point characterized at the high end of the distribution provide good accuracy throughout the expected range of the parameter.

If the parameter variation has a strong effect on delay or slew and is significantly nonlinear, you should characterize two data points, one on each side of the typical data point to get higher accuracy throughout the variation range, as shown in Figure 1-6.

Figure 1-6 Delay as a Function of Parameter Q



The surrounding data points should be close enough to the typical operating point to ensure good accuracy at the middle of the distribution, where the actual parameter value is most likely to occur, but also far enough away to get good accuracy at the more extreme parameter values, where timing violations are most likely to occur.

For within-die variation, characterization of data points located one standard deviation (1s) away from the typical value is recommended, to get the best probable average accuracy along the curve. For die-to-die variations, characterization of data points located three standard deviations (3s) away from the typical value is recommended, to get better accuracy at the tail ends of the distribution where violations are more likely to occur.

To perform variation-aware analysis with two parameters, you must characterize the timing behavior at three, four, or five operating points: the typical data point, plus one or two more data points for the first parameter (with the second parameter at its typical value), plus one or two more data points for the second parameter (with the first parameter at its typical value. For example, for two parameters L and Vt, you might choose three points like the ones shown in Figure 1-7 or five points like the ones shown in Figure 1-8.

Figure 1-7 Three Characterization Points for Two Parameters

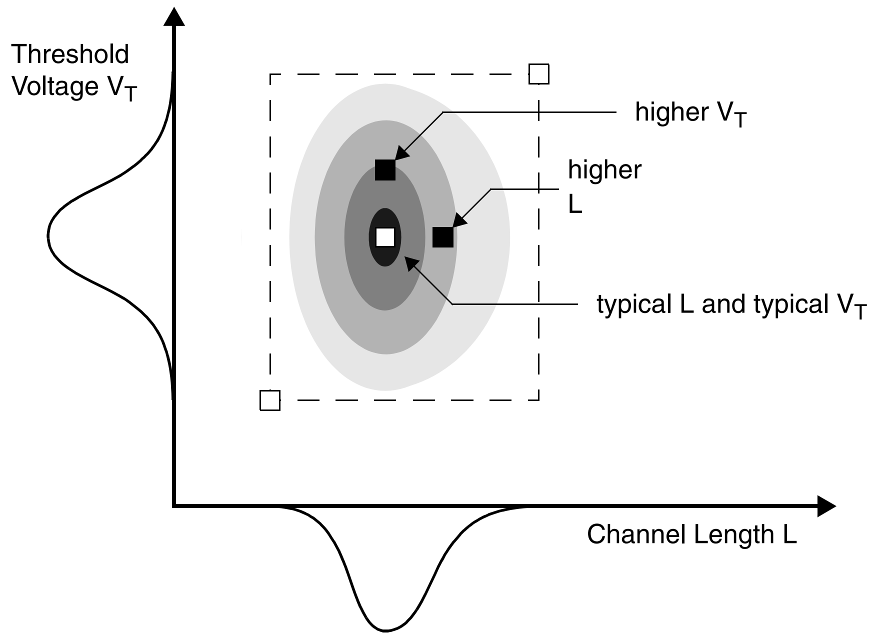
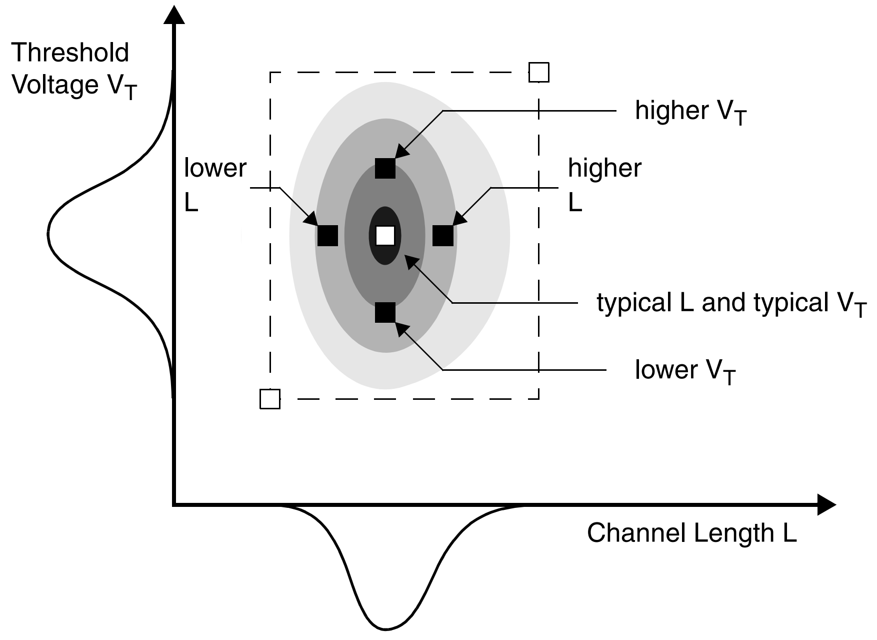


Figure 1-8 Five Characterization Points for Two Parameters



The same principle applies to three or more parameters. If *N* is the total number of parameters under consideration, the number of characterization points must be at least *N*+1, consisting of the typical data point and plus one additional data point for each parameter; or it can be as many as 2*N*+1, consisting of the typical data point plus a lower and a higher operating point for each parameter.

Characterization of the typical value plus one or two surrounding values for each parameter is sufficient for accurate timing analysis throughout the expected variation ranges, while requiring a minimum number of data points to be characterized. Any variations or changes in parameter values can be handled quickly in PrimeTime VX, avoiding the slow process of recharacterization and library generation.

Variation-Aware and Corner-Case Min/Max Analysis

Figure 1-9 and Figure 1-10 compare the analysis flows using conventional and variation-aware timing analysis.

With conventional corner-case analysis, a library must be generated for each corner case. Typically, at least three libraries are used: minimum, typical, and maximum libraries. At least three runs are required for a full analysis, one using each of the three libraries. Parameter variation can be handled by a limited extent by derating, although deciding on an appropriate derating factor can be difficult. In the end, PrimeTime reports the amount of slack under the worst-case corner conditions, possibly adjusted by derating.

Figure 1-9 Conventional Corner-Case Timing Analysis

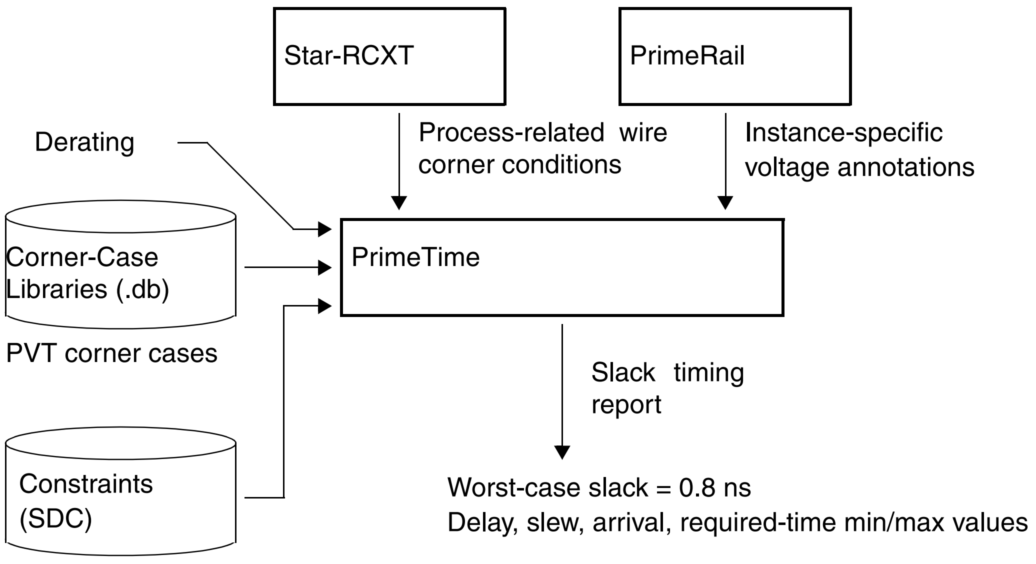
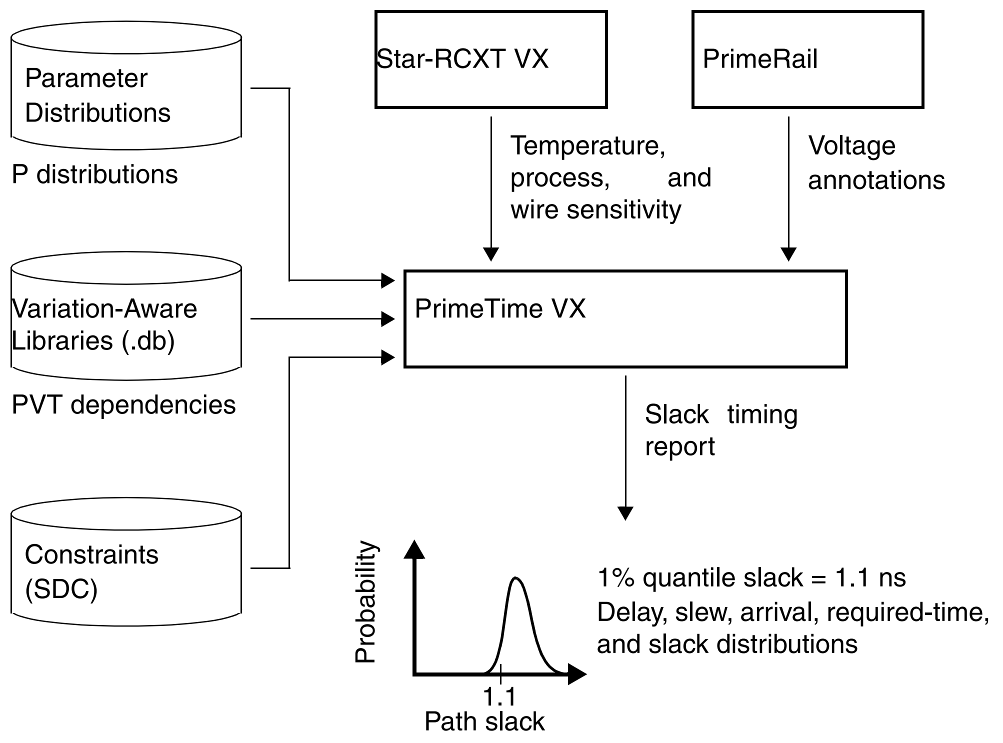

Star-RCXT
PrimeRail
Derating
Process-related wire corner conditions
Instance-specific voltage annotations
Corner-Case Libraries (.db)
PrimeTime
PVT corner cases
Slack timing report
Constraints (SDC)
Worst-case slack = 0.8 ns
Delay, slew, arrival, required-time min/max values

Figure 1-10 Variation-Aware Timing Analysis



With variation-aware analysis, a set of libraries is generated to describe dependence of delay and slew as a function of the parameters under consideration. PrimeTime VX considers the full range and distribution of the parameters of interest. In the end, PrimeTime VX reports the delay, slew, required time, arrival time, and slack based on quantiles of the value distributions.

When you run a path-based analysis with variation, PrimeTime VX maintains all the statistical information in memory. Then you can display the statistical results graphically and query the variation results with the `get_attribute` command to obtain the mean, standard deviation, and quantile values for each type of timing result.

The worst-case slack reported by conventional min-max analysis is usually worse (more pessimistic) because it combines all the most extreme, worst-case delays from each stage in the path, whereas the variation-aware analysis statistically combines the distribution of delays from each stage. Figure 1-11 shows how PrimeTime calculates the cumulative delay along this path using conventional min/max analysis, while Figure 1-12 shows how it calculates the same cumulative delay using variation-aware analysis.

Figure 1-11 Conventional Addition of Min/Max Gate Delays

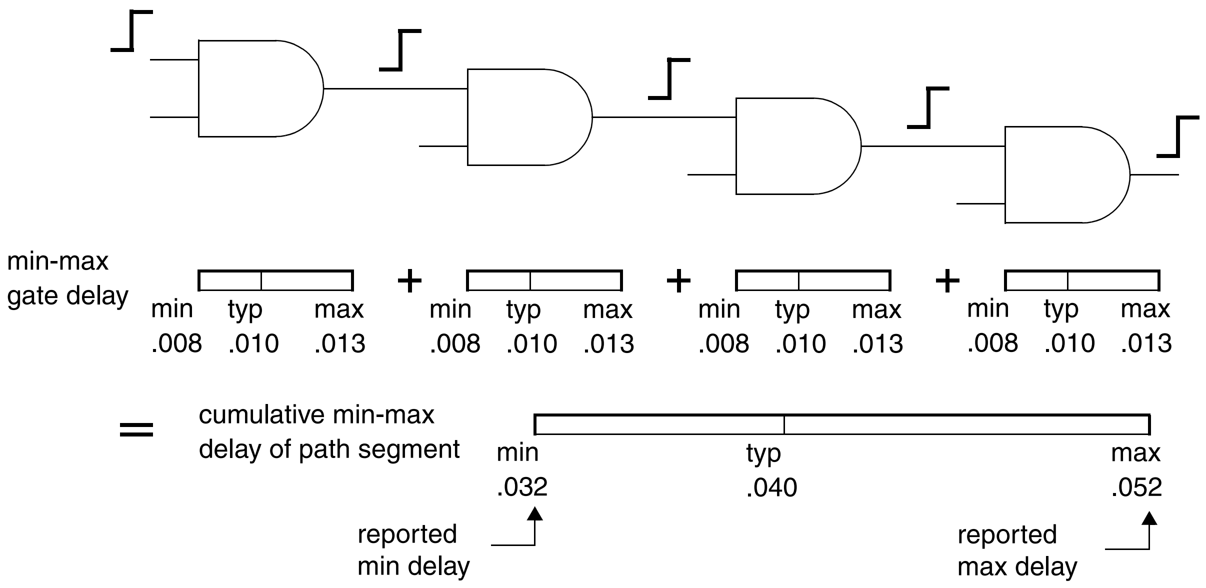
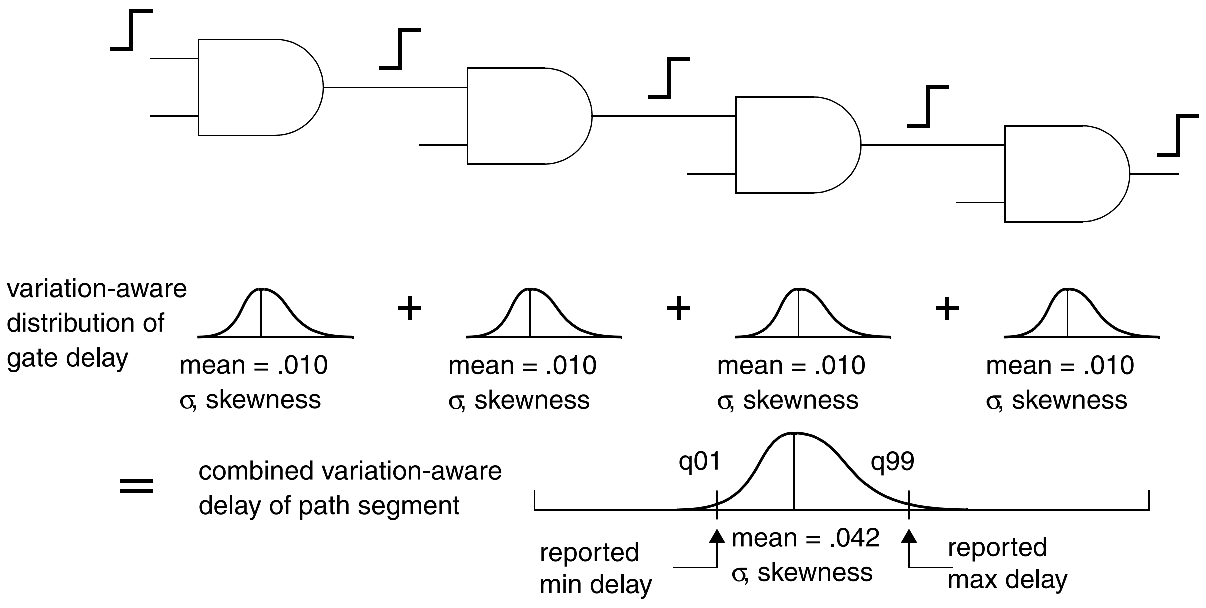


Figure 1-12 Variation-Aware Accumulation of Gate Delays



For this example, we consider only the gate delays and ignore the net delays. Assume also that the variation in gate delays is random and not correlated between different gates in the path.

Note:
In this manual, "correlated" means "statistically dependent" and "uncorrelated" means "statistically independent."

Using conventional min-max analysis as shown in Figure 1-11, PrimeTime adds the minimum delays to obtain the total minimum delay of the path segment, and it adds the maximum delays to obtain the total maximum delay of the path segment. This gives the worst-case extremes of total delay, without information about the relative likelihood between these two extremes.

Using variation-aware analysis as shown in Figure 1-12, PrimeTime VX combines the delays statistically to get the distribution of the total delay for the path segment. This results in a more realistic, less pessimistic model of total delay for the path segment.

Using variation-aware analysis, for reporting and comparison purposes, PrimeTime VX considers a low quantile point of the distribution as the worst-case low value, and a high quantile point of the distribution as the worst-case high value. By default, the 0.135 percent and 99.865 percent quantiles are used to report the minimum and maximum values, respectively, corresponding to the –3*sigma and +3*sigma points of a perfectly normal distribution. You can specify other quantiles (for example, 5 percent and 95 percent) by using the `set_variation_quantile` command.

Choosing Parameters for Analysis

In choosing variable parameters for analysis, you should focus on the few dominant parameters that have the most variation and the largest effects on delay and slack. There are three main types of parameters that you can characterize and analyze:

- Measured parameters such as channel length and threshold voltage
- First-level process parameters such as implant dose and gate oxidation temperature
- Uncorrelated parameters derived from first-level process parameters by principal component analysis

PrimeTime VX can accept any type of parameter that can be characterized, including derived parameters created by principal component analysis. A derived parameter is a mathematical representation of multiple correlated parameters. Each derived parameter has no physical meaning by itself. The process of principal component analysis transforms a number of correlated parameters into a smaller number of uncorrelated parameters. The new parameters created by this process are called the principal components of the larger set of parameters. In this way, a large number of correlated parameters can be reduced to a few principal components.

A process technology can be characterized by first analyzing the large number of process parameter variations, followed by principal component analysis to reduce those variations to a few uncorrelated components. These components can be used in a variation-aware analysis to account for variations in the analyzed process parameters.

PrimeTime VX can handle multiple components per parameter. For example, a channel-length variation can be written as a sum of uncorrelated terms such as the following:

$L_{total} = L_{ref} + DL_{fab} + DL_{lot} + DL_{wafer} + DL_{die}$

Each term can be further decomposed into a set of uncorrelated components:

$DL_{fab} = DL_{fab1} + DL_{fab2} + DL_{fab3}$

The reference component is the primary component that carries the mean value of the parameter variation. The other components represent the variation in the parameter that can be attributed to various causes, such as the fab-to-fab variation, lot-to-lot variation from a fab, wafer-to-wafer variation in a lot, die-to-die variation in a wafer, within-die variation, and so on. Each variation component has a different form and magnitude. For example, the lot-to-lot variation might be larger than the wafer-to-wafer variation. The mean value of each variation component is zero.

The reference component value is specified in the library, whereas the variation components are specified in the PrimeTime VX environment. Therefore, you can change the variation components and analyze the results with a given set of libraries, without going through the time-consuming process of generating a new set of libraries for each variation change. In PrimeTime VX, the command for invoking a library and its reference component values is `set_variation_library`, using the `-reference_value` and `-values` options. The command for specifying a variation component is `set_variation`.

Analysis Example

The following example demonstrates typical usage of variation-aware analysis in a PrimeTime VX session, including the display of arrival time and slack variations in the PrimeTime graphical user interface (GUI). In this example, the analysis considers the variation of three process parameters and their effects on delay, slew, and slack.

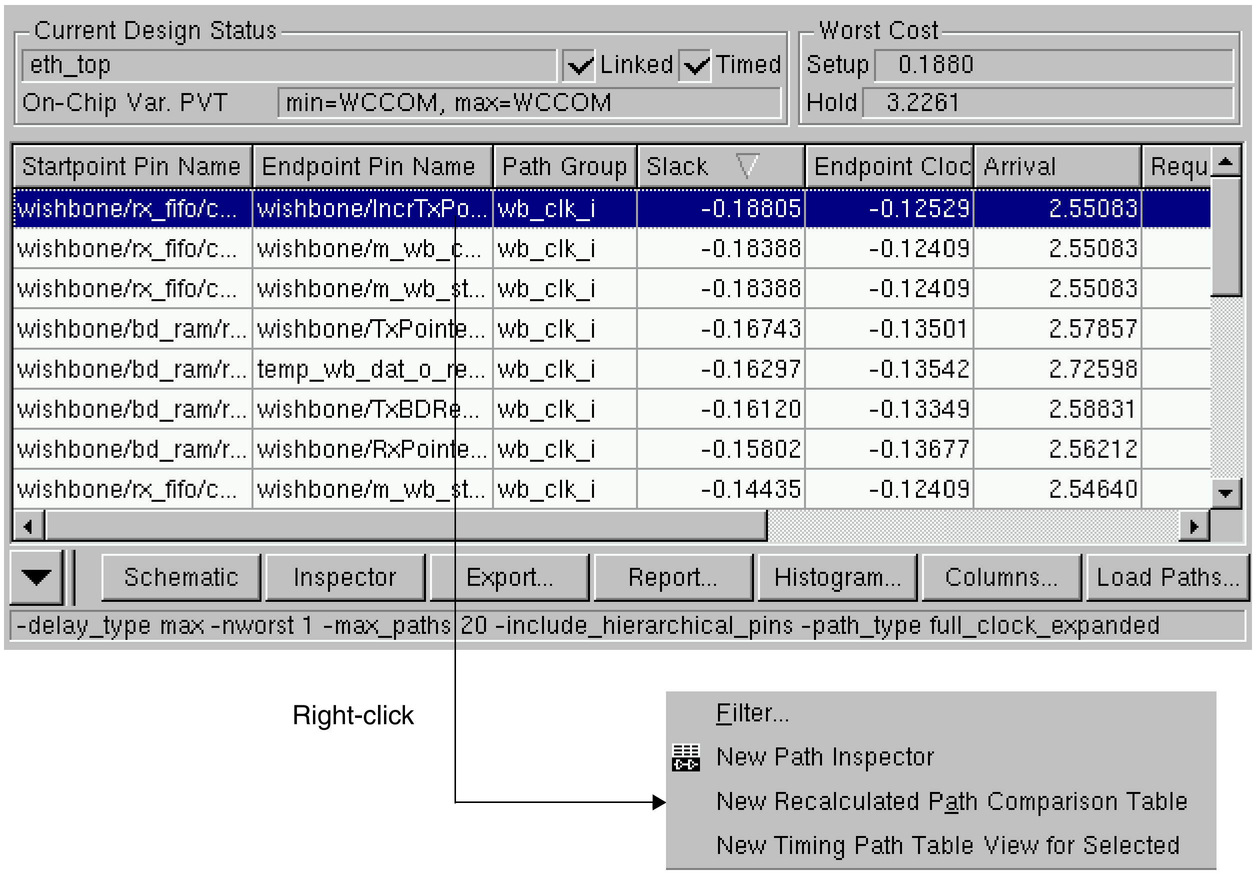
The flow in this example consists of the following steps in PrimeTime VX:

1. Read in and link the design.

2. Enable variation-aware analysis and set the variation-aware libraries used for analysis. These libraries characterize cell behavior at different parameter values.
3. Specify the variation and correlation characteristics of the characterized parameters in the design.
4. Set the timing constraints (clocks, input delays, output delays, and input transition times).
5. Read in the parasitics with variation data. You can either choose a simple normal distribution for the characterized parameters or explicitly specify the variation and correlation characteristics.
6. Run `update_timing` and generate timing reports for the most critical timing paths based on variation analysis.
7. If you want to further reduce any remaining pessimism in the analysis, run a path-based variation analysis of the critical paths. You can get detailed statistical results by reading the timing reports, by viewing graphs and plots in the GUI, and by reading statistical values directly from the design with the `get_attribute` command.

Figure 1-13 shows the GUI Timing Analysis Driver window containing a list of the worst-slack maximum-delay paths in the design. The paths are sorted in order of increasing slack. The worst path has a reported slack of –0.19 time units. PrimeTime VX calculates the path slack as a variation, but for listing and comparison purposes, it uses the low quantile of the variation (0.00135 by default).

Figure 1-13 Recalculated Path Table



Current Design Status

eth_top	✓ Linked	✓ Timed
On-Chip Var. PVT	min=WCCOM, max=WCCOM	

Worst Cost

Setup	0.1880
Hold	3.2261

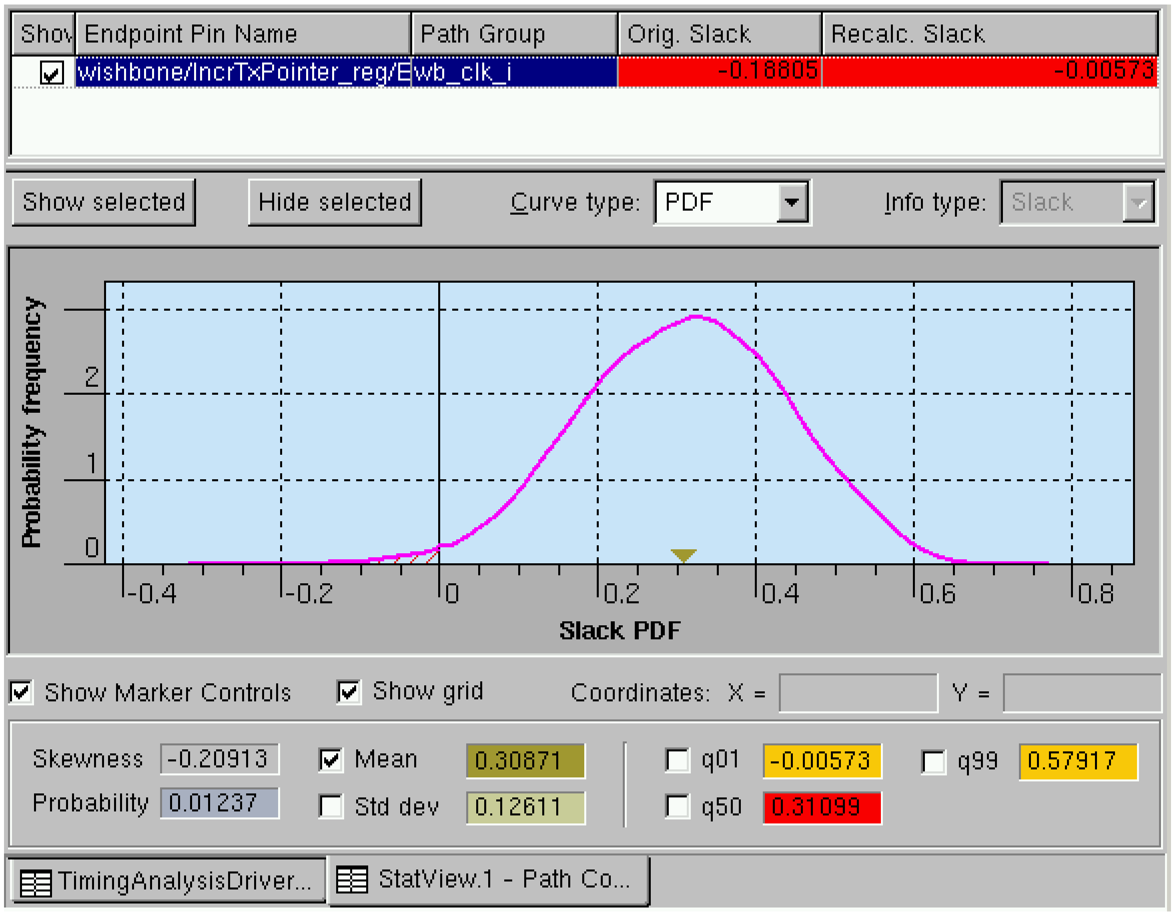
Startpoint Pin Name	Endpoint Pin Name	Path Group	Slack	Endpoint Cloc	Arrival	Requ
wishbone/rx_fifo/c...	wishbone/IncrTxPo...	wb_clk_i	-0.18805	-0.12529	2.55083	
wishbone/rx_fifo/c...	wishbone/m_wb_c...	wb_clk_i	-0.18388	-0.12409	2.55083	
wishbone/rx_fifo/c...	wishbone/m_wb_st...	wb_clk_i	-0.18388	-0.12409	2.55083	
wishbone/bd_ram/r...	wishbone/TxPointe...	wb_clk_i	-0.16743	-0.13501	2.57857	
wishbone/bd_ram/r...	temp_wb_dat_o_re...	wb_clk_i	-0.16297	-0.13542	2.72598	
wishbone/bd_ram/r...	wishbone/TxBDRe...	wb_clk_i	-0.16120	-0.13349	2.58831	
wishbone/bd_ram/r...	wishbone/RxPointe...	wb_clk_i	-0.15802	-0.13677	2.56212	
wishbone/rx_fifo/c...	wishbone/m_wb_st...	wb_clk_i	-0.14435	-0.12409	2.54640	

```
-delay_type max -nworst 1 -max_paths 20 -include_hierarchical_pins -path_type full_clock_expanded
```

Right-click
Filter...
New Path Inspector
New Recalculated Path Comparison Table
New Timing Path Table View for Selected

To get a report on the slack variations for the paths in the list, you select those paths, right-click, and choose New Recalculated Path Comparison Table. PrimeTime VX recalculates the selected paths and saves all the variation-aware timing parameters (arrival times, required times, and slacks) for those paths. It displays the results in a table and a graph as shown in Figure 1-14.

Figure 1-14 Statistical View With Slack PDF Plot



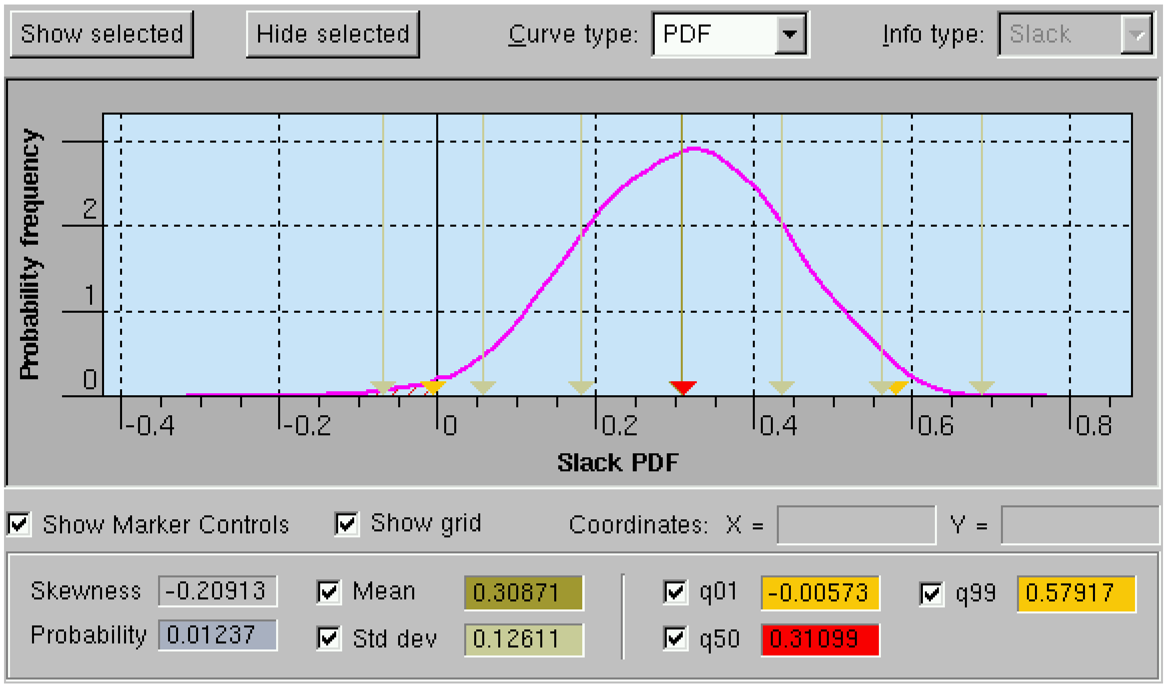
Show	Endpoint Pin Name	Path Group	Orig. Slack	Recalc. Slack
✓	wishbone/IncrTxPointer_reg/E	wb_clk_i	-0.18805	-0.00573

Skewness	-0.20913	✓ Mean	0.30871	q01	-0.00573
Probability	0.01237	Std dev	0.12611	q50	0.31099
				q99	0.57917

In the slack probability density function (PDF) plot, the curve shows the distribution of slack values for the path, given the specified variation in the characterized parameters. The total area under the curve is 1.0. The cross-hatched area under the curve to the left of slack=0.0 represents the probability of timing failure for this constraint. In this case, the probability of failure is 1.2 percent.

The check boxes below the plot let you display or not display the mean, standard deviation, bound, and quantile points of the plot. Figure 1-15 shows the plot with all the markers enabled. The markers are color-coded in the plot.

Figure 1-15 Slack PDF Plot With Markers



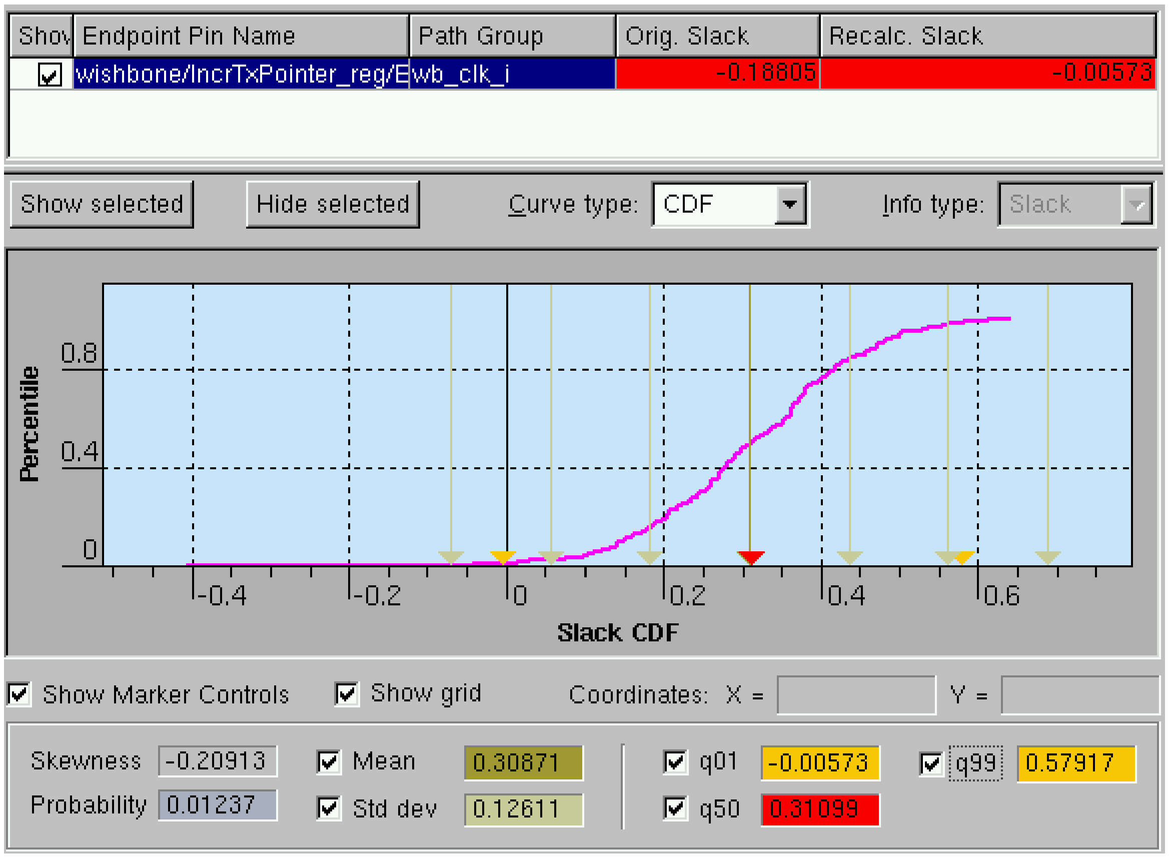
Skewness -0.20913
Probability 0.01237
Mean 0.30871
Std dev 0.12611
q01 -0.00573
q50 0.31099
q99 0.57917

The mean value is marked with a dark olive color. This particular distribution is nearly symmetrical, so the mean is very close to the q50 or median value, marked in red. Multiple standard deviation (s) values around the mean are marked in a light olive color. In this example, essentially all of the curve lies between plus or minus three standard deviations from the mean.

The quantile points marked in the distribution represent the arrival times at which the probability of arrival at less than or equal to that time will be a given value: 1, 50, or 99 percent. For example, the 99 percent (q99) quantile point is at the time value 0.58, which means that the probability of arrival at or before time = 0.58 is 99 percent.

An alternative type of plot of the same data is a cumulative distribution function (CDF). To view a CDF plot, select CDF in the Curve Type pull-down option above the plot. See Figure 1-16. The CDF plot shows the probability that the arrival time will be *less than or equal to* each value.

Figure 1-16 Slack CDF Plot



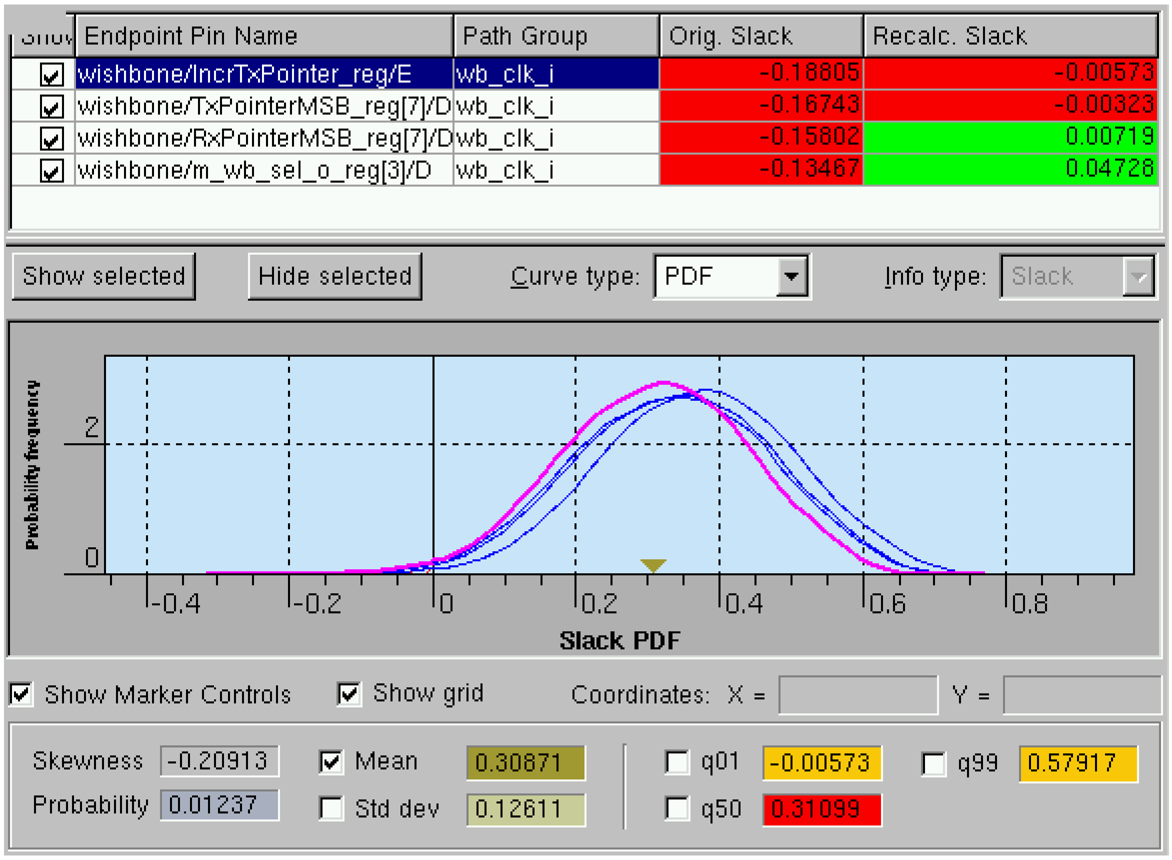
Show	Endpoint Pin Name	Path Group	Orig. Slack	Recalc. Slack
✓	wishbone/IncrTxPointer_reg/E	wb_clk_i	-0.18805	-0.00573

Skewness	-0.20913	✓ Mean	0.30871	✓ q01	-0.00573	✓ q99	0.57917
Probability	0.01237	✓ Std dev	0.12611	✓ q50	0.31099		

The CDF starts at zero for low values of arrival time and increases to 1.0 for high values. This plot is the integral (cumulative area under the curve) of the PDF plot.

You can display multiple, overlapping PDF plots for different paths in the same graph. First select the paths of interest in the Timing Analysis Driver window or by using the Select > Paths From/Through/To. Then choose Timing > New Recalculated Path Comparison Table. See Figure 1-17 for an example.

Figure 1-17 Multiple PDF Plots in One Graph



Show	Endpoint Pin Name	Path Group	Orig. Slack	Recalc. Slack
✓	wishbone/IncrTxPointer_reg/E	wb_clk_i	-0.16805	-0.00573
✓	wishbone/TxPointerMSB_reg[7]/D	wb_clk_i	-0.16743	-0.00323
✓	wishbone/RxPointerMSB_reg[7]/D	wb_clk_i	-0.15802	0.00719
✓	wishbone/m_wb_sel_o_reg[3]/D	wb_clk_i	-0.13467	0.04728

Curve type: PDF
Info type: Slack

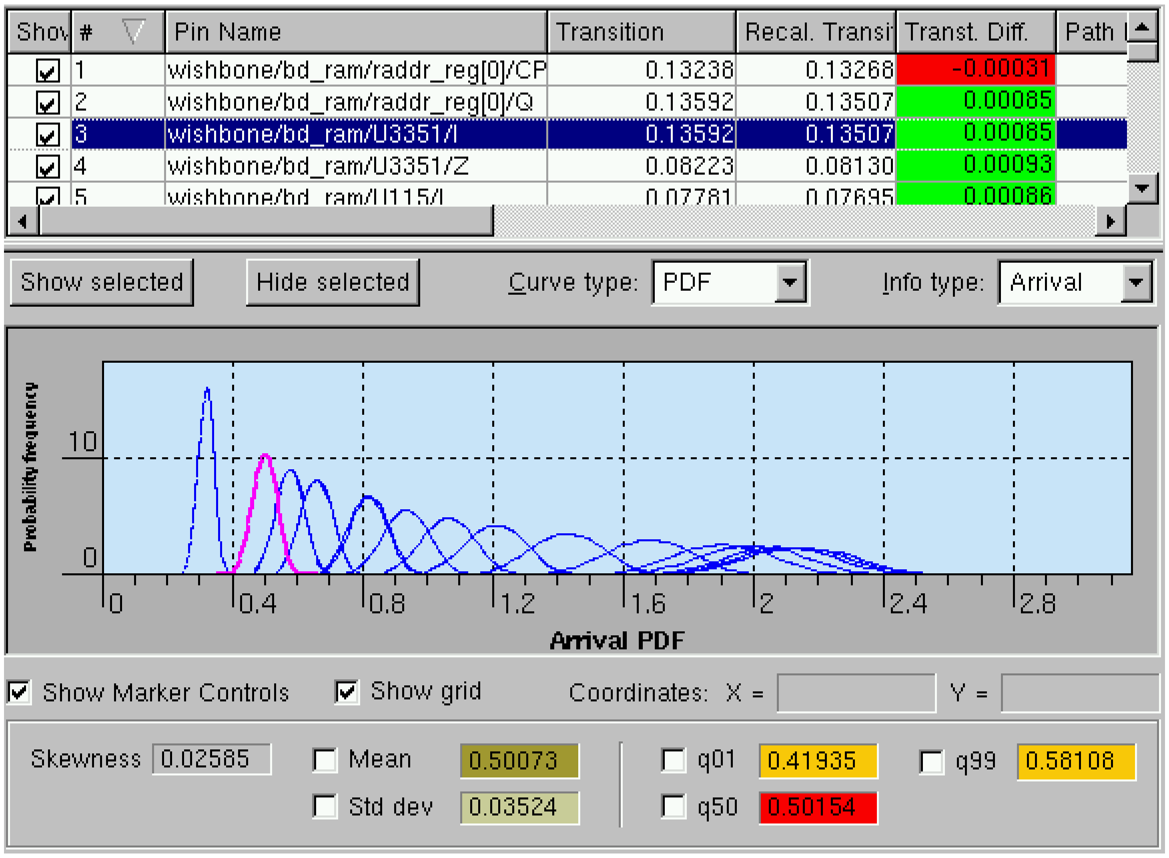
[✓] Show Marker Controls [✓] Show grid Coordinates: X = Y =

Skewness	-0.20913	[✓] Mean	0.30871	[ ] q01	-0.00573	[ ] q99	0.57917
Probability	0.01237	[ ] Std dev	0.12611	[ ] q50	0.31099		

You can show or hide each PDF curve by checking or unchecking the boxes in the Show column of the path list. You can select a path by clicking on its entry in the list or on its PDF curve in the graph. The PDF curve for the selected path is drawn in magenta and the rest are drawn in blue. The statistical markers (mean, quantiles, and so on) are shown for the single selected path only. If multiple paths are selected or no paths are selected, no statistical markers are shown.

For a particular path, you can display the distributions of arrival times at different points along the path. From the recalculated path comparison table, right-click the path of interest and then choose New Path Pin Comparison Table. This displays the arrival PDFs for successive points along the path, as shown in Figure 1-18. Note that the arrival distributions become wider and flatter as you go further along the path.

Figure 1-18 Arrival PDFs for Points Along a Path



Shov	#	Pin Name	Transition	Recal. Transi	Transt. Diff.	Path l
✓	1	wishbone/bd_ram/raddr_reg[0]/CP	0.13238	0.13268	-0.00031	
✓	2	wishbone/bd_ram/raddr_reg[0]/Q	0.13592	0.13507	0.00085	
✓	3	wishbone/bd_ram/U3351/I	0.13592	0.13507	0.00085	
✓	4	wishbone/bd_ram/U3351/Z	0.08223	0.08130	0.00093	
✓	5	wishbone/bd_ram/U115/I	0.07781	0.07695	0.00086	

Curve type: PDF

Info type: Arrival

✓ Show Marker Controls ✓ Show grid Coordinates: X = Y =

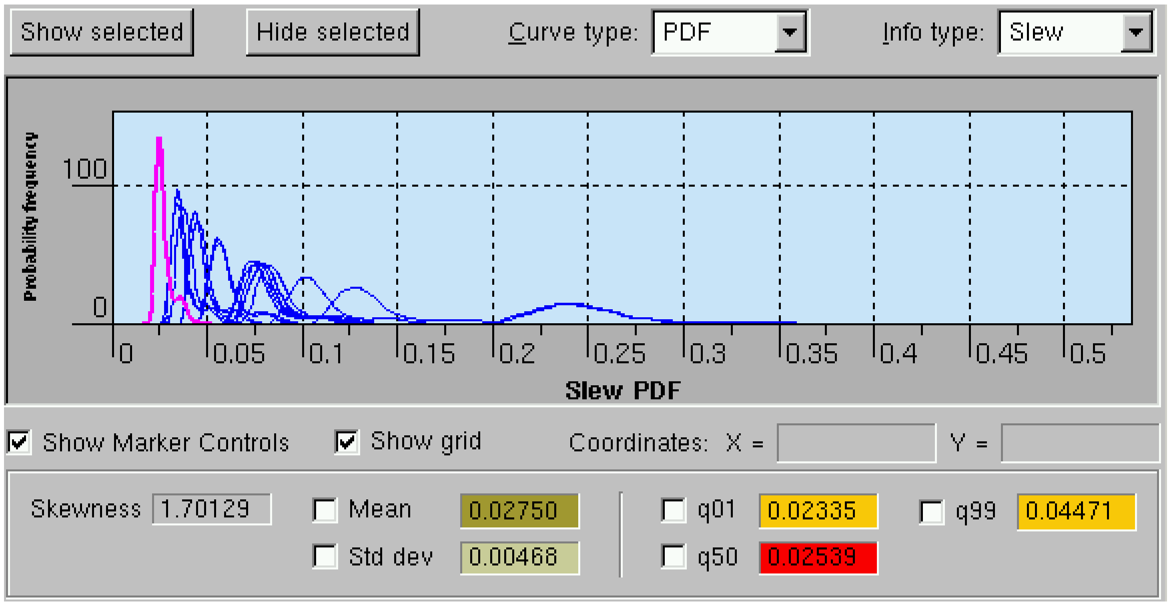
Skewness 0.02585	☐ Mean	0.50073	☐ q01	0.41935	☐ q99 0.58108
	☐ Std dev	0.03524	☐ q50	0.50154	

To view statistical information such as the mean and standard deviation for the arrival time at one point in the path, click on its distribution curve or select the corresponding driver or receiver pin in the path pin list. In the foregoing figure, the arrival at pin U3351/I is selected. The mean arrival time is at 0.50 time units.

To view the plot for just one point in the path, select all of the points, click the Hide Selected button, and then click the Show box in the path pin list.

To view the slew rather than arrival distribution, select Slew in the Info Type pull-down option above the plot. For example, Figure 1-19 shows the slew PDF plots for all the pins along the path.

Figure 1-19 Slew PDF Plot



Skewness 1.70129
Mean 0.02750
Std dev 0.00468
q01 0.02335
q50 0.02539
q99 0.04471

Enabling and Using PrimeTime VX Analysis

Variation-aware analysis is enabled in PrimeTime VX by setting the `variation_enable_analysis` variable to `true`. It is set to `false` by default. You must set it to `true` before you use any variation-related commands. A PrimeTime VX license is required.

Variation Analysis Modes

When the `variation_enable_analysis` variable is set to `true`, a variable called `variation_analysis_mode` specifies the scope of variation-aware analysis. The variable can be set to `default` or `path_based_only`.

In the `default` mode, PrimeTime VX performs variation-aware analysis for both full-chip analysis and path-based analysis. In path-based analysis, you use the `get_timing_paths` and `get_recalculated_timing_paths` commands to analyze a selected set of paths with enhanced accuracy.

In the `path_based_only` mode, PrimeTime VX performs conventional corner-case analysis for full-chip analysis and variation-aware analysis for path-based analysis. For full-chip analysis, PrimeTime uses the corner-case libraries specified by the `link_path` variable. For path-based analysis, PrimeTime VX uses the variation libraries specified by the `set_variation_library` command.

Libraries, Variations, and Correlations

Variation-aware libraries must be read into PrimeTime VX for variation-aware analysis. These libraries are characterized at selected variation operating points, as illustrated in the example shown in Figure 1-7 on page 9. Multiple variation-aware can be merged into a single library to simplify the reading process, or they can be maintained and read in separately. You specify the information about the variations and operating points in PrimeTime VX by using the `set_variation_library` command. For details, see Chapter 2, “Libraries and Interconnect Parameters.”

You specify the actual parameter values in PrimeTime VX by using the `create_variation` command. This command specifies the parameter name, value, and mathematical distribution of the parameter. To apply a variation to the whole design or a timing object in the design, you use the `set_variation` command. The `create_correlation` and `set_variation_correlation` commands specify the correlation relationship between different instances of the same variation. For details, see Chapter 3, “Variations and Correlations.”

Quantile and Distribution Reporting

The `report_timing` command reports the worst-case quantile values for arrival and slack for a corner-case analysis, or statistical distributions of these values for a statistical analysis. In the following example, a corner-case analysis is performed first, followed by reslection and statisitical analysis of the critical path.

```
pt_shell> set variation_anlaysis_mode path_based_only
pt_shell> set_variation_quantile -quantile_low 0.01 -quantile_high 0.99
pt_shell> report_timing -delay_type max -group wb_clk_i
   ...
  Startpoint: wishbone/rx_fifo/cnt_reg[0]
               (rising edge-triggered flip-flop clocked by wb_clk_i)
  Endpoint: wishbone/IncrTxPointer_reg
               (rising edge-triggered flip-flop clocked by wb_clk_i)
  Path Group: wb_clk_i
  Path Type: max

  Point                                                 Incr       Path
  -------------------------------------------------------------------
  clock wb_clk_i (rise edge)                            0.00       0.00
  clock network delay (propagated)                      0.37       0.37
  wishbone/rx_fifo/cnt_reg[0]/CP (SEDFCNQD1)       0.00        0.37 r
  wishbone/rx_fifo/cnt_reg[0]/Q (SEDFCNQD1)        0.43 &      0.80 r
  wishbone/rx_fifo/U71/ZN (NR4D2)                  0.12 &      0.92 f
  wishbone/rx_fifo/U21/Z (AN2D2)                   0.12 &      1.04 f
   ...
  wishbone/U1059/Z (OR2D4)                         0.13 &      2.55 f
  wishbone/IncrTxPointer_reg/E (SEDFCND0)          0.00 &      2.55 f
  data arrival time                                            2.55

  clock wb_clk_i (rise edge)                       2.50        2.50
  clock network delay (propagated)                 0.24        2.74
  clock reconvergence pessimism                    0.00        2.74
  clock uncertainty                               -0.10        2.64
  wishbone/IncrTxPointer_reg/CP (SEDFCND0)                     2.64 r
  library setup time                              -0.28        2.36
  data required time                                           2.36
  ------------------------------------------------------------------
  data required time                                           2.36
  data arrival time                                           -2.55
  ------------------------------------------------------------------
  slack (VIOLATED)                                            -0.19


pt_shell> set path1 [get_timing_paths -delay_type max  \
          -group wb_clk_i -path_type full_clock_expanded]

pt_shell> set path2 [get_recalculated_timing_paths $path1]

pt_shell> report_timing $path2
   ...
  Startpoint: wishbone/rx_fifo/cnt_reg[0]
              (rising edge-triggered flip-flop clocked by wb_clk_i)
  Endpoint: wishbone/IncrTxPointer_reg
              (rising edge-triggered flip-flop clocked by wb_clk_i)
  Path Group: wb_clk_i
  Path Type: max

  Point                                            Incr        Path
  ------------------------------------------------------------------
  clock wb_clk_i (rise edge)                       0.00        0.00
  clock source latency                             0.00        0.00
  wb_clk_i (in)                                    0.00 &      0.00 r
  wishbone/BUFFD16_G1B1I4_2/Z (CKBD16)             0.19 &      0.19 r
  wishbone/BUFFD16_G1B2I135/Z (BUFFD16)            0.17 &      0.36 r
  wishbone/rx_fifo/cnt_reg[0]/CP (SEDFCNQD1)       0.00 &      0.36 r
  wishbone/rx_fifo/cnt_reg[0]/Q (SEDFCNQD1)        0.42 &      0.78 r
  wishbone/rx_fifo/U71/ZN (NR4D2)                  0.12 &      0.90 f
  wishbone/rx_fifo/U21/Z (AN2D2)                   0.12 &      1.02 f
   ...
  wishbone/U1059/Z (OR2D4)                         0.13 &      2.48 f
  wishbone/IncrTxPointer_reg/E (SEDFCND0)          0.00 &      2.48 f
  data arrival time                                            2.48
  clock wb_clk_i (rise edge)                          2.50       2.50
  clock source latency                                0.00       2.50
  wb_clk_i (in)                                       0.00 &     2.50 r
  BUFFD16_G1B1I2/Z (BUFFD16)                          0.16 &     2.66 r
  wishbone/BUFFD16_G1B2I56/Z (BUFFD16)                0.09 &     2.75 r
  wishbone/IncrTxPointer_reg/CP (SEDFCND0)            0.00 &     2.75 r
  clock reconvergence pessimism                       0.00       2.75
  clock uncertainty                                  -0.10       2.65
  library setup time                                 -0.28       2.37
  data required time                                             2.37
  -----------------------------------------------------------------
  data required time                                             2.37
  data arrival time                                             -2.48
  -----------------------------------------------------------------
  slack                                                         -0.11
  statistical adjustment (slack)                      0.10      -0.01
  slack (VIOLATED) (with probability 0.012)                     -0.01
```

The first timing report shows the worst-case analysis results using fixed values derived from the quantiles of the variations. The reported slack is –0.19 time units.

The second timing report shows the variation-aware, path-based analysis of the critical path. In a path-based analysis with variation, PrimeTime VX retains all of the statistical delay, transition time, arrival time, and slack distribution information. As in conventional path-based analysis, pessimism is reduced by a more exact accounting of input conditions.

The slack is initially calculated by subtracting the quantile of the “data arrival time” from the quantile of the “data required time,” giving a result of –0.11 time units. However, this is not the same as subtracting the distributions of these two time values and then taking the quantile of the result. The latter method is the correct way to calculate the slack with variation. Accordingly, an additional “statistical adjustment” of 0.10 time units is applied to arrive at the correct statistical value. The final result is a reported slack of –0.01 time units at the q01 quantile of the slack distribution (refer to Figure 1-14 on page 1-17 for a graphical display of the slack distribution). The probability of failure (slack less than zero) is reported as 0.012, or 1.2 percent.

By default, the 0.00135 and 0.99865 quantile points of a distribution are considered the minimum and maximum values for reporting purposes, corresponding to the –3*sigma and +3*sigma points of a perfectly normal distribution. For example, the 0.00135 quantile point of a slack distribution is considered the worst-case slack. You can specify other quantiles with the `set_variation_quantile` command. For example, the command to use the 0.02 and 0.98 quantile points is:

```
pt_shell> set_variation_quantile \
          -quantile_low 0.02 -quantile_high 0.98
```

You can change the quantile settings and generate a new timing report without running a new timing update.

Variation and Crosstalk Analysis

If you have a PrimeTime SI license, you can perform variation-aware analysis and crosstalk analysis together. To determine the arrival windows for aggressor transitions, PrimeTime SI uses the worst-case variation-aware arrival times.

To calculate the worst delta delay and bump height for a crosstalk situation, PrimeTime SI considers the parameter variations of cells, parasitic data, pin capacitance, and aggressor slew. It finds the worst-case delta delay based on the distribution of each parameter and the impact of the parameter on overall coupled stage delay. Each delta delay value shifts the entire victim arrival distribution by the calculated amount. This shift can affect uncoupled variation-aware arrival windows downstream from the victim net.

In the current release, the PrimeTime SI composite aggressor feature is not compatible with variation-aware analysis.

2

Libraries and Interconnect Parameters

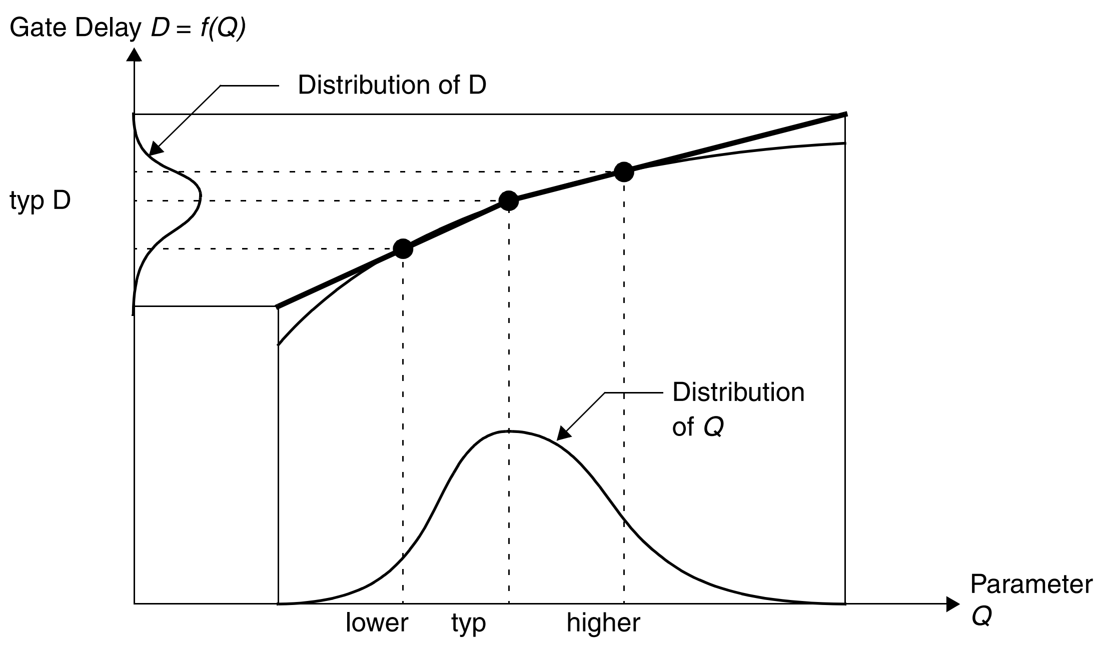
Variation-aware static timing analysis requires library cells that have been characterized at multiple points in the variation space. You need to load and link your design to the variation-aware library or libraries containing the variation data. To read in parasitic data with statistical variation, you must use the `-keep_variations` option of the `read_parasitics` command.

This chapter contains the following sections:

Variation-Aware Library Creation

A library used for analysis in PrimeTime VX specifies the timing behavior of cells at one or more variation data points. To determine the circuit behavior under a distribution of parameter values, the libraries need to characterize the behavior at two or three values of each variation parameter. Figure 2-1 shows an example. For more information about selecting characterization points, see “Handling Variation With Distribution Data” on page 1-6.

Figure 2-1 Delay as a Function of Parameter P, Three Data Points



For timing analysis in PrimeTime VX, you read in the libraries and specify the actual parameter values with the `create_variation` command and apply them to the design or to design objects with the `set_variation` command, as described in Chapter 3, “Variations and Correlations.” Given the distribution of the parameter, PrimeTime VX calculates the distribution of delays and slews continuously throughout the range of values, using interpolation and extrapolation of data surrounding the library-defined operating points.

During library characterization, you can create either a single unified library containing all of the process data points or a set of separate libraries, one for each variation data point. For a process technology defined to have N process parameters, variation modeling requires up to 2*N*+1 sets of characterization data to be read into PrimeTime VX. All this characterization data contained can be combined into a single unified variation-aware library.

The Liberty NCX characterization tool from Synopsys can create a single unified library and can also merge multiple existing libraries into a single unified library. Then you can read the unified library into PrimeTime VX as a single step. PrimeTime VX analysis results and runtime are essentially the same whether you use a single unified library or separate libraries. However, using a unified library is preferred because it is simpler and less prone to error. Unified libraries always use the compact CCS format, whereas separate libraries can use either the compact or expanded CCS format.

A unified library contains all the information on the characterized variation values. When you use a unified library, you only need to link in that library and then use the `create_variation` and `set_variation` commands to specify the distribution of parameter values for timing analysis. On the other hand, if you are using a set of 2*N*+1 separate libraries, you first link the nominal library as the main library and then use 2*N*+1 `set_variation_library` commands to establish the variation parameters, before you use the `create_variation` and `set_variation` commands.

If you are using 2*N*+1 separate variation libraries, you need to know the characterized parameter values for each library so that you can specify these same values when you use the libraries in PrimeTime VX. It is a good idea to name each library in a manner to suggest the operating point, for example, "lowL3.2.lib" and "highL3.8.lib."

Using Unified Libraries

Before you can perform a variation-aware analysis, you must read in the cell library data contained in the unified library. The commands used to read the library data depend on whether you are performing full-chip variation-aware analysis or conventional corner-case analysis followed by variation-aware, path-based analysis.

Full-Chip Variation-Aware Analysis

For full-chip variation-aware analysis using a single unified library, put the library in the link path as shown in the following example:

```
set link_path "* vaccs.db ... "
...
link_design

# Define variations
```

```
set_variation [create_variation -name len \
 -parameter_name len  -type normal -values {0 1}]
set_variation [create_variation -name vt \
 -parameter_name vt  -type normal -values {0 1}]
...

# Analysis
set variation_enable_analysis true
update_timing
```

You can read and use multiple unified variation-aware libraries containing different cells, as long as the libraries have the same variation parameters, characterized at the same values. For example, you might have one unified library named va_lib1.db containing high-Vt cells, and another library named va_lib2.db containing low-Vt cells. To use both unified libraries at the same time, put them both in the link path. For example,

```
set_link_path "* va_lib1.db va_lib2.db other_lib.db "
 ...
link_design

# Define variations
set_variation [create_variation -name len \
 -parameter_name len  -type normal -values {0 1}]
set_variation [create_variation -name vt \
 -parameter_name vt  -type normal -values {0 1}]
 ...

# Analysis
set variation_enable_analysis true
update_timing
```

Path-Based-Only Variation-Aware Analysis

Some analysis flows use conventional corner-based analysis for the whole chip, followed by variation-aware analysis of critical paths. This flow is invoked by setting the `variation_analysis_mode` variable to `path_based_only`. In this flow, you link the maximum library as the main library and set the minimum library relationship for corner-based analysis. After whole-chip analysis, you select the critical paths for a more accurate variation-aware, path-based analysis.

If you are using a single unified library for the statistical part of this analysis, put the maximum corner-case library in the link path. To read in the variation-aware library, use the `read_db` command. For example,

```
set link_path "* maxlib.db ... "
 ...
link_design
# Create the min library relationship
set_min_library -min_library minlib.db maxlib.db
```

```
 ...
# Corner-based analysis
update_timing
 ...
# For path-based analysis
# Read in the variation library
read_db vaccs.db

# Define variations
set_variation [create_variation -name len \
 -parameter_name len  -type normal -values {0 1}]
set_variation [create_variation -name vt \
 -parameter_name vt  -type normal -values {0 1}]
 ...
# Path-based variation-aware analysis
set variation_enable_analysis true
set variation_analysis_mode path_based_only
report_timing ... [get_recalculated_timing_paths ... \
 [get_timing_paths ...]]
```

The cells in both the CCS-based variation-aware library and the minimum library must match the cells in the maximum library with respect to cell names, pin names, and timing arcs. The `variation_enable_analysis` variable must be set to true prior to the `read_db` command.

If you are using multiple unified variation-aware libraries containing different sets of cells, put the corner-case libraries in the link path. Read in the variation-aware libraries as needed with the `set_variation_library -link_library` and `read_db` commands. For example,

```
set_link_path "* corner_1.db corner_2.db other_lib.db"
 ...
update_timing
 ...
set variation_enable_analysis true
 ...
set_variation_library -link_library corner_1.db
read_db va_lib1.db
set_variation_library -link_library corner_2.db
read_db va_lib2.db
 ...
```

The `variation_enable_analysis` variable must be set to true before the `set_variation_library` command is used.

The following is an alternative way to read in multiple variation libraries after `variation_enable_analysis` has been set to true:

```
set_variation_library -link_library {corner_1.db corner_2.db}
read_db va_lib1.db va_lib2.db
```

The library association is in corresponding order, as listed in the commands.

Using Separate (2N+1) Libraries

Before you can perform a variation-aware analysis, you must read in the cell library data contained in the set of 2*N*+1 separate libraries. The commands used to read the library data depend on whether you are performing full-chip variation-aware analysis or conventional corner-case analysis followed by variation-aware, path-based analysis.

Full-Chip Variation-Aware Analysis

If you are using a single set of 2*N*+1 separate libraries, you can first link the nominal library as the main library and then use 2*N*+1 `set_variation_library` commands to establish the variation parameters. For example,

```
set link_path "* nominal.db ... "
 ...
link_design

# set variation libraries
set_variation_library -parameter_names {len vt} \
 -values {0 0} nominal.db -reference_value
set_variation_library -parameter_names {len} \
 -values -1 lib_slen_neg1s_tiny.db
set_variation_library -parameter_names {len} \
 -values +1 lib_slen_pos1s_tiny.db
set_variation_library -parameter_names {vt} \
 -values -1 lib_svt_neg1s_tiny.db
set_variation_library -parameter_names {vt} \
 -values +1 lib_svt_pos1s_tiny.db
 ...
# Define variations
set_variation [create_variation -name len \
 -parameter_name len  -type normal -values {0 1}]
set_variation [create_variation -name vt \
 -parameter_name vt  -type normal -values {0 1}]
 ...
# Analysis
set variation_enable_analysis true
update_timing
```

The `set_variation_library` command with the `-reference_value` option specifies the reference or typical library on which the related libraries are based. This must be the first `set_variation_library` command in the session. The subsequent `set_variation_library` commands specify the libraries that have been characterized at surrounding parameter values. Each command specifies the names of the parameters, the corresponding parameter values at which the library was characterized, and the name of the library.

You can read and use multiple sets of variation-aware libraries containing different cells, as long as the libraries have the same variation parameters, characterized at the same values. For example, you might have one set of 2*N*+1 libraries containing high-Vt cells, and a similar set of libraries containing low-Vt cells. In that case, put the nominal libraries in the link path. Then, in each group of `set_variation_library` commands, use the `-link_library` option to specify the nominal library in the link path associated with the variation library set. For example,

```
set_link_path "* va_lib1_nom.db va_lib2_nom.db other_lib.db"
 ...
set_variation_library -parameter_names "len vth" -values "0 0" \
 va_lib1_nom.db -link_library va_lib1_nom.db -reference
set_variation_library -parameter_names "len vth" -values "1 0" \
 va_lib1_len_plus1.db
set_variation_library -parameter_names "len vth" -values "-1 0" \
 va_lib1_len_minus1.db
set_variation_library -parameter_names "len vth" -values "0 1" \
 va_lib1_vth_plus1.db
set_variation_library -parameter_names "len vth" -values "0 -1" \
 va_lib1_vth_minus1.db
 ...
set_variation_library -parameter_names "len vth" -values "0 0" \
 va_lib2_nom.db -link_library va_lib2_nom.db -reference
set_variation_library -parameter_names "len vth" -values "1 0" \
 va_lib2_ len_plus1.db
set_variation_library -parameter_names "len vth" -values "-1 0" \
 va_lib2_len_minus1.db
set_variation_library -parameter_names "len vth" -values "0 1" \
 va_lib2_vth_plus1.db
set_variation_library -parameter_names "len vth" -values "0 -1" \
 va_lib2_vth_minus1.db
```

In the foregoing example, there are two sets of variation libraries, `va_lib1_*.db` and `va_lib2_*.db`, and an unrelated library, `other_lib.db`. The `-link_library` option in the nominal-library `set_variation_library` command associates the current set of variation libraries to a library in the link path. The `set_variation_library` commands for a group must be together, with the nominal library specified first.

Path-Based-Only Variation-Aware Analysis

Some analysis flows use conventional corner-based analysis for the whole chip, followed by variation-aware analysis of critical paths. This flow is invoked by setting the `variation_analysis_mode` variable to `path_based_only`. In this flow, you link the maximum library as the main library and set the minimum library relationship for corner-based analysis. After whole-chip analysis, you select the critical paths for a more accurate variation-aware, path-based analysis.

If you are using a single set of 2*N*+1 separate libraries for the statistical part of the analysis, you use 2*N*+1 `set_variation_library` commands to establish the variation parameters. For example,

```
set link_path "* maxlib.db ... "
 ...
link_design
# Create the min library relationship
set_min_library -min_library minlib.db maxlib.db
 ...
# Corner-based analysis
update_timing
 ...
# For path-based analysis
# Set variation libraries
set_variation_library -parameter_names {len vt} \
 -values {0 0} nominal.db -reference_value
set_variation_library -parameter_names {len} \
 -values -1 lib_slen_neg1s_tiny.db
set_variation_library -parameter_names {len} \
 -values +1 lib_slen_pos1s_tiny.db
set_variation_library -parameter_names {vt} \
 -values -1 lib_svt_neg1s_tiny.db
set_variation_library -parameter_names {vt} \
 -values +1 lib_svt_pos1s_tiny.db
 ...
# Define variations
set_variation [create_variation -name len \
 -parameter_name len  -type normal -values {0 1}]
set_variation [create_variation -name vt \
 -parameter_name vt  -type normal -values {0 1}]
 ...
# Path-based variation-aware analysis
set variation_enable_analysis true
set variation_analysis_mode path_based_only
report_timing ... [get_recalculated_timing_paths ... \
 [get_timing_paths ...]]
```

If you are using multiple sets of 2*N*+1 variation-aware libraries containing different sets of cells, put the corner-case libraries in the link path. Read in the variation-aware libraries as needed with the `set_variation_library -link_library` and `read_db` commands. For example,

```
set_link_path "* corner_1.db corner_2.db other_lib.db"
 ...
set_variation_library -parameter_names "len vth" -values "0 0" \
 va_lib1_nom.db -link_library corner_1.db -reference
set_variation_library -parameter_names "len vth" -values "1 0" \
 va_lib1_len_plus1.db
set_variation_library -parameter_names "len vth" -values "-1 0" \
 va_lib1_len_minus1.db
```

```
set_variation_library -parameter_names "len vth" -values "0 1" \
 va_lib1_vth_plus1.db
set_variation_library -parameter_names "len vth" -values "0 -1" \
 va_lib1_vth_minus1.db
 ...
set_variation_library -parameter_names "len vth" -values "0 0" \
 va_lib2_nom.db -link_library corner_2.db -reference
set_variation_library -parameter_names "len vth" -values "1 0" \
 va_lib2_ len_plus1.db
set_variation_library -parameter_names "len vth" -values "-1 0" \
 va_lib2_len_minus1.db
set_variation_library -parameter_names "len vth" -values "0 1" \
 va_lib2_vth_plus1.db
set_variation_library -parameter_names "len vth" -values "0 -1" \
 va_lib2_vth_minus1.db
```

Interconnect Parameter Variation

PrimeTime VX accepts statistical interconnect data from Star-RXCT VX and performs timing analysis in the presence of variations in parameters such as metal width, metal thickness, and dielectric thickness. A statistical analysis more accurately predicts design violations resulting from metal layer mismatch and avoids the pessimism of worst-case corner analysis.

To read in parasitic data with variation sensitivity information generated by Star-RCXT VX, use the `-keep_variations` option in the `read_parasitics` command. You can either choose to create normal distributions for all interconnect variations, or you can explicitly specify the interconnect variations with the `create_variation` and `set_variation` commands.

Interconnect Parameter Calculations

Star-RCXT VX supports the extraction of interconnect resistivity variation, via resistance variation, and temperature variation. PrimeTime VX supports the reading and usage of the variation-aware resistivity and via resistance information from Star-RCXT VX. The temperature-dependent data used for timing analysis is based on the temperature condition set before `read_parasitics`.

In Star-RCXT VX, the interconnect capacitance at a given point is specified as follows:

$$C = C_0\left(1 + \sum_i c_i \Delta v_i\right)$$

where C_0 is the nominal capacitance, c_i is the sensitivity coefficient for capacitance with respect to a given variation parameter v_i, and Dv_i is the change in that variation parameter.

The interconnect resistance at a given point is specified as follows:

$$R = R_0\left(\frac{1 + \sum_i n_i \Delta v_i}{1 + \sum_j d_j \Delta v_j}\right)$$

where R_0 is the nominal resistance, n_i is the sensitivity coefficient for resistance with respect to a given process variation v_i that affects the resistance in the numerator, Dv_i is the change in that variation parameter, d_j is the sensitivity coefficient for resistance with respect to a given process variation v_j that affects the resistance in the denominator, and Dv_j is the change in that variation parameter.

The interconnect resistance can also depend on temperature as follows:

$$R = R_0(1 + a\Delta T + b(\Delta T)^2)$$

where R_0 is the nominal resistance, *a* and *b* are the temperature sensitivity coefficients for resistance, and *DT* is the change in temperature.

PrimeTime VX supports the reading of parasitic data files to recognize the resistance variation term n_iDv_i, thereby fully taking into account the resistivity and via resistance variation information extracted by Star-RCXT. For static timing analysis, PrimeTime VX considers the interconnect capacitance and resistance sensitivity to the defined variations.

PrimeTime VX uses the temperature condition set with the `set_operating_conditions` command prior to `read_parasitics` and reads in the variation-dependent resistance data for that temperature only. If you later write out the parasitic data with the `write_parasitics` or `write_ilm_parasitics` command, PrimeTime VX writes the data as a temperature corner, using the temperature value in effect at the time that `read_parasitics` was used.

Statistical Path-Based Analysis with Worst RC

The use of worst RC for path-based analysis is specified by entering:

```
set variation_pba_use_worst_parasitics TRUE|FALSE
```

The default value of this variable is `false`; to enable statistical path-based analysis with worst parasitics, set this variable to `true`.

Note:

You must use the `-path_type full_clock_expanded` option for getting the correct worst RC vector for the complete path. This is mandatory because the intent is to find the parasitic corner configuration which gives the worst slack and not the worst data path alone.

The worst corner is evaluated only for parameters which have variation set on them. If `set_parasitic_corner` is used for certain parameters without any variation applied to the same parameters, worst RC mode is not calculated for the worst corner for those parameters.

This statistical path-based analysis is carried out using an automatic two-pass approach inside path recalculation when the worst RC feature is enabled.

The two passes are carried out as follows:

- PrimeTime VX first performs an initial path recalculation with no device variations to get the worst RC vector for the interconnect parameters.
- PrimeTime VX then performs a regular path recalculation with the interconnect parameters fixed at worst vector, and with the normal statistical variations for any device parameters.

This feature only affects path-based analysis results (`-pba_mode` path or `-pba_mode` exhaustive). The unrecalculated graph timing will still continue to use any statistical interconnect variation behaviors which have been defined. It is therefore possible for the recalculated worst RC slack to be worse than the graph slack if additional margin is not added to the graph to bound the worst RC calculation.

Reading Interconnect Parameters With Variation

Star-RCXT VX can generate parasitic data files that contain the variation sensitivity information each element. PrimeTime VX can read and use this information for variation-aware analysis. The recommended format for variation-aware parasitic data is the Synopsys Binary Parasitic Format (SBPF). However, SPEF data with statistical variation is also supported.

To read in interconnect parameter data with variation sensitivity information, use the `-keep_variations` option in the `read_parasitics` command. For example,

```
pt_shell> read_parasitics -keep_variations \
          -create_default_variations \
          -format SBPF my_chip.sbpf
```

The `-create_default_variations` option creates normal distributions for all interconnect variations using the mean and sigma values taken from in the parasitic data file.

As well as normal distributions, it is possible to specify the creation of uniform and discrete distributions. This is done using the `parasitic_variation_default_type` variable.

- In normal distribution (the default), a Gaussian distribution is created for the parasitic variation parameters.
- In uniform distribution, a uniform distribution from mean-abs(3*sigma) to mean+abs(3*sigma) is created.
- In discrete distribution, a 2-value distribution for the parasitic variation parameters is created. The two values used are mean+abs(3*sigma) and mean-abs(3*sigma), each with equal probability of 0.5.

To set the interconnect variations explicitly in PrimeTime VX, do not use the `-create_default_variations` option. Instead, use the `create_variation` and `set_variation` options. For example,

```
pt_shell> set var1 [create_variation -type normal \
          -values {0.02 0.0008}
          -parameter_name METAL1_WIDTH]

pt_shell> set_variation $var1
```

To report annotated parasitic data and show the interconnect parameter variation information, use the `-variation` option of the `report_annotated_parasitics` command. For example,

```
pt_shell> report_annotated_parasitics -variation
 ...

  ParamID      Name                       Stddev
  ------------------------------------------------------
      0        PO_T                       0.0247
      1        PO_W                       0.0175
      2        M1_T                       0.0533
      3        M1_W                       0.028
      4        M12_T                      0.036571
    ...
```

If you write the annotated parasitics back out using `write_parasitics` or `write_ilm_parasitics`, the statistical information can be written as SPEF or SBPF.

Interconnect Parameters With Temperature Variation

Star-RCXT VX can extract variation parasitics information with temperature as a variation parameter. In this flow, extraction is performed around a single extraction temperature (the “global temperature” in the Star-RCXT ITF file) but with additional sensitivity information so that the RC effects of temperature can be predicted across a temperature range. PrimeTime VX supports the use of this temperature variation information in its analysis.

When you run `read_parasitics -keep_variations` and the parasitics file has temperature variation information, the parasitic information is remapped to the global analysis temperature specified by `set_operating_conditions`, even if this temperature is different from the global temperature used for extraction by Star-RCXT. If the `-keep_variations` option is not used, the parasitics information is fixed at the global extraction temperature. If desired, temperature scaling can also be performed for a non-VX PrimeTime analysis by using the `-keep_variations` option. However, note that a PrimeTime VX license is needed to perform the conversion.

In all cases, the parasitic remapping to the analysis temperature is performed only during `read_parasitics -keep_variations`. After this conversion is done, the temperature sensitivity information is no longer kept in memory. Therefore, the global operating condition should be defined before the parasitics are read in. Any subsequent changes in operating condition temperature do not affect in the in-memory parasitics.

Interconnect Parasitic Corners

A parasitic data file generated by Star-RCXT VX can serve for both variation-aware analysis and conventional corner-case analysis. You can use the `set_parasitic_corner` command to establish the default values of the interconnect data used for analysis.

With variation analysis enabled (`variation_enable_analysis` set to `true`), the `set_parasitic_corner` command establishes the default behavior for any interconnect parameter that has no variation. You can override the default behavior by using the `-create_default_variations` option of the `read_parasitics` command or by creating interconnect variations with the `create_variation` and `set_variation` commands. For conventional corner-case analysis with variation analysis disabled and for the conventional corner-case portion of analysis in the path-based-only mode, the `set_parasitic_corner` command establishes the fixed interconnect values used for analysis.

The command works in conjunction with a text file that specifies a set of variation ratios used to calculate the interconnect parasitic values. For example,

```
pt_shell> set_parasitic_corner -name CMAX my_corner_file
```

You need to provide a simple ASCII text file containing the corner definitions as a set of variation ratios. For example:

```
CORNER_NAME: CMAX
     M1_W        1.2
     M2_T        0.4
     ILD_c_T    -2.0
CORNER_NAME: RCMAX
 ## my definition of RCMAX
     M1_W        0.25
     M2_T        0.16
     ILD_c_T    -0.22
CORNER_NAME: RCMIN
     M1_W        -0.23
     M2_T        -0.11
     ILD_c_T      0.04
CORNER_NAME: CMIN
     M1_W        -0.18
     M2_T        -0.14
     ILD_c_T      0.06
```

This example defines four sets of corner parasitics called CMAX, RCMAX, RCMIN, and CMIN. For the CMAX corner, the M1_W parameter is changed in the positive direction from nominal (+1.2 multiplied by the M1_W variation coefficient), the M2_T parameter is changed in the positive direction from nominal (+0.4 multiplied by the M2_T variation coefficient), and the ILD_c_T parameter is changed in the negative direction from nominal (–2.0 multiplied by the ILD_c_T variation coefficient). The variation coefficients are contained within the sensitivity parasitics file.

The general form of the file is:

```
CORNER_NAME: name
  param_name  param_variation_ratio
  param_name  param_variation_ratio
  ...
CORNER_NAME: name
  param_name  param_variation_ratio
  param_name  param_variation_ratio
  ...
```

The file should contain at least one corner definition. A corner definition can specify any number of parameter names (including zero names) and a corresponding *param_variation_ratio* value for each parameter. PrimeTime VX multiplies the *param_variation_ratio* value by the variation coefficient specified for the parameter in the parasitic data file, and then multiplies that result by the nominal parameter value to get the change in value used for analysis.

To report parasitic corner information that has been set, use the command `report_annotated_parasitics -variation`. To report the parasitic sensitivity of nets, use the `report_variation` command. For details, see Chapter 3, "Variations and Correlations."

To remove parasitic corner information, use the `remove_parasitic_corner` command.

3

Variations and Correlations

PrimeTime VX determines the distributions of delays, slews, and slacks based on the variations of selected parameters. The actual distributions of parameters are specified in PrimeTime VX with the `create_variation` and `set_variation` commands.

This chapter contains the following sections:

Parameter Variation and Correlation

The cell libraries specify the dependency of cell delays and slews on parameters, but not the probability distributions of those parameters. You specify the distributions in PrimeTime VX by using the following commands:

- The `create_variation` command creates a variation object that is associated with a given characterized parameter. The variation object specifies the type of statistical distribution (normal, piecewise-linear, empirical, uniform, lognormal, or discrete) and the values associated with that distribution (for example, the mean and standard deviation of a normal distribution).
- The `set_variation` command assigns a given variation object to the current design or some objects within the current design, resulting in the specified statistical distribution of the characterized parameter in the design.
- The `create_correlation` command creates a correlation relationship object. For auto-correlation between two instances of a variation, the correlation can be set to 0.0 (uncorrelated), 1.0 (fully correlated), or some number between 0.0 and 1.0 (partially correlated).
- The `set_variation_correlation` command assigns a correlation object to a list of variations or to all variations, resulting in the specified correlation relationship between different instances of a variation.

This organization of parameter variation data is flexible and efficient. You can easily change the parameter values, distributions, and correlation characteristics to analyze different analysis conditions, without changing the cell libraries.

Creating and Setting Variations

To perform variation-aware analysis, you need to specify the statistical variation of the parameters that affect the timing characteristics of the circuit. To specify the variation of a parameter, you create a variation object with the `create_variation` command and then apply the variation to the design with the `set_variation` command.

This is the syntax of the two commands:

```
create_variation
  [-name variation_name]
  [-parameter_name parameter_name]
  -type distribution_type
  -values values_list
  [-lower_bound lower_bound]
  [-upper_bound upper_bound]
```

```
set_variation
  variation_list
  [object_list]
```

For example, the following commands load a variation library having parameters “len” and “vth,” create a variation named “Lfab” for parameter “len,” and set the variation on the current design:

```
pt_shell> set_variation_library \
          -parameter_names "len vth" \
          -values "100 0.2394" maxvth.db

pt_shell> create_variation -parameter_name len \
          -name Lfab -type normal -values {0 1}

pt_shell> set_variation [get_variations Lfab]
```

create_variation

The `create_variation` command creates a new variation object and returns a collection containing that variation. You can specify a variation name with the `-name` option, allowing you to reference that variation later with the `get_variations` command. Any string that has not already been used can be specified as the variation name.

The `-parameter_name` option specifies the process parameter associated with the variation. The specified parameter must be defined with the `set_variation_library` command or defined in the parasitic data file read in with the `read_parasitics` command. Multiple variations can be associated with one parameter.

If the `-parameter_name` option is not used, the variation is not associated with any parameter and has no direct effect on the design. However, it can still be used by the variation commands such as `add_variation`, `sub_variation`, `max_variation`, and `min_variation`.

The `-type` and `-values` settings specify the statistical distribution of the variation. Table 3-1 shows the types of distributions you can specify and the values associated with each type.

Table 3-1 Distribution Types

Distribution Type	**Values**
`normal`	{*mean sigma*}
`pwl`	{*x1 f1 x2 f2 x3 f3 ...*}
`empirical`	{*x1 x2 x3 ...*}
`constant`	{*a b*}
`uniform`	{*a b*}
`lognormal`	{*mean sigma*}
`discrete`	{*x1 p1 x2 p2 x3 p3 ...*}

You can optionally specify upper and lower bounds by using the `-lower_bound` and `-upper_bound` options.

To remove a variation that you no longer want, use the `remove_variation` command. This is the command syntax:

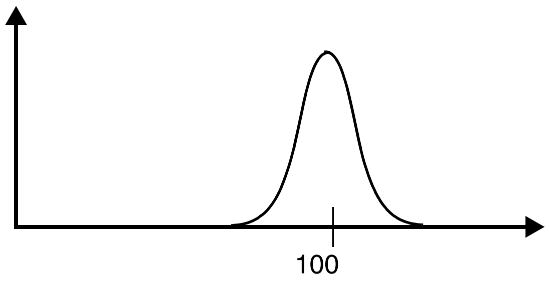
```
remove_variation
  -all | variation_list
```

Normal Distribution

The following command creates a normal (Gaussian) distribution with a mean value of 100 and a sigma (standard deviation) of 3.34:

```
pt_shell> create_variation -name test_normal \
          -type normal -values {100 3.34}
```

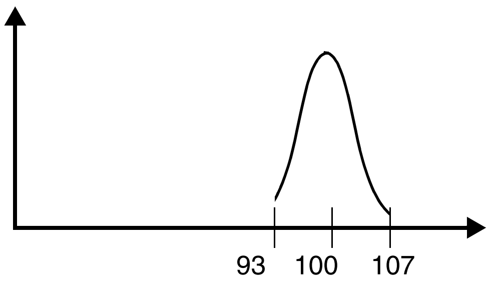
Figure 3-1 PDF Plot of Normal Distribution



By default, the distribution is unbounded at both ends. To truncate the distribution at one end or both ends, use one or both of the options `-lower_bound` and `-upper_bound`. For example,

```
pt_shell> create_variation -name test_normal \
          -type normal -values {100 3.34} \
          -lower_bound 93.0 -upper_bound 107.0
```

Figure 3-2 PDF Plot of Truncated Normal Distribution

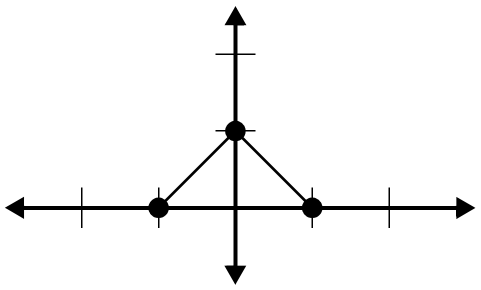


Piecewise-Linear Distribution

The following command creates a piecewise-linear distribution. The pairs of values in the value list specify the data points in the distribution.

```
pt_shell> create_variation -name test_pwl \
          -type pwl -values {-1.0 0.0  0.0 1.0  1.0 0.0}
```

Figure 3-3 PDF Plot of Piecewise-Linear Distribution



This particular example is the triangular distribution.

The piecewise-linear distribution has *n* pairs of numbers x_i , f_i . The sequence of values x_i must be monotonically increasing and all the values f_i must be positive or zero. The first and last values f_1 and f_n must be zero. The area under the curve must be 1.0 because it is a probability distribution function.

Empirical Distribution

The following command creates an empirical distribution. The sequential list of values (possibly generated by simulation or other empirical means) specifies the distribution. The likelihood that each sample value will occur is 1.0 divided by the sample size.

```
pt_shell> create_variation -name test_empirical \
          -type empirical \
          -values {0.0 91.4 96.3 99.5 102.1 90.0}
```

Constant Distribution

The following command creates a constant distribution that models the case where there is no statistical variation. The value is a single number. This distribution is the same as a discrete distribution having a single value with a probability 1.0. The constant distribution can be used to mathematically shift another distribution by a fixed amount, using the commands described in "Math Operations on Variations" on page 3-15.

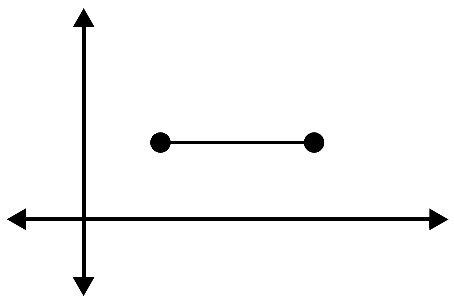
```
pt_shell> create_variation -name test_const \
          -type constant -values {2.0}
```

Uniform Distribution

The following command creates a uniform distribution. A single pair of values specifies the minimum value *a* and maximum value *b* of the range. The probability density function has a constant value equal to 1/(*b*–*a*).

```
pt_shell> create_variation -name test_uniform \
          -type uniform -values { 1.0  3.0 }
```

Figure 3-4 PDF Plot of Uniform Distribution



Lognormal Distribution

The following command creates a lognormal distribution. The first value is *mu* (m) and the second value is *sigma* (s).

```
pt_shell> create_variation -name test_lognorm \
          -type lognormal -values { 0.0  0.5 }
```

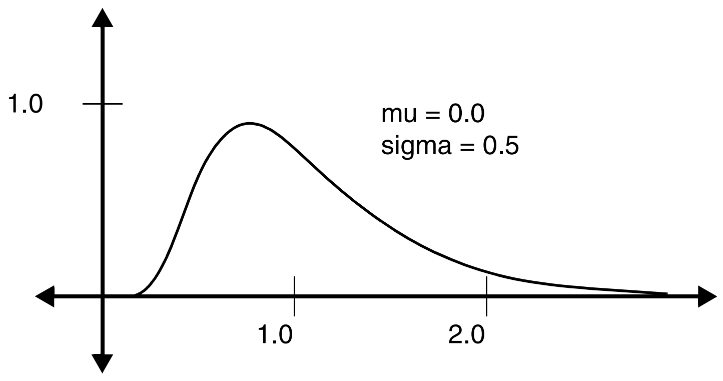


Figure 3-5 PDF Plot of Lognormal Distribution

A random variable *X* has a lognormal distribution with parameters m and s if *log*(*X*) has a normal distribution with mean m and standard deviation s. The lognormal distribution has support over the range (0.0, infinity).

The lognormal distribution has a probability density function

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(\ln x - \mu)^2 / (2\sigma^2)}$$

for *x* greater than zero.

By default, the distribution is unbounded at the upper end. To truncate the upper range, use the `-upper_bound` option. For example,

```
pt_shell> create_variation -name test_lognorm \
          -type lognormal -values { 0.0  0.5 } \
          -upper_bound 2.5
```

Discrete Distribution

The following command creates a finite discrete distribution. The list contains a sequence of paired numbers. In each pair of numbers x_i and p_i , the first number x_i is an outcome of the variation and p_i is the probability of that outcome. The sequence of numbers x_i must be monotonically increasing. The numbers p_1 through p_n must be strictly positive and must add up to 1.0.

```
pt_shell> create_variation -name test_discr \
      -values { 0.5 0.25  1.0 0.40  1.5 0.15  2.0 0.20 }
```

set_variation

The `set_variation` command applies one or more variation objects to timing objects in the design or to the whole design. Each variation object specified in the command must have been created previously with the `create_variation` command using the `-parameter_name` option. Applying a variation object on a timing object affects the timing behavior of that object.

This is the syntax of the command:

```
set_variation
  variation_list
  [design_object_list]
```

You specify a single variation, a list of variations, or a collection of variation to apply, and optionally a list of objects in the design on which to apply the variations, such as a collection cell arcs. If no object list is specified, the variations apply to the whole design, affecting all occurrences of the associated parameter.

To remove a variation from an object in the design or the whole design, use the `reset_variation` command. This is the command syntax:

```
reset_variation
  [-all]
  [variation_list]
  [design_object_list]
```

Creating and Setting Correlations

Different instances of variations can have different levels of correlation. Some variations may be highly correlated, while other are not correlated at all. For the best possible analysis accuracy, you should specify the correlation relationships between variations with the `create_correlation` and `set_variation_correlation` commands. This is the syntax of the commands:

```
create_correlation
  -name string
  -constant float | -cross_correlations float_list |
    -physical_distance float_list

set_variation_correlation
  -name variation_correlation_name
  -correlation correlation_name
  -all | variation_object_list
```

The `create_correlation` command creates a correlation object. It assigns a name to the correlation and specifies a floating-point value or a set of such values, depending on the correlation type.

There are three types of correlation:

- constant correlation (also known as auto-correlation) – the correlation between different instances of the same library variation or parasitic data variation
- cross-correlation – the correlation between different variations within a given instance
- spatial correlation – the correlation between device variations that depend on the physical distance between devices within a single die.

You must specify the correlation type in the `create_correlation` command using one of the following: `-constant`, `-cross_correlations`, or `-physical_distance`.

The `set_variation_correlation` command applies a named correlation object to one or more variations previously defined by the `create_variation` command. The `-name` string is an arbitrary name that you assign to the variation correlation assignment. The `-correlation` string is the name of the correlation object previously given in the `create_correlation` command. You must either specify a list of variation objects on which to apply the correlation or use the `-all` option to apply it to all variations.

For example, the following commands load a variation library having parameters “len” and “vth,” create variations named “Lfab1” and “Lfab2” for the two parameters, create a full correction object, and apply that correlation object to the two variations:

```
pt_shell> set_variation_library \
          -parameter_names "len vth" \
          -values "100 0.2394" len_vth.db

pt_shell> create_variation -parameter_name len \
          -name Lfab1 -type normal -values {0 1}

pt_shell> set_variation [get_variations Lfab1]

pt_shell> create_variation -parameter_name vth \
          -name Lfab2 -type normal -values {0 1}

pt_shell> set_variation [get_variations Lfab2]

pt_shell> create_correlation -name full_corr \
          -constant 1.0

pt_shell> set_variation_correlation -name device_len \
          -correlation full_corr [get_variations Lfab*]
```

Auto-Correlation

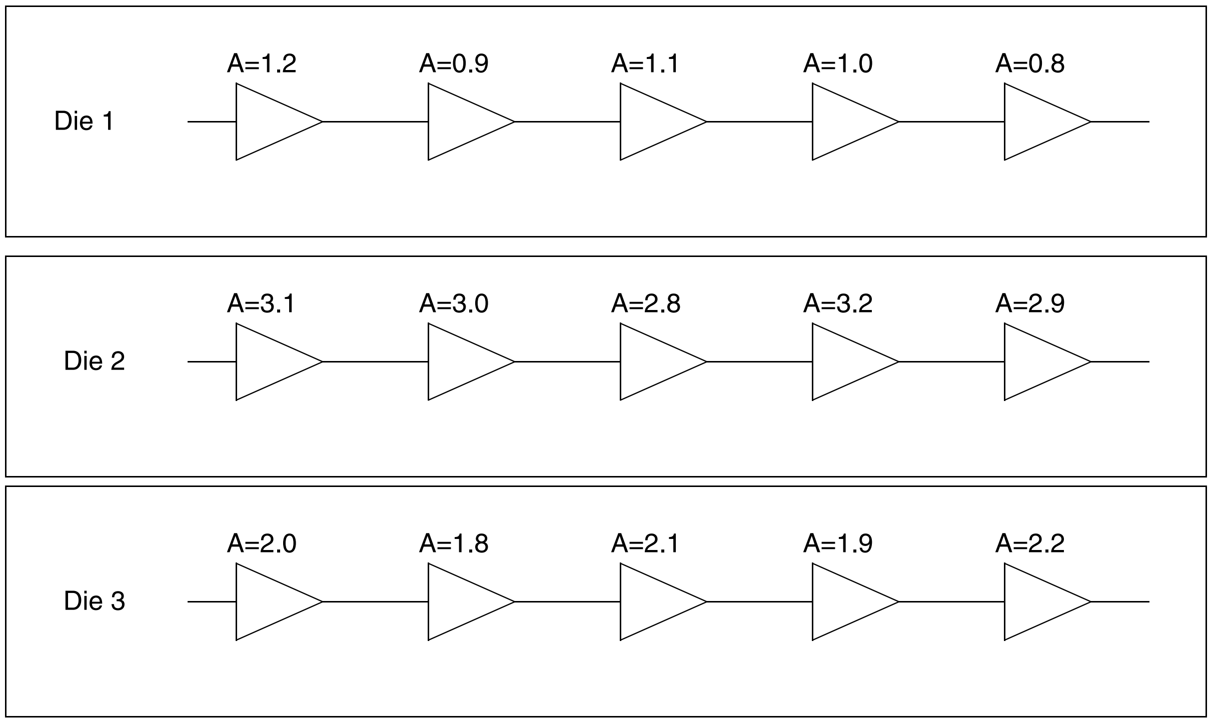
Auto-correlation is the correlation between different instances of the same variation. You specify the correlation value by using the `create_correlation` command with the `-constant` option. For example,

```
pt_shell> create_correlation -name full_corr \
          -constant 1.0
```

The constant is the correlation value between two instances of the variation. A value of 0.0 means that different instances of the variation within a die are statistically independent. This correlation value is recommended for modeling within-die, on-chip variation because such random variations are uncorrelated. A value of 1.0 means that different instances of the variation are all identical within a die. This correlation value is recommended for modeling die-to-die variations because such variations are highly correlated within a die.

Within-die and die-to-die variations are illustrated in Figure 3-6. In each die is a string of buffer cells, each having a parameter "A" that varies from cell to cell and from die to die. Within a given die, the parameter has small random variations from cell to cell. There is also a larger variation from die to die, but these variations are highly correlated; all of the cells within a given die tend to have the same higher or lower values.

Figure 3-6 Within-Die and Die-to-Die Variations



This type of variation can be modeled as two components. One component is the small, uncorrelated within-die variation. The other component is the larger, correlated die-to-die variation.

A parameter can also be partially correlated, having a correlation value between 0.0 and 1.0. For a partial correlation, the variation is assumed to have a normal distribution and that two instances of the variation have a bivariate normal distribution with a correlation equal to the specified constant value. If a correlation is not specified on a variation, it is assumed to be a constant correlation with a value of 1.0

Cross-Correlation

Cross-correlation is the correlation between different variations applied to the same timing object. To specify a cross-correlation, you first specify the partial correlation properties by using the `create_correlation` command with the `-cross_correlation` option. Then you apply the correlation object to the parameter variations. For example,

```
pt_shell> create_correlation -cross_correlations \
          {-0.85 0.45 0.09} -name xcor_A_B_C

pt_shell> set_variation_correlation -name xcor \
          -correlation xcor_A_B_C {A B C}
```

The variations applied to each device or parasitic variation are assumed to be jointly normal. The values specified with the `-cross_correlations` correspond to the number of parameters in the lower triangular correlation matrix. If (A, B, C, D) are four variations, the sequential list of values corresponds to the correlations

{ *corr(B,A) corr(C,A) corr(C,B) corr(D,A) corr(D,B) corr(D,C)* }

The number of cross-correlation terms entered in the command must correspond to the number of variations to which the correlation object is applied; *N* parameters require *N(N-1) / 2* coefficients. In addition, the matrix must be positive definite, which implies that the set of variations is the minimal set, without a linear relationship between any members of the set, and which cannot be reduced to a lower number of primitive variations. Joint normality of a vector of variations implies normality of each variation.

You can define multiple sets of cross-correlated variations. However, a given variation can belong to no more than one cross-correlated set. All parameters within one cross-correlation set must have the same auto-correlation value, either all constant 0.0 or all constant 1.0.

Certain interconnect parasitic parameters may tend to vary together on some processes; some of them may be positively correlated (increase in one parameter leads to an increase in the other parameter) and some of them may be negatively correlated (increase in one parameter leads to a decrease in the other parameter). In addition, by correlating some types of parameters together (like the width and thickness of a given layer), the statistical interconnect analysis can more efficiently explore the interesting timing corner cases.

PrimeTime VX supports `-1` and `+1` cross-correlation of *k*-value random parameters, thereby improving the accuracy of variation-aware timing analysis through a modeling of the subtleties of the semiconductor manufacturing process.

The command usage is as follows:

```
create_variation  [-name variation_x] [-parameter_name parameter_x]
        -type discrete -values  "x1 1/k x2 1/k x3 1/k ... xk 1/k"

create_variation  [-name variation_y] [-parameter_name parameter_y]
        -type discrete -values  "y1 1/k y2 1/k y3 1/k ... yk 1/k"

create_correlation -name  cross_corr -cross_correlations {-1}

set_variation_correlation -name cross_corr_xy -correlation cross_corr
{variation_x variation_y}
```

When this feature is used, it can result in better accuracy, as the behavior of the interconnect parasitic parameters during the manufacturing process is accounted for with the cross-correlation specifications. There is no detrimental impact on the run-time performance of PrimeTime VX.

Conversely, there is no quality of results impact on default PrimeTime VX performance and capacity when this feature is not used.

Spatial Correlation

Spatial correlation is correlation between device variations that is a function of the physical distance between the objects on the die. This within-die correlation applies only to device variations only, not to parasitic data variations.

Spatial correlation is described by a correlation function $r(d)$. For a variation $X(u)$ at the physical location u, $r(d) = Corr(X(u), X(u+d))$. The position u is in a two-dimensional plane and d is the Euclidean distance separating the two devices. This function has a value of 1.0 at $d = 0.0$ and typically decreases and tapers off to 0.0 as d becomes larger.

It is assumed that the correlation function is isotropic, or in other words, the correlation between two devices depends only on the distance that separates them, not on their respective positions. Under this assumption, the correlation function is symmetric and independent of the X and Y directions.

The correlation function must be estimated by the user. You enter the correlation information as a list of points: (d_1 r_1 d_2 r_2 ... d_n r_n) that approximate the function. The distances *d* must be greater than 0.0 and must be in increasing order, and the values r_1, r_2, ... r_n must be between –1.0 and +1.0. The variations are assumed to have a normal distribution, or in other words, for all integers *n* and all locations $u_1, u_2, ..., u_n$, the vector $(X(u_1),X(u_2),...,X(u_n))$ has a multivariate distribution.

The physical location information must be specified in the SBPF data files. The variable `read_parasitics_load_locations` must be set to true prior to reading in the parasitics. You enter the correlation information by using the `create_correlation` command with the `-physical_distance` option, with distance units in nanometers. For example, the following script creates a spatial correlation named "c_spatial" and applies it to a variation called "len" in the SPEF data file sptl_corr.spef.

```
# Create device variation
set_variation_library -parameter_names {len} \
 -values {100} lib_len_nom.db -reference_value
set_variation_library -parameter_names {len} -values 95 lib_len_small.db
set_variation_library -parameter_names {len} -values 104 lib_len_large.db

# It has the normal distribution
set_variation [create_variation -name len -parameter_name len \
-type normal -values {0 3.0 }]

# Set variable so physical information is read in
set read_parasitics_load_locations "true"

# SBPF file contains location information
read_parasitics -format SBPF sptl_corr.sbpf -keep_variation
create_correlation -name c_spatial \
 -physical_distance { 500000.0 0.9 1000000.0 0.7 \
                      2000000.0 0.4 4000000.0 0.1 }
# len now has a spatial within-die correlation
set_variation_correlation -name s_spatial -correlation c_spatial  {len}
```

Unknown Variation

Unknown variation allows PrimeTime VX to model uncertain variation effects. It gives you the flexibility to model any combination of inter-die, intra-die, or instance-specific uncertain effects. This feature allows you to estimate timing uncertainty in the variation domain as an alternative to conventional derate-based methods.

Unknown variation also enables the full Statistical Static Timing Analysis (SSTA) flow in PrimeTime VX, without the need for statistical libraries or statistical parasitic files. It allows you to exercise PrimeTime VX and familiarize yourself with the SSTA flow before spending time and effort on statistical library characterization or parasitic extraction.

PrimeTime VX supports both global and random variations on unknown parameters, as outlined in the following sections.

Setting Global Variation of Unknown Parameter

The `create_variation` command is used to create an unknown variation parameter. This leverages the power of this command to support any kind of distribution that you desire.

The magnitude of an unknown variation parameter is interpreted as a percentage of the library value. The following example creates a Gaussian unknown distribution with 0 mean (centered around the library value) and standard deviation of 0.01 (1 percent of mean value). Since this is an unknown variation, the distribution is applied directly to the underated nominal delay of a cell-timing arc to which the variation is applied.

```
create_variation -unknown_type \
      -name global_unknown_01 \
      -type normal \
      -values {0 0.01}
```

PrimeTime VX specifies that it is a global variation (autocorrelation=`1`) and applies it to the design, library cells, or instances:

```
create_correlation -name global \
      -constant 1
set_variation_correlation -name cgu01 \
      -correlation global [get_variations global_unknown_01]

set gu01 [get_variations global_unknown_01]

# set "unknown_type" variation on the design
set_variation $gu01
```

Setting Random Variation of Unknown Parameter

The random variation is set in a similar manner to the global unknown variation:

```
create_variation -unknown_type \
      -name random_unknown_01 \
      -type normal \
      -values {0 0.01}
```

PrimeTime VX specifies that it is a random variation (autocorrelation=`0`) and applies it to the design, library cells, or instances:

```
create_correlation -name random \
      -constant 0
set_variation_correlation -name cru01 \
     -correlation random [get_variations random_unknown_01]

set ru01 [get_variations random_unknown_01]

# set "unknown_type" variation on the design
set_variation $ru01
```

Math Operations on Variations

You can perform various types of mathematical operations with variations, including addition, subtraction, minimum/maximum value, and random number generation.

The following PrimeTime VX commands perform these operations:

```
add_variation var_object_list
sub_variation var_object_list
max_variation var_object_list
min_variation var_object_list
get_random_numbers
  -sample_size num_points
  -seed seed_value
  variation_object
```

The `add_variation` command statistically adds two or more variations, producing a new variation. The object list specified in the command must be a collection of two or more variations. Note that adding two variations, then adding the resulting sum to a third variation, can produce a different result from adding the three variations together at the same time.

The `sub_variation` command statistically subtracts one variation from another. The object list specified in the command must be a collection of two variations.

The `max_variation` command returns a variation that is the statistical maximum of the variations in the specified collection. The result can be different from any of the original variations in the specified collection. The `min_variation` command works in a similar manner, returning a new variation that is the minimum of the variations in the collection.

The `get_random_numbers` command generates a list of random numbers from a specified variation object previously created with the `create_variation` command. The `-sample_size` parameter specifies the desired number of data points. The `-seed` setting is an integer used to initialize the pseudo-random number generator, allowing you to generate the same sequence or a different sequence of numbers each time.

The following command creates a variation object named v1 having a normal distribution with a mean value of 15.0 and a standard deviation of 2.0:

```
pt_shell> create_variation -name v1 -type normal \
          -values { 15.0 2.0 }]
```

The following command creates a variation object named v2 having a normal distribution with a mean value of 10.0 and a standard deviation of 5.0:

```
pt_shell> create_variation -name v2 -type normal \
          -values { 10.0 5.0}]
```

The following command creates a collection containing the variations v1 and v2. It sets the variable c1 to the new collection:

```
pt_shell> set c1 [add_to_collection $v1 $v2]
```

The following command sets the random variable a1 to the sum of v1 and v2 (previously collected in c1):

```
pt_shell> set a1 [add_variation $c1]
```

The following command subtracts v2 from v1 and sets the variable s1 to the result:

```
pt_shell> set s1 [sub_variation $c1]
```

The following command sets the variable mx to the larger of v1 or v2 (previously collected in c1):

```
pt_shell> set mx [max_variation $c1]
```

The following command sets the variable mn to the smaller of v1 or v2 (previously collected in c1):

```
pt_shell> set mn [min_variation $c1]
```

The following command generates a statistically independent sequence of 1,000 floating-point numbers having the probability distribution previously defined in the variation v1. The variable rnd_lst is set to the list of generated numbers. The seed value is set to 72, so the same sequence can be generated later by specifying the same seed value.

```
pt_shell> set rnd_lst [get_random_numbers \
          -sample_size 1000 -seed 72 \
          [get_variations v1]]
```

Derating Statistical Distributions

The variation-aware features of PrimeTime VX can accurately account for the effects of known variations. You can use derating to adjust the variation effects upward or downward, for example, to model the additional effects of unknown variations, or to model variation effects for macrocells or IP cells for which there is no variation library data available.

The `set_timing_derate` command adjusts the calculated delays of the paths in the whole design or in specified library cells or cell instances. You can get a more conservative analysis by making long path delays longer and short path delays shorter, or get a less conservative analysis by doing the opposite.

This is the syntax of the command:

```
set_timing_derate
 [-early] [-late]
 [-rise] [-fall]
 [-clock] [-data]
 [-net_delay] [-cell_delay] [-cell_check]
 [-static] [-dynamic]
 [-scalar] [-variation]
 value
 [object_list]
```

It may be desirable to use larger derating factors for corner-case delays, and to use smaller factors for statistical delays that already account for the effects of known variations. For this reason, there are separate derating settings for corner-case and statistical delays.

For statistical derating, use the `-variation` option; or for corner-case derating, use the `-scalar` option. For example,

```
pt_shell> set_timing_derate -scalar -early 0.9
pt_shell> set_timing_derate -scalar -late 1.2
pt_shell> set_timing_derate -variation -early 0.96
pt_shell> set_timing_derate -variation -late 1.08
```

The early and late scalar derating factors 0.9 and 1.2 apply to corner-case timing and to cells or nets that do not have variation data, such as macrocells, pad cells, or other cells without variation models. Early (minimum) path delays are decreased by 10 percent and late (maximum) path delays are increased by 20 percent. This causes a more conservative analysis than leaving delays at their original calculated values.

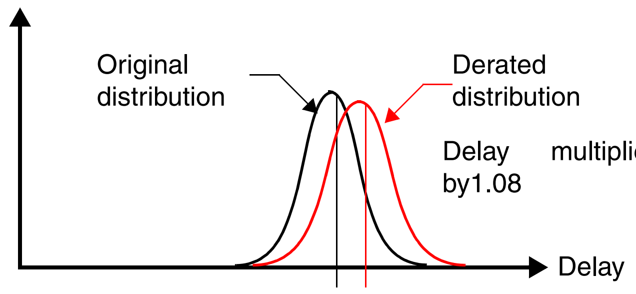
The early and late variation derating factors 0.96 and 1.08 apply to statistical delays, including cell arcs of cell instances with variation data and net arcs of nets with interconnect variation data. Early (minimum) path delay distributions are decreased by 4 percent and late (maximum) path delay distributions are increased by 8 percent.

Derated delays are adjusted according to the following formula:

new_delay = *old_delay* * *derating_factor*

For statistical delay variations, PrimeTime VX multiplies the derating factor by the whole distribution. This causes the mean value, standard deviation, and peak value of the distribution to change, as shown by the example in Figure 3-7.

Figure 3-7 Derated Maximum-Delay Distribution

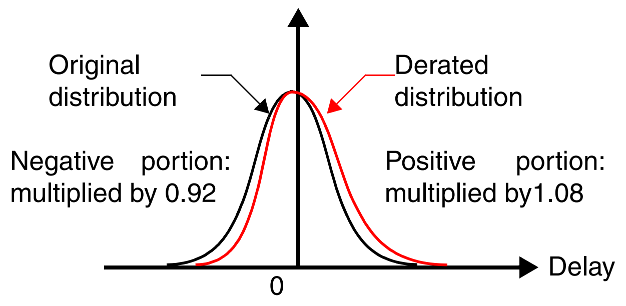


In the unusual case of negative delay values, a different derating formula is used to consistently give a more conservative analysis:

new_delay = *old_delay* * (2 – *derating_factor*)

When part or all of the delay distribution is negative, the negative part of the delay is adjusted as shown in Figure 3-8.

Figure 3-8 Derated Maximum-Delay Distribution With Negative Portion


Original distribution
Derated distribution
Negative portion: multiplied by 0.92
Positive portion: multiplied by1.08
Delay
0

To report derating factors that have been set, use the `report_timing_derate` command. Use the `-variation` option to report statistical derating settings, or omit the option to report scalar settings.

Reporting Variations

You can report variations used in the design by using the `report_variation` command. In the command, you specify a collection of objects to be reported. The collection can contain paths collected with the `get_recalculated_timing_paths` command; cells collected with the `get_cells` command; nets collected with the `get_nets` command; or library cells collected with the `get_lib_cells -of_objects` command. If no collection is specified, the command applies to the whole current design.

You can sort and filter collections based on variation-aware slack, arrival times, and so on by setting a variable that controls the variation-derived attribute mode:

```
variation_derived_scalar_attribute_mode
```

Then you can use the `sort_collection` and `filter_collection` commands.

This is the syntax of the `report_variation` command:

```
report_variation
  [-significant_digits digits]
  [-verbose]
  [-nosplit]
  [-delay_type delay_type]
  [-slack_lesser_than slack_limit]
  [-nworst num_worst]
  [-clock_network]
  [-all_cells]
  [-all_nets]
  [-transition]
  [-parametric_sensitivity parameter_list]
  [-input_pins]
  [collection]
```

Variation Sensitivity

The statistical sensitivity of a timing distribution attribute (such as arrival time or slack) is a measure of how much that attribute changes in response to changes in related variations. The sensitivity values in a design and the sensitivity values of timing arcs in a library can be reported with the `report_variation` command.

For any timing distribution “A,” the sensitivity of that distribution is defined to be the normalized variance of its distribution, calculated as follows:

sensitivity(A) = std_dev(A) / | mean(A) |

The sensitivity of distribution “A” is the standard deviation of “A” divided by the absolute value of the mean of “A.” If the mean of “A” is zero or very close to zero, a small threshold value is used instead of the zero or near-zero value.

The sensitivity of a constrained timing path is defined as the sensitivity of the path’s slack, whereas the sensitivity of an unconstrained path (such as a clock path) is defined as the path’s arrival sensitivity:

sensitivity(constrained_path) = sensitivity(variation_slack)

sensitivity(unconstrained_path) = sensitivity(variation_arrival)

Each timing point within a path has an arrival time sensitivity and a transition time sensitivity:

delay_sensitivity(point) = sensitivity(variation_arrival)

slew_sensitivity(point) = sensitivity(variation_transition)

Library arcs have sensitivity values that indicate the sensitivity of delay and slew with changes in a random variable *rv*:

delay_sensitivity(*lib_arc*) = D(*delay*) / D(*rv*)

slew_sensitivity(*lib_arc*) = D(*slew*) / D(*rv*)

The random variable *rv* is defined by the `set_variation_library` command. Multiple libraries are usually involved with each random variable, and the delay and slew functions depend on input slew and load capacitance, so multiple lookup tables represent the delay and slew sensitivity functions.

Note that the two different types of sensitivities described here are fundamentally different in what they represent. The path and timing point sensitivity values contained within path objects are indicators of the design sensitivity to variations in the context of the design. These values represent the amount of variation relative to the mean value of the measured path or point attribute. This type of normalized sensitivity is known as “design sensitivity.”

The sensitivity values obtained from library arcs are called “library sensitivity” values. Rather than representing the amount of normalized variation relative to the arc itself, library sensitivity represents the amount of variation relative to a specific variation parameter of interest. Library sensitivity is not normalized; its magnitude depends on how the library arc timing data changes with a given change in parameter value.

Variation-Derived Scalar Attributes

When variation-aware analysis is enabled, getting a `timing_path` attribute called `slack` returns the low quantile slack value for the path. For example,

```
pt_shell> set path1 [get_timing_paths -to reg[31]/D]
...
pt_shell> set myrecalc \
          [get_recalculated_timing_paths $path1]
...
pt_shell> get_attribute $myrecalc slack
-0.455260
```

In this example, the quantile slack obtained by variation-aware analysis is –0.455 time units.

PrimeTime VX uses separate attributes to hold the fixed slack and variation-aware slack. The name of the variation-aware attribute is `variation_slack`, made by adding the string `variation_` as a prefix. There are also variables named `variation_arrival`, `variation_transition`, and so on.

The `variation_derived_scalar_attribute_mode` variable controls how the distributions in these variation attributes are converted to the corresponding numerical attributes. By default, this variable is set to `quantile` during variation-aware analysis, which causes conservative statistical values from the edge of the distribution to be returned. If you set it to `mean`, the mean values are reported, or if you set it to `none`, the corner-case values are reported. For example,

```
pt_shell> set variation_derived_scalar_attribute_mode mean
mean
pt_shell> get_attribute $myrecalc slack
-0.092147
pt_shell> set variation_derived_scalar_attribute_mode none
none
pt_shell> get_attribute $myrecalc slack
-1.042500
```

Depending on the variable setting, PrimeTime reports the less conservative mean value of the slack distribution or the more conservative corner-case slack value.

The quantile values returned depend on the `-quantile_low` and `-quantile_high` option settings of the `set_variation_quantile` command. The default quantile low and high settings are 0.00135 and 0.99865, respectively, corresponding to the –3*sigma and +3*sigma points of a perfectly normal distribution. For quantities like delay and slew, the use of the low or high quantile value depends on whether the most conservative value is on the low or high side of the distribution. Slack values always use `-quantile_low` because lower slack values are more conservative.

The `variation_derived_scalar_attribute_mode` setting only affects the reporting of attributes in the database, not the actual values stored. It affects the behavior of `get_attribute`, `sort_collection`, and `filter_collection`. For example, with the variation-derived scalar attribute mode set to `mean`, the `sort_collection` command sorts a collection of recalculated timing paths in order of mean rather than quantile slack.

Variations in Timing Paths

To get a report on the variations used in timing paths that have been recalculated with variation-aware analysis enabled, create a collection of those paths and apply the `report_variation` command. For example, the following script creates a collection of the two worst timing paths, recalculates the timing of those paths with variations, sorts the paths by quantile slack, and generates a summary report on the variations used in the paths.

```
set paths [get_timing_paths -nworst 2]
set variation_derived_scalar_attribute_mode quantile
report_variation $paths
set recalc_paths [get_recalculated_timing_paths $paths]
set var_paths [sort_collection $recalc_paths "slack"]
report_variation $var_paths
```

A standard report on path variations looks like this:

```
  Path  startpoint  endpoint   quantile      mean     stddev   sensitiv
  ---------------------------------------------------------------------
     0          in       ff1   3.051712  3.054828   0.001437   0.047031
     1         ff1       ff2   9.868665  9.886533   0.007124   0.072059
```

The report shows the path startpoint, path endpoint, slack at the low quantile, mean slack, standard deviation of slack, and the slack sensitivity. Sensitivity values are reported in percent:

[*std_dev(A)* / | *mean(A)* |] * 100

A verbose report on path variations, using the `-verbose` option, looks like this:

```
 Path      startpoint      endpoint   quantile   sensitiv  mean stdd
-------------------------------------------------------------------
    1            ff1           ff2   9.868665   0.072059  ...  ...

         Path attribute   quantile   sensitiv       mean  stdd incr
-------------------------------------------------------------------
                arrival   0.219267   3.440833   0.202071   ... ...
                  slack   9.868665   0.072059   9.886533   ... ...
               required  10.084497   0.018024  10.088605   ... ...
startpoint_clock_latency   0.065878   2.346163   0.062349   ... ...
  endpoint_clock_latency   0.089497   1.942685   0.093604   ... ...

          Point arrival    quantile   sensitiv       mean    std inc
--------------------------------------------------------------------
               clk (in)    0.000000   0.000000   0.000000    ... ...
         b01/ZN (INVD1)    0.009146   2.494643   0.008588    ... ...
         b02/ZN (INVD1) *  0.033515   3.722505   0.030891    ... ...
         b11/ZN (INVD1)    0.048736   2.773270   0.046092    ... ...
         b12/ZN (INVD1)    0.065878   2.346163   0.062349    ... ...
          ff1/CP (DFD1)    0.065878   2.346163   0.062349    ... ...
           ff1/Q (DFD1) *  0.200045   3.751771   0.183375    ... ...
         b21/ZN (INVD1)    0.219267   3.440833   0.202071    ... ...
        ...
```

The verbose report includes the quantile and sensitivity values, mean, standard deviation, and incremental standard deviation for the arrival time of each timing point along the path; and the quantile, sensitivity, mean, and standard deviation values for the whole-path arrival time, slack, required time, startpoint clock latency, and endpoint clock latency, where applicable. The asterisk characters (*) in the list indicate the stages having the highest incremental standard deviation of the arrival time in the launch path, data path, and capture path. Use the `-input_pins` option if you want the variation report to include the arrival times at cell inputs as well as at the cell outputs along the path.

Variations in Cells

To get a report on the sensitivity of one or more cells in the design, create a collection of those cells and apply the `report_variation` command. For example, the following commands report the variations in three specified cells in the design:

```
pt_shell> set mycells [get_cells {u13 u24 u02}]
 ...
pt_shell> report_variation $mycells -significant_digits 6
 ...
cell       lib_cell       slack         delay          mean        stddev
 -------------------------------------------------------------------------
 u13       AN2D1       0.542629      0.114985      0.096514      0.007269
 u24       OR2D1       0.493780      0.090585      0.075847      0.005648
 u02       BUFFD1      0.493780      0.086466      0.075197      0.004733


pt_shell> report_variation $mycells -verbose -significant_digits 6
 ...
Cell delay variation: u13 (AN2D1)
 from      to         sense                slack     ...    ...
stddev
 -------------------------------------------------------------------------
----
 u13/A1    u13/Z    r positive_unate    0.553502     ...    ...
0.006643
 u13/A1    u13/Z    f positive_unate    0.556909     ...    ...
0.005430
 u13/A2    u13/Z    r positive_unate    0.542629     ...    ...
0.007269 *
 u13/A2    u13/Z    f positive_unate    0.546749     ...    ...
0.005876

Cell delay variation: u24 (OR2D1)
 from      to          sense               slack     ...    ...
stddev
 -------------------------------------------------------------------------
----
 u24/A1    u24/Z    r positive_unate    0.542162     ...    ...
0.005648 *
 u24/A1    u24/Z    f positive_unate    0.571965     ...    ...
0.004758
 u24/A2    u24/Z    r positive_unate    0.533186     ...    ...
0.004827
 u24/A2    u24/Z    f positive_unate    0.493780     ...    ...
0.005454

Cell delay variation: u02 (BUFFD1)
 from      to          sense               slack     ...    ...
stddev
 -------------------------------------------------------------------------
----
 u02/I     u02/Z    r positive_unate    0.533186     ...    ...
0.003757
 u02/I     u02/Z    f positive_unate    0.493780     ...    ...

0.004733 *
```

The default report lists the slack, the scalar arc delay, and the mean and standard deviation of the statistical arc delay for the worst timing arc through the cell. The worst arc is the arc with the highest standard deviation of arc delay among all timing arcs of the cell. The slack of an arc is the slack of the worst-case timing path through the arc. In order to report slack values, the variable `timing_save_pin_arrival_and_slack` must be set to true prior to `update_timing`.

The verbose report shows the same information for all the timing arcs through the cell, with the worst arc marked with an asterisk. (In the foregoing example, some columns of numbers were omitted so that the report would fit on the page.)

Variations in Nets

To get a report on the sensitivity of one or more nets in the design, create a collection of those nets and apply the `report_variation` command. For example, the following commands report the variations in nets whose worst slack is negative:

```
pt_shell> report_variation [get_nets] -slack_lesser_than 0
 ...

 net            slack     delay      mean    stddev
 --------------------------------------------------
 udat/in3     -4.0884    0.0100    0.0071    0.0012
 udat/w5      -3.7312    0.0088    0.0071    0.0007
 udat/w8      -3.6901    0.0067    0.0053    0.0006
 udat/in1     -3.5159    0.0064    0.0052    0.0005

pt_shell> report_variation [get_nets] -slack_lesser_than 0 -verbose

 Net delay variation: udat/in3
 from         to                slack     delay      mean    stddev
 ------------------------------------------------------------------
 ffi3/Q       udat/u4/A1  r   -2.9094    0.0100    0.0071    0.0012 *
 ffi3/Q       udat/u4/A1  f   -2.1000    0.0092    0.0069    0.0009
 ffi3/Q       udat/u2/A1  r   -3.9189    0.0094    0.0071    0.0010
 ffi3/Q       udat/u2/A1  f   -4.0884    0.0088    0.0070    0.0008

 Net delay variation: udat/w5
 from         to                 slack     delay      mean    stddev
 -------------------------------------------------------------------
 udat/u1/ZN   udat/u6/A2  r   -2.7033    0.0083    0.0071    0.0005
 udat/u1/ZN   udat/u6/A2  f   -3.7312    0.0082    0.0071    0.0005
 udat/u1/ZN   udat/u4/A2  r   -3.6901    0.0088    0.0071    0.0007 *
 udat/u1/ZN   udat/u4/A2  f   -3.5759    0.0086    0.0071    0.0007

 Net delay variation: udat/w8
 from         to                 slack     delay      mean    stddev
 -------------------------------------------------------------------
 udat/u4/Z    udat/u5/A2  r   -3.6901    0.0067    0.0053    0.0006 *
 udat/u4/Z    udat/u5/A2  f   -3.5759    0.0065    0.0052    0.0006

 Net delay variation: udat/in1
 from         to                 slack     delay      mean    stddev
 -------------------------------------------------------------------
 ffi1/Q       udat/u1/A1  r   -3.5142    0.0063    0.0051    0.0005
 ffi1/Q       udat/u1/A1  f   -3.5159    0.0064    0.0052    0.0005 *
```

The default report lists the slack, the scalar net arc delay, and the mean and standard deviation of the statistical net arc delay for the worst timing arc of the net. The worst arc is the arc with the highest standard deviation of net arc delay among all timing arcs of the net. The slack of an arc is the slack of the worst-case timing path through the net. In order to report slack values, the variable `timing_save_pin_arrival_and_slack` must be set to true prior to `update_timing`.

The verbose report shows the same information for all the timing arcs through the net, with the worst arc marked with an asterisk.

Parameter-Specific Net Sensitivity

The `report_variation` command can report the sensitivity of net delay to specified variation parameters. This type of reporting helps you find the nets that have the most delay sensitivity to given parameters, which can provide guidance on where to fix or reroute nets to reduce the delay sensitivity and produce a more reliable design.

For example, the following report shows the delay sensitivity of net n1 to all variation parameters:

```
pt_shell> report_variation -parametric_sensitivity *\
          [get_nets n1]
...
net          parameter              sensitivity
-----------------------------------------------
n1           metal2_T                  0.035652
```

```
            metal2_W                   0.026661
            metal3_T                   0.025091
            metal3_W                   0.017360
            metal1_T                   0.015018
            metal1_W                   0.011877
```

The reported sensitivity values show the standard deviation of the net delay contributed by each variation parameter, based on a calculation using changes to the net resistance and capacitance with changes in the specified parameters.

Variations in Library Cells

To get a report on the sensitivity of library cells, create a collection of those cells and apply the `report_variation` command. For example, the following script creates a collection of all library cells and generates a summary report on the variation sensitivities of the cells.

```
pt_shell> set my_lib_cells \
          [get_lib_cells -of_objects [get_cells]]

pt_shell> report_variation $my_lib_cells
```

A report on the library cell sensitivities looks like this:

```
       Library sensitivity for cell: INVD1

       delay sensitivity for cell INVD1 arc I->ZN
       timing sense                  r negative_unate      f
negative_unate
       -----------------------------------------------------------------
----
         len: min/avg/max   0.000127/0.001236/0.003896
0.000066/0.000678...
         vth: min/avg/max   0.040482/0.409654/1.425395
0.021974/0.286770...

       slew sensitivity for cell INVD1 arc I->ZN
       timing sense                  r negative_unate      f
negative_unate
       -----------------------------------------------------------------
----
         len: min/avg/max  -0.000214/0.000842/0.003220  -
0.000149/0.000278...
         vth: min/avg/max   0.012500/0.268289/0.972281  -

0.062368/0.102758...
```

In this example, there are two independent random variables, len and vth, specified by the `set_variation_library` command. The report shows the minimum, average, and maximum sensitivity of each arc's rise/fall delay and slew. Note that the library sensitivity report collects only the static library information, which does not directly reflect the actual behavior of the cells in the design.

In order to have slack values appear in the report, the `timing_save_pin_arrival_and_slack` variable must be set to `true` before `update_timing`.

Variations in Design Objects

The `report_variation` command can be used to report information about the variation sources used in the design. For example, to report on the whole current design,

```
pt_shell> report_variation [current_design]

Variation libraries in design: TEST

       library     par1       par2       par3       par4       par5
------------------------------------------------------------------
 ncx_mc_nom.db   0.0000     0.0000     0.0000     0.0000     0.0000
 ncx_mc_-P1.db  -1.0000     0.0000     0.0000     0.0000     0.0000
 ncx_mc_+P1.db   1.0000     0.0000     0.0000     0.0000     0.0000
 ncx_mc_-P2.db   0.0000    -0.3333     0.0000     0.0000     0.0000
 ncx_mc_+P2.db   0.0000     0.3333     0.0000     0.0000     0.0000
 ncx_mc_-P3.db   0.0000     0.0000    -0.3333     0.0000     0.0000
 ncx_mc_+P3.db   0.0000     0.0000     0.3333     0.0000     0.0000
 ncx_mc_-P4.db   0.0000     0.0000     0.0000    -0.3333     0.0000
 ncx_mc_+P4.db   0.0000     0.0000     0.0000     0.3333     0.0000
 ncx_mc_-P5.db   0.0000     0.0000     0.0000     0.0000    -0.3333
 ncx_mc_+P5.db   0.0000     0.0000     0.0000     0.0000     0.3333

Cell variation for design: TEST

      parameter         name        type          mean        stddev
  -----------------------------------------------------------------
           par1         par1      normal        0.0000        0.3330
           par2         par2      normal        0.0000        0.3330
           par3         par3      normal        0.0000        0.3330
           par4         par4      normal        0.0000        0.3330
           par5         par5      normal        0.0000        0.3330

Parasitic variation for design: TEST

    parameter        name            type          mean
stddev
  -----------------------------------------------------------------
-
```

```
            FOX_T       FOX_T           normal        0.0000
0.0583
           poly_T      poly_T           normal        0.0000
0.0333
           poly_W      poly_W           normal        0.0000
0.0283
            ILD_T       ILD_T           normal        0.0000
0.0510
         metal1_T    metal1_T           normal        0.0000
0.0333
            ...
```

The report is divided into sections. The first section reports all variation libraries with their corresponding parameters for each variation source. These libraries and parameters are defined by the `set_variation_library` command. The second section summarizes the global device variation that is present in all cell arcs in the entire design. For each distribution variable, the column labeled `component` corresponds to the `-name` specified in the `set_variation` command; and the columns labeled `type`, `mean`, and `stddev` correspond to the `-type` and `-values` settings in the `create_variation` command. The succeeding sections report the variations present in each individual cell or arc, in addition to the global variation settings. The "parasitic variation" section shows random variables that represent variation sources in parasitic networks.

Variations in Parasitic Data

The `-variation` option of `report_annotated_parasitics` generates a report of the variation sensitivity of a net. For example,

```
pt_shell> report_annotated_parasitics -variation \
          [get_nets n1]

  ParamID      Name                 Stddev
-------------------------------------------------------
      0       FOX_T                0.05882
      1       poly_T               0.03333
     ...
```

The `-variation` option of `report_annotated_parasitics` generates a report of the variation sensitivity of a net. For example,

```
      pt_shell> report_annotated_parasitics -variation \
                 -list_annotated [get_net n1]
      ...
        C in pF     Node    Pin/Port         Sensitivity
        -------------------------------------------------------------------
----
         0.0003       1     ffc/QN (driver)  FOX_T(0.006) poly_T(0.000)
...
         0.0002       2      i1/A  (load)    FOX_T(0.001)
metal1_T(0.046)...
        ...
        -------------------------------------------------------------------
----
         0.0064  Total

        R in Kohm   Left Node        Right Node        Sensitivity
        -------------------------------------------------------------------
----
          0.0091     1 ffc/QN (driver)  2 i1/A  (load)  metal2_T(0.320)
...
        -------------------------------------------------------------------
----
          0.0092  Total

                          |       |        |         |    RC   |   Not   |
      Net Type            | Total | Lumped |  RC pi  | network |Annotated|
      ------------------- +-------+--------+---------+---------+---------+
      Internal nets       |       |        |         |         |         |
        - Pin to pin nets |     1 |      0 |       0 |       1 |       0 |
        - Driverless nets |     0 |      0 |       0 |       0 |       0 |
        - Loadless nets   |     0 |      0 |       0 |       0 |       0 |

 ...
```

The report lists the variations in the SBPF file, not the variation values set with the `set_variation` command.

The `report_net`, `report_timing -capacitance`, and `report_delay_calculation` commands show only the fixed values resulting from nominal variation settings.

4

Variation-Aware Timing Reports

PrimeTime VX increases the accuracy of timing analysis and offers detailed reports on distributions of timing results, such as delay and slack, for a given set of parameter variations. Several reporting methods are available to get information about the distributions of the timing results.

This chapter contains the following sections:

Path Timing Report Increments

In a text-format timing report produced by the `report_timing` command, the arrival times shown in the tables are fixed values based on the quantiles of the arrival time distributions. In the default report format, there are columns labeled Incr (Incremental delay) and Path (arrival time in the path), as shown in the following example.

```
pt_shell> report_timing $path2
   ...
  Startpoint: wishbone/rx_fifo/cnt_reg[0]
               (rising edge-triggered flip-flop clocked by wb_clk_i)
  Endpoint: wishbone/IncrTxPointer_reg
               (rising edge-triggered flip-flop clocked by wb_clk_i)
  Path Group: wb_clk_i
  Path Type: max

  Point                                              Incr       Path
  ------------------------------------------------------------------
  clock wb_clk_i (rise edge)                         0.00       0.00
  clock source latency                               0.00       0.00
  wb_clk_i (in)                                      0.00 &     0.00 r
  wishbone/BUFFD16_G1B1I4_2/Z (CKBD16)               0.19 &     0.19 r
  wishbone/BUFFD16_G1B2I135/Z (BUFFD16)              0.17 &     0.36 r
  wishbone/rx_fifo/cnt_reg[0]/CP (SEDFCNQD1)         0.00 &     0.36 r
  wishbone/rx_fifo/cnt_reg[0]/Q (SEDFCNQD1)          0.42 &     0.78 r
  wishbone/rx_fifo/U71/ZN (NR4D2)                    0.12 &     0.90 f
  wishbone/rx_fifo/U21/Z (AN2D2)                     0.12 &     1.02 f
   ...
  wishbone/U1059/Z (OR2D4)                           0.13 &     2.48 f
  wishbone/IncrTxPointer_reg/E (SEDFCND0)            0.00 &     2.48 f
  data arrival time                                             2.48

  clock wb_clk_i (rise edge)                         2.50       2.50
  clock source latency                               0.00       2.50
  wb_clk_i (in)                                      0.00 &     2.50 r
  BUFFD16_G1B1I2/Z (BUFFD16)                         0.16 &     2.66 r
  wishbone/BUFFD16_G1B2I56/Z (BUFFD16)               0.09 &     2.75 r
  wishbone/IncrTxPointer_reg/CP (SEDFCND0)           0.00 &     2.75 r
  clock reconvergence pessimism                      0.00       2.75
  clock uncertainty                                 -0.10       2.65
  library setup time                                -0.28       2.37
  data required time                                            2.37
  -------------------------------------------------------------------
  data required time                                            2.37
  data arrival time                                            -2.48
  -------------------------------------------------------------------
  slack                                                        -0.11
  statistical adjustment (slack)                     0.10      -0.01
  slack (VIOLATED) (with probability 0.012)                    -0.01
```

The `variation_report_timing_increment_format` variable affects the display of paths which have been recalculated within the context of a variation-aware timing analysis. The transition times, delays and arrival times associated with these paths are statistical in nature. This variable lets you control the display of the delays appearing in the Incr column.

The variable can be set to `effective_delay` (the default) or `delay_variation`. When it is set to `effective_delay`, each increment displayed is the scalar difference between the arrival values appearing in the Path column. When it is set to `delay_variation`, the quantile or mean value of the statistical increment is displayed instead, according to the current setting of the `variation_derived_scalar_attribute_mode` variable.

The arrival times and transition times shown in the report are also scalar representations based on the underlying distributions, converted into scalar values based on the current setting of the `variation_derived_scalar_attribute_mode` variable.

Clock Timing Reports

The `report_clock_timing` command generates a report on clock timing in the design. You can report the same-clock skew, interclock skew, latency, or transition time attributes of a given clock network or sub-network. You can choose a summary report showing the worst instances of skew, latency, or transition time; or a verbose report showing a trace through a source-to-sink path through a clock network, including the accumulation of the reported attribute (skew, latency, or transition time) along the path.

If you use the `-variation` option of the command, the report includes variation information, including the quantile, mean, and standard deviation (for skew) or variation sensitivity (for latency or transition time). To get a clock timing report with variation information, the `variation_analysis_mode` variable must be set to the string `detailed_clock_timing`.

For example, an ordinary summary clock timing report starts with the worst latency values, like this:

```
pt_shell> report_clock_timing -type summary
   ...
  Clock: wb_clk_i
------------------------------------------------------------------------
  Maximum setup launch latency:
      wishbone/bd_ram/mem1_reg[252][12]/CP                0.43    rp-+

  Minimum setup capture latency:
      wishbone/bd_ram/mem0_reg[198][7]/CP                 0.23    rp-+

  Minimum hold launch latency:
      wishbone/bd_ram/mem0_reg[198][7]/CP                 0.23    rp-+

  Maximum hold capture latency:
      wishbone/bd_ram/mem1_reg[252][12]/CP                0.43    rp-+
 ...
```

By using the `-variation` option, the standard deviation, mean, and quantile values for the worst latency values are shown in the report:

```
pt_shell> report_clock_timing -type summary -variation
   ...
  Clock: wb_clk_i                          Stddev  Mean    Quant
-----------------------------------------------------------------------
Maximum setup launch latency:
   wishbone/bd_ram/mem1_reg[252][12]/CP     0.03    0.36    0.43    rp-+

Minimum setup capture latency:
   wishbone/bd_ram/mem0_reg[198][7]/CP      0.02    0.29    0.23    rp-+

Minimum hold launch latency:
   wishbone/bd_ram/mem0_reg[198][7]/CP      0.02    0.29    0.23    rp-+

Maximum hold capture latency:
   wishbone/bd_ram/mem1_reg[252][12]/CP     0.03    0.36    0.43    rp-+
 ...
```

The quantile values are the same as the plain worst-case latency values shown in the conventional report: the 99% quantiles for launch latency values and the 1% quantiles for capture latency values. The quantiles used for reporting are affected by the setting of the `variation_derived_scalar_attribute_mode` variable.

In a verbose report showing the point-by-point timing along a clock path, a variation-aware report shows the mean, quantile, and sensitivity of the delay values along the path. For example,

```
pt_shell> report_clock_timing -type latency -verbose -variation
   ...
  Clock: wb_clk_i

  Endpoint: wishbone/bd_ram/mem1_reg[248][14]
              (rising edge-triggered flip-flop clocked by wb_clk_i)

                                                --- Delay ---
  Point                          Trans     Sensit  Mean    Quant     Path
  -----------------------------------------------------------------------
---
  clock wb_clk_i (rise edge)
  clock source latency                                     0.00       0.00
  wb_clk_i (in)                   0.00                     0.00 &     0.00
r
  CKBD12_G1B1I5/I (BUFFD12)       0.03     0.03     0.00    0.00 &     0.00
r
  CKBD12_G1B1I5/Z (BUFFD12)       0.51     0.07     0.16    0.19 &     0.19
r
```

```
    BUFFD12_G1B2I266/I (BUFFD12)
                                    0.51     0.03     0.00     0.00 &      0.19
  r
    BUFFD12_G1B2I266/Z (BUFFD12)
                                    0.08     0.13     0.10     0.13 &      0.32
  r
    wishbone/bd_ram/CKBD6_Id_024/I (CKBD6)
                                    0.08     0.00     0.00     0.00 &      0.32
  r
    wishbone/bd_ram/CKBD6_Id_024/Z (CKBD6)
                                    0.18     0.07     0.10     0.12 &      0.43
  r
    wishbone/bd_ram/mem1_reg[248][14]/CP (SEDFQD1)
                                    0.18     0.00     0.00     0.00 &      0.43
  r
    VA latency adjustment                                     -0.00        0.43
    total clock latency                                                    0.43
    ----------------------------------------------------------------------------
  ---
    latency (quantile)                                                     0.43
    latency (mean)                                                         0.36
    latency (sensitivity)                                                  0.08
```

For information about variation sensitivity, see “Variation Sensitivity” on page 3-19.

Custom Statistical Reporting

PrimeTime VX can provide detailed statistical reports on delays, arrival times, and slacks. These types of information are available as timing point, timing path, variation, and distribution attributes. They can also be reported by the PrimeTime graphical user interface in the form of tables and graphs.

The statistical variation attributes of the `timing_point`, `timing_path`, and `variation` object types are available for paths with statistical timing data. These attributes are listed in Table 4-1 through Table 4-3. You can extract the information stored in these attributes by using the `get_attribute` and `get_variation_attribute` commands.

Table 4-1 Timing Point Object Attributes

Name and type	Description
variation_arrival, collection	Arrival time variation of the timing point
variation_slack, collection	Slack time variation of the timing point
variation_transition, collection	Transition time variation of the timing point

Table 4-2 Timing Path Object Attributes

Name and type	Description
sensitivity, float	Variation sensitivity of the path
variation_arrival, collection	Arrival time variation of the path
variation_common_path_pessimism, collection	The variation of the clock reconvergence common path pessimism.
variation_endpoint_clock_latency, collection	Capture-clock arrival time variation
variation_endpoint_hold_time_value, collection	The variation of the register hold time at the timing endpoint.
variation_endpoint_recovery_time_value, collection	The variation of the recovery time at the timing endpoint.
variation_endpoint_removal_time_value, collection	The variation of the removal time at the timing endpoint.
variation_endpoint_setup_time_value, collection	The variation of the register setup time at the timing endpoint.
variation_required, collection	Required-time variation of the path
variation_slack, collection	Slack variation of the path
variation_startpoint_clock_latency, collection	Launch-clock arrival time variation

Table 4-3 Variation Object Attributes

Name and type	Description
full_name, string	Name of the variation
parameter_name, string	Name of the associated parameter defined in the library or parasitic data file
mean, float	Mean of the distribution of the variation
std_dev, float	Standard deviation of the distribution of the variation
variance, float	Variance of the distribution of the variation
skewness, float	Skewness of the distribution of the variation
summary, string	Text information about the distribution of the variation, including mean, standard deviation, skewness, and five quantile points
cdf	A single Cumulative Distribution Function value obtained with `get_variation_attribute`.
cdf_values	A sequence of successive Cumulative Distribution Function value obtained with `get_variation_attribute`.
pdf	A single Probability Density Function value obtained with `get_variation_attribute`.
pdf_values	A sequence of successive Probability Density Function values obtained with `get_variation_attribute`.
quantile	A single quantile value obtained with `get_variation_attribute`.

Variation Distribution Attributes

You can use the `get_attribute` command to get information about a statistical distribution in a variation object. For example, to get the standard deviation of variation p1:

```
pt_shell> get_attribute [get_variations p1] std_dev
0.310000
```

These are the distribution attributes you can get with the `get_attribute` command from a variation:

- `mean` – The mean value.
- `std_dev` – The standard deviation.
- `variance` – The variance.
- `skewness` – The skewness.
- `summary` – A string containing a statistical summary of the distribution, consisting of the mean, standard deviation, skewness, and quantiles q01, q05, q50 (median), q95, and q99.

You can query this same type of information from a path. For example, for the path stored in the variable “a_path” with the `get_timing_paths` command:

```
pt_shell> set path_slack \
          [get_attribute $a_path variation_slack]
```

you can get its mean value:

```
pt_shell> get_attribute $path_slack mean
-0.092147
```

and its statistical summary:

```
pt_shell> get_attribute $path_slack summary
mean=   -0.092147
sdev=   0.134774
skew=   -0.125206
q01=    -0.455260
q05=    -0.312938
...
```

Some variation attributes require a value to be specified before you can retrieve them, such as quantiles. You can access these attributes by using the `get_variation_attribute` command. For example, to get the q93 quantile value:

```
pt_shell> get_variation_attribute \
          [get_variations p1] quantile .93
0.350722
```

These are the distribution attributes you can get with the `get_variation_attribute` command:

- `quantile` *`q_value`* – The value of the distribution at the q-value quantile point, where *`q_value`* is a floating-point number greater than 0.0 and less than 1.0.
- `cdf` *`cdf_value`* – The value of the cumulative distribution function at a CDF-value point, where *`cdf_value`* is a floating-point number.
- `cdf_values` *`number_of_values`* – A list of distribution function values at the number of data points specified by *`number_of_values`* (an integer), centered around the mean.
- `pdf` *`pdf_value`* – The value of the probability function at a PDF-value point, where *`pdf_value`* a floating-point number between 0.0 and 1.0.
- `pdf_values` *`number_of_values`* – A list of probability density function values at the number of data points specified by *`number_of_values`*, centered around the mean.

Attribute Access Examples

The following script creates two procedures, `report_my_attr` and `report_point_attr`, which demonstrate some examples of accessing attributes in the design.

```
proc report_my_attr { path } {
    set points [get_attribute $path points]

    foreach_in_collection point $points {
        # report each timing point
        report_point_attr $point
    }
}

proc report_point_attr { point } {
    # get string of pin
    set pin [get_attribute [get_attribute $point object] full_name]
    puts "Pin: $pin  "

    # get rise or fall
    set rise_fall [get_attribute $point rise_fall]
    puts "type: $rise_fall"

    # for two variables arrival and transition
foreach v {arrival transition} {
  # get corresponding random variable
  set val [get_attribute $point $v]
  set var variation_$v
  set rv [get_attribute $point ${var}]
  set mean [get_attribute $rv mean]
  set sdev [get_attribute $rv std_dev]
  puts "$v mean: $mean sdev: $sdev"
```

```
        puts "99 percent quantile "
        set q [get_variation_attribute $rv quantile 0.99]
        puts -nonewline [format "q%02d = $q " $p]
        }
      }
```

Reporting Variation Correlations

The `variation_correlation` command provides the correlation between variation timing attributes, as well as the correlation of variation timing attributes to variation parameters. Note that when variation parameters are used, this capability is supported only for global variations, that is, for an autocorrelation of `1`.

User-defined reports can be created as desired. An example of a Tcl procedure to generate the correlation report of variation slack is as follows:

```
proc report_slack_variation_data { slack_variation } {
      echo "Average : " [get_attribute $slack_variation mean]
      echo "Variation : " [get_attribute $slack_variation std_dev] \
           " @ 1sigma"
      echo "Correlation with variation parameters :"

      foreach_in_collection variation [get_variations *] {
           set var_name [get_attribute $variation parameter_name]
           set var_corr [variation_correlation $slack_variation
            $variation]
           echo [format " %s\t : %s" $var_name $var_corr] }
}
```

Session Examples

The following script is a variation-aware analysis session using two variations, channel length and threshold voltage.

```
set variation_enable_analysis true

set_variation_library -parameter_names "len vth" \
 -values "90 0.22" -reference_value typical.db
set_variation_library -parameter_names "len vth" \
 -values "90 0.24" vthplus.db
set_variation_library -parameter_names "len vth" \
 -values "90 0.20" vthminus.db
set_variation_library - parameter_names "len vth" \
 -values "95 0.22" lenplus.db
set_variation_library - parameter_names "len vth" \
 -values "85 0.22" lenminus.db

create_variation -name len -parameter_name len \
 -type normal -values {0.0 10.0}]
create_variation -name vth -parameter_name vth \
 -type normal -values {0.0 0.05}]
set_variation [get_variations len]
set_variation [get_variations vth]

update_timing -full

report_timing -nworst 2

set p1 [get_timing_paths -nworst 2]
report_timing $p1

set p2 [get_recalculated_timing_paths $p2]
report_timing $p2
```

The following script performs a variation-aware analysis using four variations named A, B, C, and D.

```
# set variable defaults
set report_default_significant_digits 3
set variation_enable_analysis true

# define operating conditions
set_operating_conditions -analysis_type on_chip_variation

# set +/- sigma variation libraries
set_variation_library -parameter_names {A B C D} \
 -values {0 0 0 0} \
 lib_snom.db -reference_value
set_variation_library -parameter_names {A} \
 -values -1 libA_neg.db
set_variation_library -parameter_names {A} \
 -values +1 libA_pos.db
set_variation_library -parameter_names {B} \
 -values -1 libB_neg.db
set_variation_library -parameter_names {B} \
 -values +1 libB_pos.db
set_variation_library -parameter_names {C} \
 -values -1 libC_neg.db
set_variation_library -parameter_names {C} \
 -values +1 libC_pos.db
set_variation_library -parameter_names {D} \
 -values -1 libD_neg.db
set_variation_library -parameter_names {D} \
 -values +1 lib_sD_pos1s.db

# set parameter variations
create_variation -name A -parameter_name A \
 -type normal -values {0 1}
create_variation -name B -parameter_name B \
 -type normal -values {0 1}
```

```
create_variation -name C -parameter_name C \
 -type normal -values {0 1}
create_variation -name D -parameter_name D \
 -type normal -values {0 1}
set_variation [get_variations A]
set_variation [get_variations B]
set_variation [get_variations C]
set_variation [get_variations D]

create_correlation -name die_to_die -constant 1
set_variation_correlation -name all_die_to_die \
 -correlation die_to_die -all

# read parasitics, constraints, update timing
read_parasitics -keep_capacitive_coupling \
 test_astro.spef.gz
read_sdc test.sdc

update_timing -full
# perform statistical reporting
set path1 [get_timing_paths -fall_from wb_adr_i[9] \
 -to ethreg1/MODER_1/DataOut_reg[1]/D]
set path2 [get_timing_paths \
 -rise_through wishbone/ram_addr_reg[1]/Q \
 -rise_to wishbone/bd_ram/mem2_reg[211][23]/E]
set recalc1 [get_recalculated_timing_paths $path1]
set recalc2 [get_recalculated_timing_paths $path2]
report_timing -input_pins $path1
report_timing -input_pins $path2
report_timing -input_pins $recalc1
report_timing -input_pins $recalc2
```

Validation Flow

PrimeTime VX produces analysis results that show the distributions of delay and slew along selected paths. To validate the accuracy of the PrimeTime VX analysis, you can compare the reported results against a sequence of SPICE simulations of the same path, using a Monte Carlo statistical representation of the variation distributions. This is the general procedure for SPICE validation:

1. Perform the variation-aware analysis in PrimeTime VX.

   For each timing path analyzed by PrimeTime VX, you can get detailed statistical reports on the delay and slew attributes at each pin by using the `get_attribute` and `get_variation_attribute` commands. The statistical reports can include the mean, standard deviation, quantile values, and PDF or CDF delay and slew values.

2. Create the simulation data files for Monte Carlo SPICE simulation. The following command generates multiple SPICE decks from PrimeTime VX, using randomly distributed parameter and parasitic element values.

   ```
   pt_shell> write_spice_deck -output path.spo \
             -sample_size 24 \
             -sub_circuit_file my_sub.sp \
             [get_timing_paths -justify]
   ...
   ```

   This example generates 24 SPICE decks in the directory named path.spo, using the statistical distribution of parameter and parasitic element values defined in the variation-aware design. Cross-coupling capacitors, if any, are not sampled, but are assigned their corner values.

3. Start the SPICE simulation runs and collect the statistical results.

   The SPICE simulation should be performed *n* times, where *n* is large enough to capture the statistical properties of delay and slew on each pin according to the statistical properties of each variation component. Each SPICE run must use a different set of random numbers.

   After the Monte Carlo sequence of SPICE runs is finished, generate reports on the delay and slew values on each pin with *n* samples. You should create a report on the statistical distribution of delay and slew on each pin, including the mean, standard deviation, quantile values, and PDF or CDF plots.

4. Compare the PrimeTime VX results against the Monte Carlo SPICE results. The statistical distributions should be similar, thus validating the accuracy of the variation-aware analysis in PrimeTime VX.

For device variations, the `write_spice_deck` command writes out the sample values in the subcircuit portion as shown in the following example:

```
* .pin(sub_node) for inv1 (INVD1): .I(inv1/I) .ZN(inv1/ZN)
xinv1 inv1/I inv1/ZN INVD1 ssta_l=4.27218 ssta_delvto=-0.0114263
* .pin(sub_node) for inv2 (INVD1): .I(inv2/I) .ZN(inv2/ZN)
xinv2 inv2/I inv2/ZN INVD1 ssta_l=4.27218 ssta_delvto=-0.0114263
```

Each cell variation defined in PrimeTime VX is written in the form of *variation_name* = *sample_value*. This format works with HSPICE only if the variations are also defined in the cell subcircuit file, as shown in the following example:

```
.subckt AN2D0 A1 A2 Z ssta_l=1.0 ssta_delvto=0.0
MM3 M3:DRN M3:GATE M3:SRC M3:BULK nch ad=0.03p as=0.078p
l='0.1u*ssta_l' nrd=0.333 nrs=0.924 pd=0.5u ps=1.12u ps=1.12u
sa=2.6e-07 sb=9e-07 w=0.3u delvto="+ssta_delvto"
MM1 M1:DRN M1:GATE M1:SRC M1:BULK nch ad=0.072p as=0.039p
l='0.1u*ssta_l' nrd=0.863 nrs=0.715 pd=1.08u ps=0.56u sa=9.2e-07
sb=2.4e-07 w=0.3u delvto='+ssta_delvto'
```

You can use the generated SPICE decks directly to run HSPICE multiple times. Then you can compare the collected HSPICE simulation results against the statistical results reported by PrimeTime VX.

Glossary

CDF
See cumulative distribution function (CDF).

corner case
A set of two or more parameter values that is a combination of the most extreme values of those parameters, such as the combination of the highest threshold voltage and the longest channel length.

correlation
The strength of linear relationship or co-dependence between two random variables. For example, there is a very high correlation (nearly 1.0) between the channel lengths of two adjacent transistors on a die.

cumulative distribution function (CDF)
A mathematical function that represents the cumulative probability that a parameter will be less than or equal to a given value, across the full range of the parameter; the integral of the probability density function.

derating
The scaling of delay values accomplished by multiplying the calculated delay values by a fixed scaling factor, as specified by the `set_timing_derate` command.

distribution
A mathematical description of the range of values of a variation along with the likelihood of occurrence of those values. A distribution is typically expressed in terms of a few parameters, such as the mean and standard deviation of a normal (Gaussian) distribution.

Gaussian
The normal distribution having a bell-shaped curve.

min/max analysis
Static timing analysis that is based on fixed worst-case minimum and maximum values rather than a probability distribution.

normal (distribution)

The Gaussian distribution having a bell-shaped curve and described by two parameters, the mean and the standard deviation.

parameter

A characteristic or property that has a variable numeric value and has a effect on delay or slew that can be measured or simulated.

PDF

See probability density function (PDF).

piecewise-linear

A function that is made of linear segments connected end-to-end, which can be characterized by the segment endpoints.

percentile

For a random variable X with continuous CDF function *F*, the percentile *p* associated with value *x* is:

$$F(x) = P[X \leq x] = p$$

For example, if the slack at a pin follows a normal distribution with mean 0.0 and standard deviation 1.0, the probability that the slack is negative is 0.50, so 0.50 is the percentile associated with x-value 0.0.

principal component analysis

A mathematical procedure that simplifies a data set by transforming a number of correlated parameters into a smaller number of uncorrelated parameters, called the principal components. The first principal component contains as much of the data variability as possible, the second component contain as much of the remaining variability as possible, and so on.

probability density function (PDF)

A mathematical function that represents the likelihood that a parameter will have a specific value across the full range of values of that parameter; the derivative of the cumulative distribution function. It is similar to a histogram that has been smoothed to make a continuous function.

quantile

For a random variable X with continuous CDF function *F*, the quantile *q* associated with value *p* is given by:

$$q = F^{-1}(p)$$

For example, if a path arrival time *A* has a 99 percent quantile equal to 4.3, it means that the probability is 0.99 that an actual arrival (an outcome of *A*) will be less than or equal to 4.3.

For any timing distribution “A,” the sensitivity of that distribution is defined to be the normalized variance of its distribution, calculated as follows:

sensitivity(A) = std_dev(A) / | mean(A) |

sensitivity

For any timing distribution “A,” the sensitivity of that distribution is defined to be the normalized variance of its distribution, calculated as *sensitivity(A) = std_dev(A) /* |*mean(A)*| .

sigma (s)

The standard deviation, the most common measure of statistical dispersion; the square root of the variance. A large sigma means that the data samples are widely dispersed around the mean. A small sigma means that the data samples are tightly clustered around the mean.

skewness

A measure of the asymmetry of a random variable. The skewness *S* of a random variable *X* with mean m and standard deviation s is defined as

$$S = \frac{E[(X-\mu)^3]}{\sigma^3}$$

where *E* denotes the expected value with respect to the probability distribution of *X*. If the skewness is zero, the PDF plot is symmetric around its mean. If the skewness is positive or negative, the plot is not symmetric around its mean.

slew

The transition time of a signal; the amount of time a signal takes to change from high to low or from low to high.

standard deviation

The sample estimate of sigma (s).

uniform distribution

A distribution that has the same value throughout a given range.

variation

A random variable that represents a process parameter or device parameter that can be characterized, such as channel length or threshold voltage.

variation-aware static timing analysis

A technique to accurately analyze circuit performance in the presence of variation in process and environment parameters (such as channel length and threshold voltage). This approach significantly improves the accuracy of traditional static timing analysis by using the actual distributions of parameter variations instead of worst-case corner assumptions.

Index