Parametric On-Chip Variation (POCV) Application Note

Version 2.0, February 2017

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

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

Disclaimer

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

Trademarks

Synopsys company and certain product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

Introduction 5
POCV Analysis 6
POCV Input Data 8
  POCV Input from Side File (Single Coefficient) 8
  POCV Characterization 9
POCV Coefficient (Side File) Flow 10
  Set Up POCV Analysis 10
    Import SPEF/SBPF with Physical Locations (optional) 11
    Enabling POCV Analysis 11
    Loading POCV Information 11
    Guard-banding in POCV 12
    Scaling POCV Coefficient 12
    Running PBA Analysis with Derate-Only Mode 13
  POCV Reporting 13
    Reporting POCV Coefficient and Deratings 13
    Reporting POCV Analysis Results 14
POCV Slack and Arrival Attributes 20
POCV User Interface 21
How to Convert AOCV to POCV Side File 21
POCV Results 22
  POCV – AOCV Timing Comparison 22
  POCV – AOCV ECO Comparison 24
POCV Liberty Variation Format (LVF) Flow 25
  POCV Input in Liberty Variation Format (LVF) 25
  POCV LVF Characterization 26
  POCV LVF Analysis 27
    Enabling POCV LVF Analysis 27
    Loading POCV LVF Information 27
  POCV LVF Reporting 28

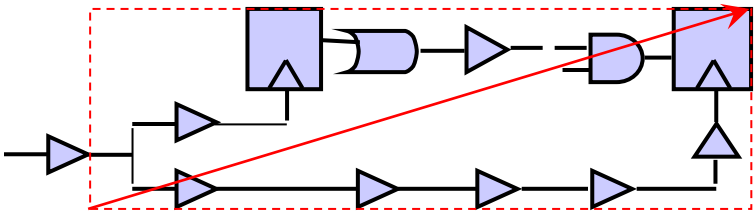
Reporting POCV LVF 28
Reporting POCV Analysis Results 29
How Slack Sensitivity is Calculated 33
POCV LVF Accuracy 34
POCV LVF – Monte Carlo Correlation 35
PrimeTime Without SI – HSPICE Correlation 35
PrimeTime POCV LVF – MC HSPICE Correlation 36
POCV Transition Variation 38
LVF Format for Transition Variation 38
Enabling Transition Variation 39
How Transition Variation Is Combined Into Cell Delay Variation and Output Transition 39
Reporting with Transition Variation 40
Accuracy with Transition Variation 41
POCV Constraint Variation 42
LVF Format for Constraint Variation 42
Enabling Constraint Variation 43
Reporting with Constraint Variation 43
POCV and Latches 46
Statistical Graph Pessimism 48
POCV and SI Analysis 51
POCV and Voltage Scaling 51
ETM with POCV Analysis 51
Running `check_library` from Library Compiler 52
Conclusion 53
Appendix 54
Example POCV Timing Analysis Script 54
POCV Side File Format for Single Coefficient 57
Syntax Definition 57
POCV Side File Example 59
POCV Library Variation Format (LVF) 60

Introduction

Process variation can be briefly categorized into chip-to-chip (die-to-die) variation and on-chip variation (OCV). In the traditional static timing analysis, the chip-to-chip variation effect is captured by analyzing the design in different timing corners. On-chip variation is captured by applying early and late timing derating factors to timing path elements.

As the feature size of the technology nodes keep decreasing, on-chip variation has become more complicated. Applying a single derating factor has become too simplistic and can either cause too much pessimism on setup paths or bring potential risk on hold paths. In order to overcome these problems, the Advanced OCV (AOCV) model was proposed to provide path depth and distance based derating factors to capture random and systematic on-die variation effects respectively, as shown in Figure 1.

Figure 1: AOCV Modeling



$$\Gamma = f(\text{distance}, \text{stage})$$

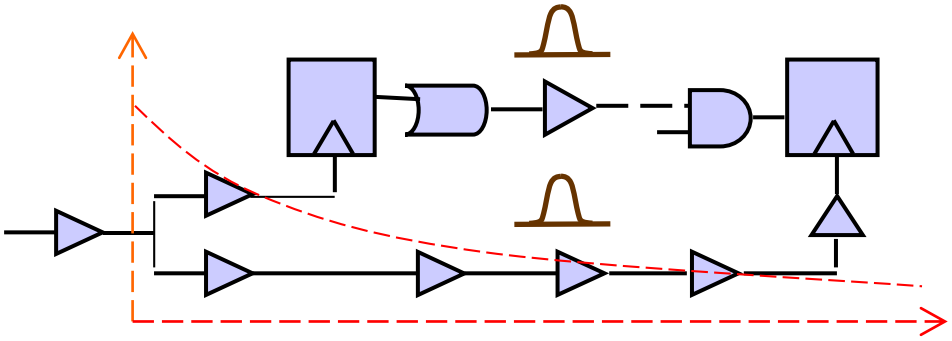
AOCV models derating factor as a combination of path depth and distance. PrimeTime models random on-die variation and uses derating factor as a function of path depth (stage count). It models systematic on-die variation and uses derating factor as a function of distance (path bounding box). Typically, random on-die variation dominates and therefore the path depth related OCV derate component tends to be the dominant factor between the two in an AOCV model.

By modeling derating factor as a function of path depth, depth-based AOCV captures the statistical cancellation effect of random variations. However, using path depth to determine derates for path elements is still a simplistic approach which can add pessimism in graph-based analysis. It also makes incremental timing very challenging because of the need to update derated delays due to stage count (path depth) dependency across the fan-in and fan-out cone from the location where the cell is inserted or removed.

An advantage of parametric on-chip variation (POCV) over AOCV is that it reduces the slack pessimism between graph-based and path-based analysis. This results in less pessimistic graph-based timing results, which can reduce the amount of effort needed in ECO fixing. Furthermore, it also makes incremental timing more efficient compared with AOCV. In addition, POCV handles the impact of spatial variation by supporting distance-based AOCV to model systematic on-die variation.

POCV advanced variation technology provides statistical benefits without the overhead of expensive statistical STA library characterization. POCV models instance delay as a function of a random variable that is specific to the instance. That is, the instance delay is parameterized as a function of the unique delay and its variation, shown in Figure 2.

Figure 2 POCV Modeling



POCV has the following features:

- Statistical single-parameter derating for random variations
- Single input format and characterization source for both AOCV and POCV table data
- Regular timing reports
- Statistical reporting (mean, sigma) for timing paths
- Compatibility with existing PrimeTime functionality

POCV Analysis

POCV is an enhanced variation methodology over AOCV. Instead of applying a specific derating factor to an individual instance, POCV models instance delay as a function of a random variable which is specific to this instance. In other words, the instance delay is parameterized as a function of this random variable.

$$Delay = Delay_{\text{nom}} + Delay_{\text{var}} \cdot P \qquad (1)$$

In equation 1, $Delay_{nom}$ is the delay of the cell, and $Delay_{var}$ is the one sigma value of the cell's delay distribution and P is the standard normal random variable N(0,1).

The variation in POCV is a function of the library cell. Its dependency on number of stages is handled during timing analysis. The value of delay variation can be easily

obtained from the data used to generate depth-based AOCV tables. Once delay variation information is provided, PrimeTime propagates the delay distributions through the timing graph using a generic value-enabled timing engine to calculate arrival time, required time, and slack, as shown in Figure 3.

Figure 3 Delay distributions are propagated through the timing graph as independent variables

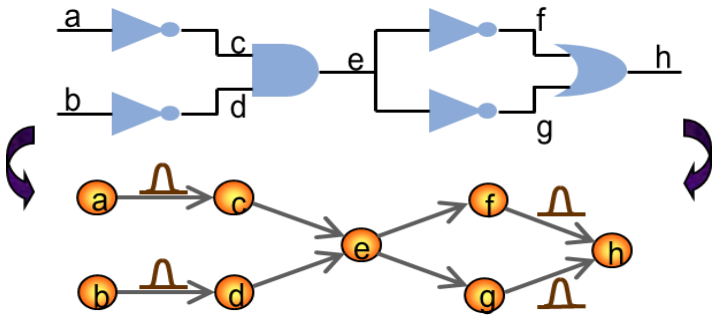
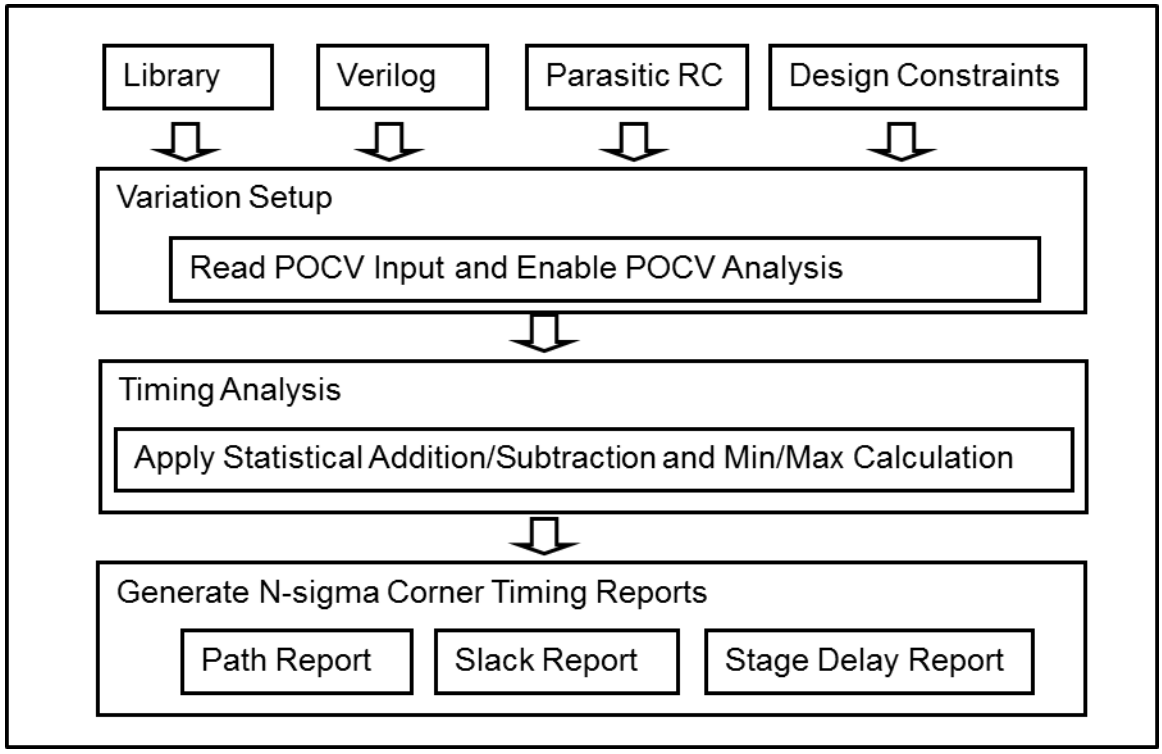


Figure 4 shows the POCV analysis and reporting flow. POCV analysis is embedded in the regular timing analysis flow. It reads variation information from either an AOCV-like table or the library, and then applies that information to the corresponding design, hierarchical cells, and library cells. During analysis, it propagates both nominal and random variations across the timing graph. At the end of analysis, all arrival times and slacks have a random variation component. During the reporting, the variation data is evaluated at a specific sigma corner (default is 3-sigma), and a corner-type timing report is generated.

Figure 4 POCV Analysis Flow



POCV Input Data

There are two forms of POCV input data that you can feed into PrimeTime:

- A side file with POCV single coefficient
- A library with POCV slew-load table per timing arc

A side file with POCV single coefficient applied on a library cell has higher precedence than POCV slew-load table.

POCV Input from Side File (Single Coefficient)

POCV extends the AOCV table format in version 4.0. With the extended format, you can specify POCV and AOCV distance information. The AOCV table format version 4.0 has the following changes:

- New `ocvm_type` field, which specifies the OCV methodology type; allowed values are `aocvm` and `pocvm`.
- New `coefficient` field, which specifies the random variation coefficient (sigma).
- If you specify `ocvm_type: pocvm`,
  - You cannot specify the `depth` field
  - The `coefficient` and `distance` fields are mutually exclusive

You need to specify different tables for POCV coefficients (random variation) and distance-based variation. An example is described in Table 1.

Table 1 Coefficient and Distance-Based POCV Tables

Variation type	POCV	Example
Coefficient on cell delay	Library cells	version : 4.0 ocvm_type : pocvm object_type: lib_cell rf_type : rise fall delay_type : cell derate_type: early object_spec: lib28nm/invx* coefficient: 0.05
Coefficient on net delay	Design level	version : 4.0 ocvm_type : pocvm object_type: design rf_type : rise fall delay_type : net derate_type: late object_spec: coefficient: 0.05
Distance-based	Design level	version : 4.0 ocvm_type : pocvm object_type: design rf_type : rise fall delay_type : cell derate_type: early object_spec: distance : 1 10 50 100 500 table : 0.9911 0.9116 0.9035 0.8917 0.8718

For more details on all the syntax in POCV tables, please refer to Appendix POCV Side File Formats for Single Coefficient.

POCV coefficients for a library cell can be extracted from Monte-Carlo HSPICE simulation using the following equation.

$$POCV\ Coefficient = \frac{\sigma\ (delay\ variation)}{\mu\ (nominal\ delay)}$$

The POCV data generation is usually faster than AOCV data generation. Monte-Carlo simulation has to be set up on a long chain of cells for AOCV, while it is limited to only a single or few stages for POCV data generation.

For the detailed syntax of POCV Library Variation Format (LVF), see POCV Input in Library Variation Format.

POCV Characterization

For POCV input characterization, Synopsys offers SiliconSmart, a library characterization tool supporting POCV data generation.

Silicon Smart offers the following:

- Characterizing POCV a slew-load table per timing arc in Liberty syntax

- Characterizing POCV a single coefficient per cell in side-file format
- 100% PrimeTime compliant POCV flow and methodology

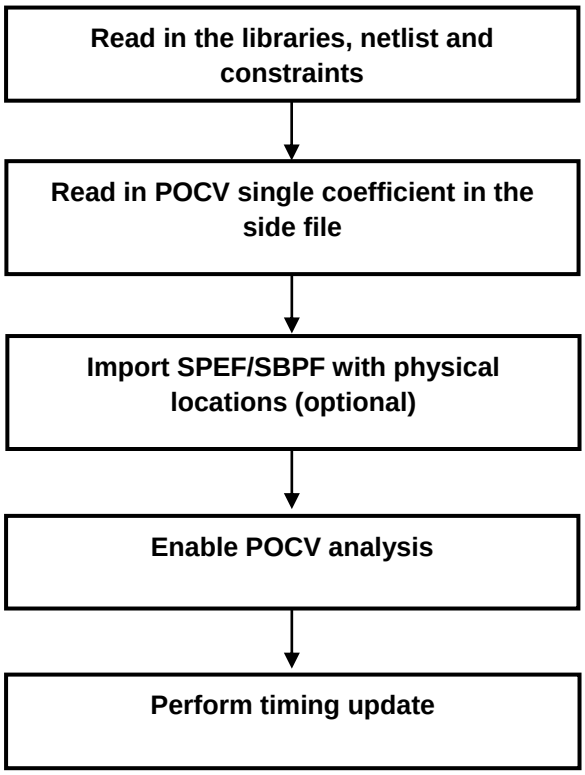
Distance-based derating is usually generated from silicon data measurements from the test chip. No known characterization tool can generate the distance-based table for spatial systematic variation.

POCV Coefficient (Side File) Flow

The POCV flow consists of setup, analysis, and reporting phases. In the setup phase, you need to read in the design, load the parasitics with coordinate information (optional, only if using distance-based derating), apply the constraints, enable POCV analysis, and specify the POCV information. In the analysis phase, PrimeTime performs a timing update based on the design input and POCV input data. The reporting phase includes reporting POCV derating- and analysis results.

Set Up POCV Analysis

To set up POCV analysis, perform the following steps:



Import SPEF/SBPF with Physical Locations (optional)

Location coordinates are required to calculate path distance but are not required to calculate the path depth. Coordinates only needed if the distance-based derating tables are applied in POCV analysis. To read the coordinates of various nodes of nets, pins, and ports into PrimeTime from files in the Standard Parasitic Exchange Format (SPEF), Galaxy Parasitic Database (GPD) or Synopsys Binary Parasitics Format (SBPF) file, set the `read_parasitics_load_locations` variable to `true`, and then use the `read_parasitics` command:

```
pt_shell> set_app_var read_parasitics_load_locations true
pt_shell> read_parasitics …
```

Enabling POCV Analysis

To enable graph-based POCV analysis, set the `timing_pocvm_enable_analysis` variable to `true`:

```
pt_shell> set_app_var timing_pocvm_enable_analysis true
```

PrimeTime performs graph-based POCV timing updates automatically as part of the `update_timing` command. By default, POCV analysis uses corner values taken at 3 sigma. To change this value for more conservative analysis, you can set the variable `timing_pocvm_corner_sigma` to the value larger than 3.

```
pt_shell> set_app_var timing_pocvm_corner_sigma 3
```

Loading POCV Information

In order to load POCV single-coefficient information or POCV distance-based derating table from a text file, use the `read_ocvm` command (similar as loading AOCV tables in AOCV flow):

```
pt_shell> read_ocvm pocv_coefficient_file_name
pt_shell> read_ocvm pocv_distance_based_derating_file_name
```

You must specify the coefficient or derating factors in a table using the Synopsys AOCV file format version 4.0 or later, as described in the Appendix POCV Side File Format for Single Coefficient. PrimeTime applies the POCV coefficient values in the file directly to the instance delays. If POCV tables have incorrect syntax, PrimeTime issues an error message, and the `read_ocvm` command does not do annotation.

PrimeTime annotates the POCV tables onto one or more design objects: library cells, hierarchical cells, and designs. For a cell, the precedence of data tables, from the lowest to highest, is the design POCV table, hierarchical cell POCV table, and library cell POCV

table. For a net, the precedence from lowest to highest is the design POCV table and hierarchical cell POCV table.

For POCV distance-based derating, the algorithm is exactly the same as distance-based derating in AOCV analysis. For more details how bounding box is calculated in graph-based analysis (GBA) and path-based analysis (PBA) in order to extract distance-based derating in POCV analysis, please refer to AOCV Application Notes on SolvNet.

Advanced On-Chip Variation (AOCV) in PrimeTime

Guard-banding in POCV

Similar to AOCV flow, guard-banding allows you to model non-process related effects in POCV flow. To perform guard-banding in POCV, specify the `set_timing_derate` command with the `-pocvm_guardband` option.

For example, the following command applies a 5% guard-band POCV derating on both early and late mode:

```
pt_shell> set_timing_derate -cell_delay \
          -pocvm_guardband -early 0.95
pt_shell> set_timing_derate -cell_delay \
          -pocvm_guardband -late 1.05
```

The POCV guard-band derating factor is applied on both the nominal cell delay (mean) and cell delay variation (sigma).

To report only the guard-band derating factors, you must specify the `report_timing_derate` command with the `-pocvm_guardband` option. To reset guard-band derating factors, use the `reset_timing_derate` command with the `-pocvm_guardband` option.

Scaling POCV Coefficient

In addition to guard-banding, the POCV flow also offers a way to scale only the variation of cell delay without changing the coefficient in POCV tables. To scale the POCV coefficient, specify the `set_timing_derate` command with the `-pocvm_coefficient_scale_factor` option.

For example, the following command scales the POCV coefficient by 3% for both early and late derating:

```
pt_shell> set_timing_derate -cell_delay \
          -pocvm_coefficient_scale_factor -early 1.03
pt_shell> set_timing_derate -cell_delay \
          -pocvm_coefficient -late 1.03
```

The POCV coefficient scaling factor is applied only on the variation of cell delay (sigma).

To report only POCV coefficient scaling factors, you must specify the `report_timing_derate` command with the `-pocvm_coefficient_scale_factor` option. To reset guard-band derating factors, use the `reset_timing_derate` command with the `-pocvm_coefficient_scale_factor` option.

Running PBA Analysis with Derate-Only Mode

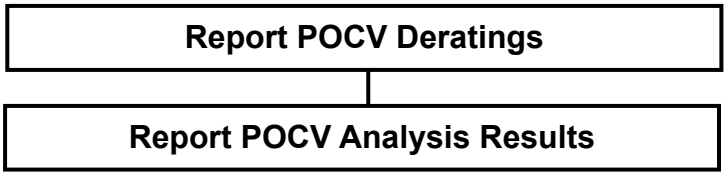
By default, PrimeTime performs both regular path-based analysis (path-specific slew propagation) and path-based POCV analysis during a path-based analysis (`report_timing` or `get_timing_paths` using the `-pba_mode` option), which removes the maximum amount of pessimism from the design, but the runtime can be large.

You can set the `pba_derate_only_mode` variable to `true` to perform path-based POCV analysis only. The analysis runtime is significantly improved at the cost of some pessimism removal.

```
pt_shell> set_app_var pba_derate_only_mode true
```

POCV Reporting

The POCV reporting phase has the following steps:



Reporting POCV Coefficient and Deratings

In order to display POCV information, including POCV coefficient and distance-based derating table data, use the `report_ocvm -type pocvm` command. This command shows design objects annotated with early, late, rise, fall, cell, net coefficient or derating tables and can also shows the annotation information for leaf cells and nets.

If the design has been annotated with POCV tables, using the `report_ocvm -type pocvm` command displays the POCV coefficient or distance-based derating factors annotated as shown in Figure 5.

```
pt_shell> report_ocvm -type pocvm -cell_delay \
 -list_not_annotated -coefficient
```

Figure 5 Example the output of report_ocvm –type pocvm command with POCV side file

```
pt_shell> report_ocvm -type pocvm [get_cell I]

****************************************************
POCV Table Set                :       *Default*
****************************************************
POCV coefficient: 0.0500
Name  Object  Process  Voltage  Sense  Path     Delay  Inherited
-------------------------------------------------------------------------------
I     cell    early    *        rise   clock    cell   lib1/INV1
I     cell    early    *        fall   clock    cell   lib1/INV1
I     cell    early    *        rise   data     cell   lib1/INV1
I     cell    early    *        fall   data     cell   lib1/INV1
****************************************************
POCV Table Set                :       *Default*
****************************************************
POCV coefficient: 0.0500
Name  Object  Process  Voltage  Sense  Path     Delay  Inherited
-------------------------------------------------------------------------------
I     cell    late     *        rise   clock    cell   lib1/INV1
I     cell    late     *        fall   clock    cell   lib1/INV1
I     cell    late     *        rise   data     cell   lib1/INV1
I     cell    late     *        fall   data     cell   lib1/INV1
```

To report all cells missing POCV coefficient and distance-based derating data, you can do the following:

```
pt_shell> report_ocvm –type pocvm –cell_delay –list_not_annotated
```

To report all cells only missing POCV coefficient data only (when you did not apply distance-based derating table in POCV analysis), you can do the following:

```
pt_shell> report_ocvm –type pocvm –cell_delay -list_not_annotated \-
coefficient
```

You can use the `report_ocvm` command before `update_timing` to check if there are any cells missing POCV coefficient or distance-based derating table.

Reporting POCV Analysis Results

POCV analysis is supported by all reporting commands in PrimeTime. By default, PrimeTime reports the timing at the 3-sigma value.

To display the variation in the timing report for POCV analysis, you can do `report_timing -derate` command, which displays the "Mean" column and the "Sensit" column. The "Mean" column is the nominal cell delay, i.e. the delay without POCV or without variation. The "Sensit" column is the sensitivity or the variation of cell delay due to the POCV coefficient.

```
pt_shell> report_timing -derate
```

By default, `report_timing -derate` command does not show "Mean" and "Sensit" column. To see these columns, set the variable `variation_report_timing_increment_format` to `delay_variation`.

```
pt_shell> set_app_var variation_report_timing_increment_format \
 delay_variation
```

Figure 6 shows an example of the output of the `report_timing -derate` command.

Figure 6 The output of the report_timing –derate command

```
pt_shell> report_timing -derate -sign 5 -nosplit
  Startpoint: i (input port clocked by CLK)
  Endpoint: o (output port clocked by CLK)
  Path Group: CLK
  Path Type: max
  Point                          Derate  Mean     Sensit   Incr       Path
  ------------------------------------------------------------------------
  clock CLK (rise edge)                                    0.00000    0.00000
  clock network delay (ideal)                              0.00000    0.00000
  input external delay                                     0.00000    0.00000 r
  i (in)                                 0.00000 0.00000 0.00000    0.00000 r
  I/I (INV1)                      1.00000 0.00000 0.00000 0.00000    0.00000 r
  I/ZN (INV1)                     1.02000 0.00975 0.00049 0.01121 &  0.01121 f
  o (out)                         1.00000 0.00007 0.00000 0.00007 &  0.01128 f
  data arrival time                                                  0.01128
  clock CLK (rise edge)                                    1.00000    1.00000
  clock network delay (ideal)                              0.00000    1.00000
  output external delay                                    0.00000    1.00000
  data required time                     1.00000 0.00000             1.00000
  ------------------------------------------------------------------------
  data required time                     1.00000 0.00000             1.00000
  data arrival time                      -0.00982 0.00049           -0.01128
  ------------------------------------------------------------------------
  slack (MET)                            0.99018 0.00049             0.98872
```

The "Incr" column is calculated from "Mean" and "Sensit" column using the equation below.

$$Incr = Mean \pm K * \frac{(Sensit_{cell})^2}{Sensit_{slack}}$$

where K is the value of `timing_pocvm_report_sigma`. The default is 3. If slack sensit is zero, the `Incr` value equals the `Mean` value.

PrimeTime also supports the `report_timing -variation` command, which only displays the "Mean" and "Sensit" columns, without the "Derate"column.

Figure 6b shows an example of the output of `report_timing -variation` command.

Figure 6b The output of report_timing –variation command

```
pt_shell> report_timing -variation -sign 5 -nosplit
  Startpoint: i (input port clocked by CLK)
  Endpoint: o (output port clocked by CLK)
  Path Group: CLK
  Path Type: max
  Point                          Mean    Sensit  Incr       Path
  ------------------------------------------------------------------------------
  clock CLK (rise edge)                                0.00000    0.00000
  clock network delay (ideal)                          0.00000    0.00000
  input external delay                                 0.00000    0.00000 r
  i (in)                         0.00000 0.00000 0.00000    0.00000 r
  I/I (INV1)                      0.00000 0.00000 0.00000    0.00000 r
  I/ZN (INV1)                    0.00975 0.00049 0.01121 &  0.01121 f
  o (out)                        0.00007 0.00000 0.00007 &  0.01128 f
  data arrival time                                         0.01128
  clock CLK (rise edge)                                1.00000    1.00000
  clock network delay (ideal)                          0.00000    1.00000
  output external delay                                0.00000    1.00000
  data required time                                   0.00000    1.00000
  ------------------------------------------------------------------------------
  data required time           1.00000 0.00000            1.00000
  data arrival time           -0.00982 0.00049           -0.01128
  ------------------------------------------------------------------------------
  slack (MET)                   0.99018 0.00049            0.98872
```

As you can see, the incremental delay is normalized with slack sensitivity. In some cases, where slack sensitivity is very small and close to zero, the incremental delay can become an extremely large value (for max path) or large negative value (for min path).

PrimeTime also supports a type of reporting where incremental delay is not normalized with slack sensitivity. With non-normalized reporting, the incremental delay is more realistic. As a result, the arrival time in timing report is also more accurate for ECO.

Non-normalized reporting is a default starting from PrimeTime 2016.12 release. For PrimeTime version prior to 2016.12, to enable this reporting mode, set the following variable:

```
pt_shell> set_app_var timing_pocvm_use_normalized_reporting false
```

The slack `Mean` and `Sensit` columns are the same as when the variable is set setting this variable to `true`. The cell delay `Mean` and `Sensit` are also identical. The only things changed are the `Incr` column and `Path` column. A `statistical adjustment` line is added at the end to make the numbers added up.

Figure 6c shows an example of the output of the `report_timing -variation` command with non-normalized reporting.

Figure 6c The output of report_timing –variation command with non-normalized reporting

```
pt_shell> report_timing -variation -sign 5 -nosplit
  Startpoint: i (input port clocked by CLK)
  Endpoint: o (output port clocked by CLK)
  Path Group: CLK
  Path Type: max
  Point                          Mean     Sensit  Incr       Path
  ---------------------------------------------------------------------------
  clock CLK (rise edge)                                0.00000    0.00000
  clock network delay (ideal)                          0.00000    0.00000
  input external delay                                 0.00000    0.00000 r
  i (in)                         0.00000 0.00000 0.00000    0.00000 r
  I/I (INV1)                      0.00000 0.00000 0.00000     0.00000 r
  I/ZN (INV1)                    0.00975 0.00049 0.01121 &  0.01121 f
  o (out)                        0.00007 0.00000 0.00007 &  0.01128 f
  data arrival time                                         0.01128
  clock CLK (rise edge)                                1.00000    1.00000
  clock network delay (ideal)                          0.00000    1.00000
  output external delay                                0.00000    1.00000
  data required time                                   0.00000    1.00000
  ---------------------------------------------------------------------------
  data required time              1.00000 0.00000             1.00000
  data arrival time              -0.00982 0.00049            -0.01128
  ---------------------------------------------------------------------------
  statistical adjustment                               0.00000    0.98872
  slack (MET)                     0.99018 0.00049             0.98872
```

For non-normalized reporting, the `Incr` column is calculated as

$$Incr = PathArrival_{current} - PathArrival_{previous}$$

$$PathArrival = PathArrival_{mean} \pm K * PathArrival_{sensit}$$

where

K is the value of the `timing_pocvm_report_sigma` variable. The default is 3.

(+) max mode

(-) min mode

`Statistical adjustment` is calculated as

$$statistical\ adjustment = slack - unadjusted\ slack$$

where

$$slack = slack_{mean} - slack_{sensit}$$

$$unadjusted\ slack = required\ time - arrival\ time$$

Starting from PrimeTime M-2016.12, the output of `report_timing -variation` has two separate sections for "`Incr`" and "`Path`". There are three new columns added.

The "`Corner`" column in the "`Incr`" section displays the corner value, i.e.

```
Mean +/- N * Sensit
```

where N is the value of `timing_pocvm_report_sigma`. This column helps you validate some `timing_path` attributes (which are using the corner value) with `report_timing` such as CRPR `common_path_pessimism`, `endpoint_setup_time_value` and `endpoint_hold_time_value`.

The "`Mean`" and "`Sensit`" column in the "`Path`" section display cumulative values at that particular pin.

For the "`Mean`" column, it is simple addition of all mean stage delays up to that point, i.e.

mean_1st_stage + mean_2nd_stage + ... + mean_previous_stage + mean_current_stage

For the "`Sensit`" column, it is simple RSS (Root Sum Square) of all sigma stage delay values up to that point, i.e.

sqrt(sensit_1st_stage^2 + sensit_2nd_stage^2+ .... + sensit_previous_stage^2 + sensit_current_stage^2)

These two new columns help you validate the path arrival up to that point. Path arrival is the corner value

```
Mean +/- N * Sensit
```

where N is the value of `timing_pocvm_report_sigma`.

Figure 6d shows an example of the output of `report_timing -variation` command using PrimeTime M-2016.12.

Figure 6d The output of report_timing –variation changes in M-2016.12

```
pt_shell> report_timing -variation -delay_type max_rise
  Startpoint: in1 (input port clocked by clk)
  Endpoint: out1 (output port clocked by clk)
  Path Group: clk
  Path Type: max
  Sigma: 3.0

                                          -------- Incr ----------------  -------- Path ----------
 Point                                     Mean     Sensit  Corner   Value    Mean     Sensit     Value
 --------------------------------------------------------------------------------------------------------
 clock clk (rise edge)                    0.00                      0.00    0.00                  0.00
 clock network delay (ideal)              0.00     0.00      0.00    0.00    0.00     0.00         0.00
 input external delay                     0.00                      0.00    0.00                  0.00 r
 in1 (in)                                 0.00     0.00      0.00    0.00    0.00     0.00         0.00 r
 b1/Z (NR3)                               1.00     3.00     10.00   10.00    1.00     3.00 H      10.00 r
 statistical graph pessimism              3.00    -2.24              0.00    4.00     2.00        10.00
 b3/Z (NR3)                               1.00     2.00      7.00    3.49    5.00     2.83 H      13.49 r
 statistical graph pessimism              5.00    -2.58              0.00   10.00     1.16        13.49
 out1 (out)                               0.00     0.00      0.00    0.00   10.00     1.16        13.49 r
 data arrival time                                                                                13.49

 clock clk (rise edge)                    2.00                      2.00    2.00                  2.00
 clock network delay (ideal)              0.00     0.00      0.00    0.00    0.00     0.00         2.00
 clock reconvergence pessimism            0.00     0.00      0.00    0.00    0.00     0.00         2.00
 output external delay                                               0.00                         2.00
 data required time                                                             2.00     0.00        2.00
 --------------------------------------------------------------------------------------------------------
 data required time                                                             2.00     0.00        2.00
 data arrival time                                                            -10.00     1.16      -13.49
 --------------------------------------------------------------------------------------------------------
 statistical adjustment                                              0.00                        -11.49
 slack (VIOLATED)                                                               -8.00     1.16      -11.49
```

To show the details of how PrimeTime calculates the final derating factor from all derating factors applied in POCV analysis, use the `report_delay_calculation -derate` command. It reports in detail how the final derating values are derived for cell delay and cell delay sigma as shown in Figure 7.

Figure 7 The output of report_delay_calculation –derate command with POCV side file

```
pt_shell> report_delay_calculation -from I/I -to I/ZN -derate
...
   Advanced driver-modeling used for rise and fall.
                              Rise         Fall
   ---------------------------------------------------------------
   Input transition time  = 0.011600    0.011600  (in library unit)
   Effective capacitance  = 0.002000    0.002000  (in pF)
   Effective capacitance  = 0.002000    0.002000  (in library unit)
   Output transition time = 0.007237    0.006342  (in library unit)
   Cell delay             = 0.008886    0.009559  (in library unit)

   POCVM coefficient       = 0.050000    0.050000
   POCVM coef scale factor = 1.000000    1.000000
   POCVM distance derate   = 1.000000    1.000000
   POCVM guardband         = 1.020000    1.020000
   Incremental derate      = 0.000000    0.000000
   Cell delay derated      = 0.009064    0.009750  (in library unit)
   Cell delay sigma        = 0.000453    0.000488  (in library unit)

   Cell  delay derated = "Cell delay" * ( "POCVM guardband" * "POCVM distance
derate" + "Incremental derate" )
   Cell  delay sigma   = "Cell delay" * ( "POCVM guardband" * "POCVM coefficient"
* "POCVM coef scale factor" )
```

As you can see in Figure 7, `report_delay_calculation -derate` command displays equations that show how the cell delay and cell delay variation are derated by applying POCV guard-banding, POCV distance-based derating, POCV coefficient, POCV coefficient scaling factor and incremental derate.

POCV Slack and Arrival Attributes

PrimeTime has slack and arrival attributes on the pin, path, and timing point to cover the mean and variation of the cell delay in POCV analysis. Table 2 shows the list of POCV slack and arrival attributes.

Table 2 POCV Slack and Arrival Attributes

Attribute	Object types
`max_fall_variation_slack`	Pin
`max_rise_variation_slack`	Pin
`min_fall_variation_slack`	Pin
`min_rise_variation_slack`	Pin
`max_fall_variation_arrival`	Pin
`max_rise_variation_arrival`	Pin
`min_fall_variation_arrival`	Pin
`min_rise_variation_arrival`	Pin
`variation_slack`	Path, path point
`variation_arrival`	Path, path point

POCV slack and arrival attributes are statistical quantities, i.e. they have mean and standard deviation. If you query POCV slack and arrival attributes directly, it returns an empty list. You can query the `mean` or `std_dev` (standard deviation) attribute of POCV slack and arrival attribute; it returns a floating-point value as shown in Figure 8.

Figure 8 Example of how to query POCV slack and arrival attribute

```
pt_shell> get_attribute [get_pin I/ZN] max_fall_variation_slack
{}
pt_shell> get_attribute [get_attribute [get_pin I/ZN] max_fall_variation_slack] \
          mean
0.990180
pt_shell> get_attribute [get_attribute [get_pin I/ZN] max_fall_variation_slack] \
          std_dev
0.000488
pt_shell> set path [get_timing_path -fall_through I/ZN]
_sel96
pt_shell> get_attribute $path variation_slack
{}
pt_shell> get_attribute $path variation_slack.mean
0.990180
pt_shell> get_attribute $path variation_slack.std_dev
0.000488
```

POCV User Interface

```
set timing_pocvm_enable_analysis ""   # Enables POCV analysis
set timing_pocvm_corner_sigma ""     # Set corner sigma for POCV analysis
set timing_pocvm_report_sigma ""     # Set report sigma for reporting
read_ocvm pocvm_file          # Read POCV tables
set_timing_derate -pocvm_guardband    # POCV guard-banding
set_timing_derate -pocvm_coefficient_scale_factor # POCV coefficient scaling
report_timing -derate           # Report POCV information in timing report
report_delay_calculation -derate # Report details how delays are derated
report_ocvm -type pocvm               # Report POCV annotation
report_timing_derate -pocvm_guardband # Report POCV guard-band
report_timing_derate -pocvm_coefficient_scale_factor # Report POCV scaling
reset_timing_derate -pocvm_guardband  # Remove POCV guard-band
reset_timing_derate -pocvm_coefficient_scale_factor # Remove POCV scaling
```

How to Convert AOCV to POCV Side File

If you are currently using the PrimeTime AOCV flow and AOCV data are available for all library cells, you can quickly convert AOCV to POCV side files (single coefficient).

Just convert the first entry (i.e. the depth is 1 and distance is 0) in the table of your existing AOCV data to POCV coefficient using the following equations.

For late table:

$$POCV\ Coefficient = \frac{AOCV_{derate} - 1}{3}$$

For early table:

$$POCV\ Coefficient = \frac{1 - AOCV_{derate}}{3}$$

These equations are based on the assumption that the first entry in AOCV table is the derating factor purely due to process variation, and no other variation lumped to that number.

The POCV table format is similar to the AOCV table with a few changes. Please refer to in the Appendix POCV Side File Format for Single Coefficient for details.

POCV Results

The advantages of using POCV analysis instead of AOCV analysis include:

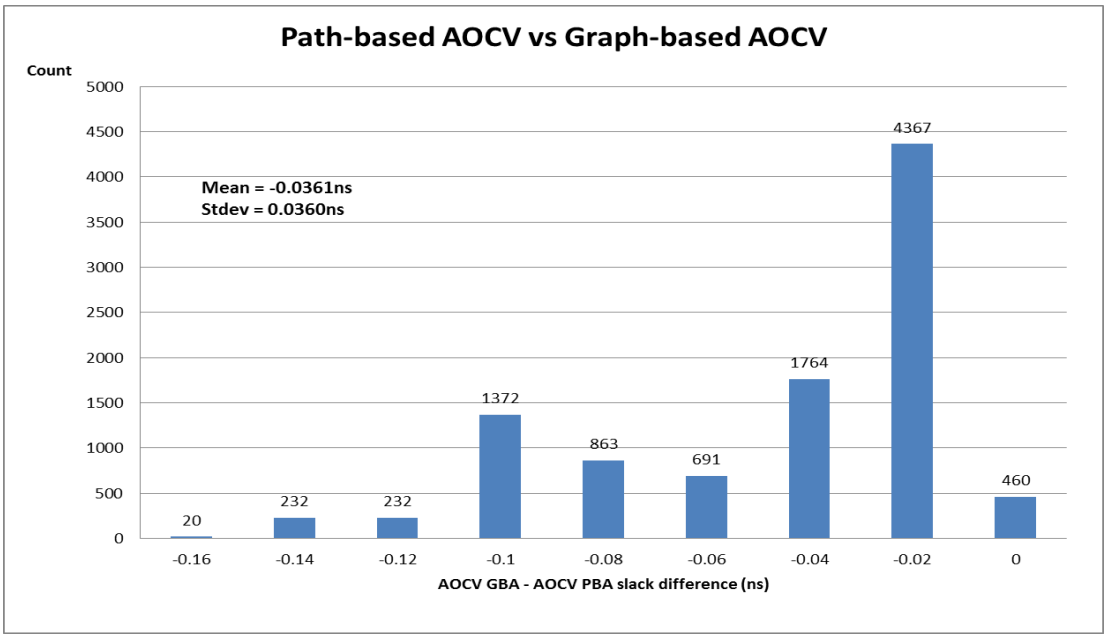
- Tighter correlation between graph-based and path-based analysis
- Reducing pessimism in graph-based analysis compared to AOCV
- Faster incremental update timing
- Faster TAT in ECO fixing

POCV – AOCV Timing Comparison

Since AOCV uses path depth to derive the derating factor and the path depth in graph-based analysis is pessimistic comparing to the one in path-based analysis. POCV can remove that pessimism since it does not depend on path depth calculation to compute path variation.

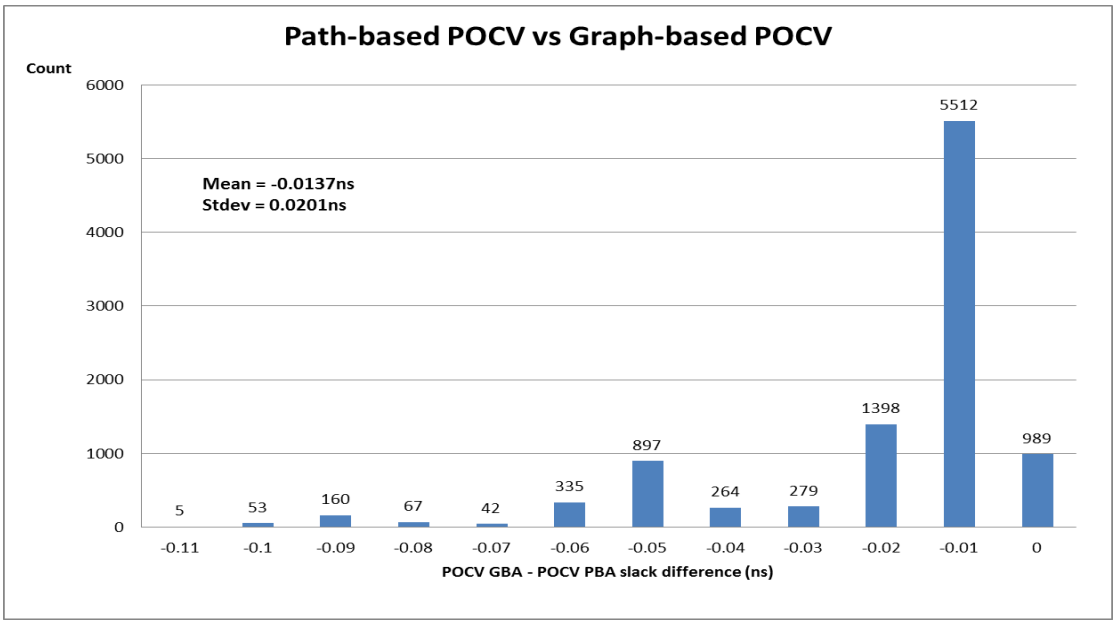
Figure 9 shows an example of path-based AOCV analysis compared to graph-based AOCV analysis.

Figure 9 Histogram of path-based AOCV versus graph-based AOCV



The equivalent POCV coefficient is converted from the AOCV derating table and applied to the design. Figure 10 shows a comparison between path-based POCV and graph-based POCV using the same endpoints as Figure 9.

Figure 10 Histogram of path-based POCV versus graph-based POCV



As you can see from the mean of the slack difference between graph-based slack and path-based slack, it is much tighter in POCV analysis compared to AOCV analysis.

POCV – AOCV ECO Comparison

Since POCV timing is less pessimistic than AOCV timing, there is less number of violations to fix in graph-based analysis. In addition, incremental update timing is also faster with POCV. As a result, the TAT is faster in POCV for ECO fixing. In addition, the area increased due to ECO fixing is less in POCV with compatible memory usage and fixed rate.

Table 3 shows an example of how POCV ECO comparing with AOCV ECO. As you can see, the ECO runtime is faster and the total increase in area is smaller for POCV analysis compared to AOCV analysis.

Table 3: An Example of POCV ECO versus AOCV ECO

	POCV	***AOCV***	***AOCV/POCV***
Runtime (min)	254	366	1.44X
Memory (GB)	8.4	8.5	1.01X
Number of endpoint violations	408590	439829	1.08X
Fixed rate	100%	100%	1.00X
Total area increase (um2)	516997.28	554905.44	1.07X

POCV currently supports all features available in PrimeTime ECO.

POCV Liberty Variation Format (LVF) Flow

PrimeTime supports POCV Liberty Variation Format (LVF). The POCV data is represented as slew-load table per delay timing arc. LVF is a standardized format approved by the Liberty Technology Advisory Board (LTAB). It improves the accuracy of POCV analysis by taking into consideration multiple slew/load points for each cell.

For 16nm technology node and smaller, it is recommended to use POCV LVF to improve the accuracy of POCV analysis.

POCV Input in Liberty Variation Format (LVF)

You can load the library with POCV LVF into PrimeTime before linking the design. The unit of POCV LVF data is the time unit of the library. Figure 11 shows an example of POCV slew-load table in the library.

Figure 11 Example of POCV LVF slew-load table

```
ocv sigma cell rise ("delay template 7x7") {
  sigma type : "early";
  index 1("0.0088000, 0.0264000, 0.0608000, 0.1296000, 0.2672000, 0.5424000, 1.0936000");
  index_2("0.0010000, 0.0024000, 0.0052000, 0.0108000, 0.0221000, 0.0445000, 0.0895000");
    values("0.000476, 0.000677, 0.001075, 0.001870, 0.003438, 0.006626, 0.012922", \
    "0.000651, 0.000901, 0.001303, 0.002081, 0.003678, 0.006818, 0.013144", \
    "0.000840, 0.001166, 0.001714, 0.002558, 0.004112, 0.007249, 0.013529", \
    "0.001115, 0.001520, 0.002193, 0.003317, 0.005087, 0.008153, 0.014445", \
    "0.001521, 0.002033, 0.002883, 0.004242, 0.006522, 0.010072, 0.016258", \
    "0.002155, 0.002793, 0.003853, 0.005563, 0.008424, 0.012955, 0.020171", \
    "0.003204, 0.003977, 0.005321, 0.007515, 0.010960, 0.016582, 0.025786");
}

ocv sigma cell rise ("delay template 7x7") {
  sigma type : "late";
  index_1("0.0088000, 0.0264000, 0.0608000, 0.1296000, 0.2672000, 0.5424000, 1.0936000");
  index_2("0.0010000, 0.0024000, 0.0052000, 0.0108000, 0.0221000, 0.0445000, 0.0895000");
    values("0.000476, 0.000677, 0.001075, 0.001870, 0.003438, 0.006626, 0.012922", \
    "0.000651, 0.000901, 0.001303, 0.002081, 0.003678, 0.006818, 0.013144", \
    "0.000840, 0.001166, 0.001714, 0.002558, 0.004112, 0.007249, 0.013529", \
    "0.001115, 0.001520, 0.002193, 0.003317, 0.005087, 0.008153, 0.014445", \
    "0.001521, 0.002033, 0.002883, 0.004242, 0.006522, 0.010072, 0.016258", \
    "0.002155, 0.002793, 0.003853, 0.005563, 0.008424, 0.012955, 0.020171", \
    "0.003204, 0.003977, 0.005321, 0.007515, 0.010960, 0.016582, 0.025786");
}
```

PrimeTime supports distance-based derating in AOCV as a side file. It also supports distance-based derating in liberty variation format. PrimeTime automatically picks up the distance-based derating table if it exists in the library while linking the design. Figure 12 shows an example of LVF distance-based derating.

Figure 12 Example of POCV LVF distance-based derating

```
ocv table template("aocvm distance template") {
          variable_1 : path_distance;
          index_1 ("0, 10, 100, 1000");
        }

……
…..

       ocv_derate(aocvm_distance_design){
          ocv derate factors(aocvm distance template) {
            rf type : rise and fall;
            derate type : late;
            path type : clock and data;
            values ( "1.1, 1.2, 1.3, 1.4" );
          }
          ocv derate factors(aocvm distance template) {
            rf type : rise and fall;
            derate type : early;
            path type : clock and data;
            values ( "0.9, 0.8, 0.7, 0.6" );
          }
        }
        default ocv derate distance group : aocvm distance design;
```

For a detailed description of POCV Library Variation Format (LVF), please refer to the Appendix POCV Library Variation Format.

POCV LVF Characterization

For POCV input characterization, Synopsys offers SiliconSmart, a library characterization tool supporting POCV input data generation.

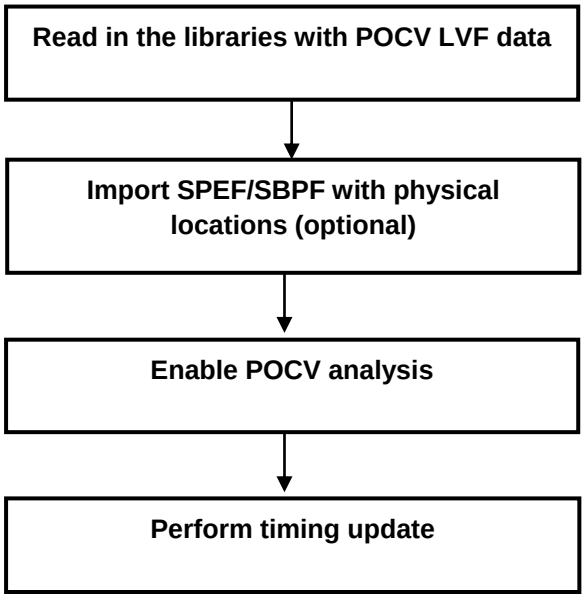
Silicon Smart offers the following:

- Characterizing POCV a slew-load table per timing arc in Liberty syntax
- Characterizing POCV a single coefficient per cell in side-file format
- 100% PrimeTime compliant POCV flow and methodology

To compile a library with POCV LVF into Synopsys *.db format, it is recommended that you use the latest Library Compiler.

POCV LVF Analysis

In order to set up POCV LVF analysis, perform the following steps:



Enabling POCV LVF Analysis

To enable graph-based POCV LVF analysis, set the `timing_pocvm_enable_analysis` variable to `true`:

```
pt_shell> set_app_var timing_pocvm_enable_analysis true
```

PrimeTime performs graph-based POCV timing updates automatically as part of the `update_timing` command. By default, POCV uses corner values taken at 3 sigma. To change this value for more conservative analysis, you can set the variable `timing_pocvm_corner_sigma` to the value larger than 3.

```
pt_shell> set_app_var timing_pocvm_corner_sigma 3
```

Loading POCV LVF Information

Loading the library with POCV LVF data is similar as loading regular timing library. Just specify it in the `search_path` and `link_path` variables. PrimeTime then automatically reads in POCV information from the library while linking the design.

You can apply both the library with POCV and the side file with POCV single-coefficient data in the same PrimeTime run. The delay variation is annotated accordingly depending on which POCV format is applied on the cell. If the library with POCV LVF and the side file with POCV single-coefficient data are applied on the same library cell, by default POCV single-coefficient data takes precedence. If a POCV side file is applied on the

design level, the POCV side file is used for all cells in the design, and POCV LVF is ignored.

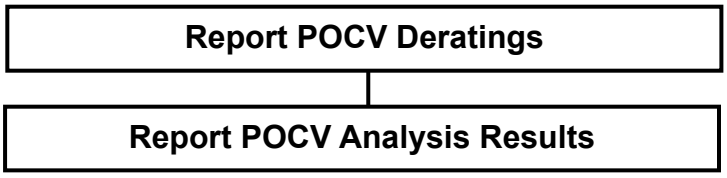
Starting from L-2016.06, PrimeTime has a variable to control the precedence between a POCV side file and LVF:

```
pt_shell> set_app_var timing_pocvm_precedence ...
```

- `file`: default - POCV side file always takes precedence regardless what level (i.e. design or lib_cell) that the POCV is applied.
- `library`: LVF always takes precedence when it is available.
- `lib_cell_in_file`: POCV side file overrides the LVF when it is applied on the same library cell.

POCV LVF Reporting

The POCV LVF reporting phase has the following steps:



Reporting POCV LVF

In order to display POCV information, including POCV coefficient and distance-based derating table data, use the `report_ocvm -type pocvm` command. This command shows design objects annotated with early, late, rise, fall, cell, net coefficient or derating tables and can also show the annotation information for leaf cells and nets.

For POCV LVF, since the slew-load table is defined on library timing arcs, you need to specify a library timing arc in the `object_list` to make it work. The `report_ocvm` command displays all the slew-load tables associated with the specified library timing arc. Figure 13 shows an example of the output of the `report_ocvm` command with POCV LVF.

Figure 13 Example of report_ocvm –type pocvm command with POCV LVF

```
pt_shell> report_ocvm -type pocvm [get_lib_timing_arcs -from */INVD1/I -to */INVD1/ZN]

min rise table
Sense / Type: negative unate

             | Slew
  Load       | 0.0010000  0.0024000  0.0052000  0.0108000  0.0221000  0.0445000  0.0895000
 ------------+-----------------------------------------------------------------------------
  0.0088000  | 0.0004760  0.0006770  0.0010750  0.0018700  0.0034380  0.0066260  0.0129220
  0.0264000  | 0.0006510  0.0009010  0.0013030  0.0020810  0.0036780  0.0068180  0.0131440
  0.0608000  | 0.0008400  0.0011660  0.0017140  0.0025580  0.0041120  0.0072490  0.0135290
  0.1296000  | 0.0011150  0.0015200  0.0021930  0.0033170  0.0050870  0.0081530  0.0144450
  0.2672000  | 0.0015210  0.0020330  0.0028830  0.0042420  0.0065220  0.0100720  0.0162580
  0.5424000  | 0.0021550  0.0027930  0.0038530  0.0055630  0.0084240  0.0129550  0.0201710
  1.0936000  | 0.0032040  0.0039770  0.0053210  0.0075150  0.0109600  0.0165820  0.0257860

min fall table
Sense / Type: negative unate

             | Slew
  Load       | 0.0010000  0.0024000  0.0052000  0.0108000  0.0221000  0.0445000  0.0895000
 ------------+-----------------------------------------------------------------------------
  0.0088000  | 0.0003780  0.0005140  0.0007840  0.0013200  0.0024020  0.0045370  0.0088430
  0.0264000  | 0.0004950  0.0006980  0.0010150  0.0015460  0.0026210  0.0047660  0.0090610
  0.0608000  | 0.0005730  0.0008580  0.0012960  0.0019850  0.0030780  0.0052090  0.0095170
  0.1296000  | 0.0006060  0.0009720  0.0015550  0.0024890  0.0039210  0.0061280  0.0103820
  0.2672000  | 0.0005040  0.0009800  0.0017400  0.0029410  0.0048280  0.0077760  0.0122220
  0.5424000  | 0.0001620  0.0007670  0.0017220  0.0033180  0.0057850  0.0095850  0.0154820
  1.0936000  | 0.0006780  0.0000330  0.0012030  0.0032310  0.0064090  0.0114070  0.0189420
```

As you can see, in addition to the data, each table displays the analysis mode as min (early) or max (late) and the sense type (negative_unate or positive_unate). If there are multiple library timing arcs with different input conditions, the *sdf_cond* is displayed as well.

To report all cells missing POCV annotation only (either side file or LVF), you can do the following:

```
pt_shell> report_ocvm –type pocvm –cell_delay \
          –list_not_annotated –coefficient
```

You can use the -lib_cell option to report all library cells missing POCV annotation.

```
pt_shell> report_ocvm –type pocvm –lib_cell –cell_delay \
          –list_not_annotated –coefficient
```

Reporting POCV Analysis Results

POCV analysis is supported by all reporting commands in PrimeTime. By default, PrimeTime reports the timing at the 3-sigma value.

To display the variation in the timing report for POCV analysis, you can do report_timing -derate command, which displays the "Mean" column and the "Sensit" column. The "Mean" column is the nominal cell delay, i.e. the delay without

POCV or variation. The "Sensit" column is the sensitivity or variation of cell delay due to the POCV coefficient.

```
pt_shell> report_timing -derate
```

By default, report_timing -derate command does not show the "Mean" and "Sensit" columns. To see these columns set the variable variation_report_timing_increment_format to delay_variation.

```
pt_shell> set_app_var variation_report_timing_increment_format \
 delay_variation
```

Figure 14 shows an example of the output of report_timing -derate.

Figure 14 The output of report_timing –derate command

```
pt_shell> report_timing -derate -sign 5 -nosplit
  Startpoint: i (input port clocked by CLK)
  Endpoint: o (output port clocked by CLK)
  Path Group: CLK
  Path Type: max
  Point                        Derate  Mean     Sensit   Incr       Path
  -----------------------------------------------------------------------
  clock CLK (rise edge)                                  0.00000    0.00000
  clock network delay (ideal)                            0.00000    0.00000
  input external delay                                   0.00000    0.00000 r
  i (in)                               0.00000 0.00000 0.00000    0.00000 r
  I/I (INV1)                    1.00000 0.00000 0.00000 0.00000    0.00000 r
  I/ZN (INV1)                   1.02000 0.00975 0.00049 0.01121 &  0.01121 f
  o (out)                       1.00000 0.00007 0.00000 0.00007 &  0.01128 f
  data arrival time                                                0.01128

clock CLK (rise edge)                                  1.00000    1.00000
  clock network delay (ideal)                          0.00000    1.00000
  output external delay                                0.00000    1.00000
  data required time                   1.00000 0.00000            1.00000
  -----------------------------------------------------------------------
  data required time                   1.00000 0.00000            1.00000
  data arrival time                    -0.00982 0.00049          -0.01128
  -----------------------------------------------------------------------
  slack (MET)                          0.99018 0.00049            0.98872
```

The "Incr" column is calculated from "Mean" and "Sensit" column using the following equation.

$$Incr = Mean \pm K * \frac{(Sensit_{cell})^2}{Sensit_{slack}}$$

where K is the value of `timing_pocvm_report_sigma`. The default is 3. If slack sensit is zero, the `Incr` value equals the `Mean` value.

In addition to `report_timing -derate`, PrimeTime also supports `report_timing -variation`, which is also supported by IC Compiler. The only difference between `-derate` and `-variation` is that `-derate` option shows "`Derate`" column and the `-variation` option does not. In addition, `-variation` option does not require the variable `variation_report_timing_increment_format` to be set to `delay_variation`.

Figure 14b shows an example of the output of `report_timing -variation` command

Figure 14b The output of report_timing –variation command

```
pt_shell> report_timing -variation -sign 5 -nosplit
  Startpoint: i (input port clocked by CLK)
  Endpoint: o (output port clocked by CLK)
  Path Group: CLK
  Path Type: max
  Point                          Mean     Sensit  Incr      Path
  ---------------------------------------------------------------------------
  clock CLK (rise edge)                             0.00000    0.00000
  clock network delay (ideal)                       0.00000    0.00000
  input external delay                              0.00000    0.00000 r
  i (in)                         0.00000 0.00000    0.00000    0.00000 r
  I/I (INV1)                     0.00000 0.00000    0.00000    0.00000 r
  I/ZN (INV1)                    0.00975 0.00049    0.01121 &  0.01121 f
  o (out)                        0.00007 0.00000    0.00007 &  0.01128 f
  data arrival time                                            0.01128

  clock CLK (rise edge)                             1.00000    1.00000
  clock network delay (ideal)                       0.00000    1.00000
  output external delay                             0.00000    1.00000
  data required time             1.00000 0.00000               1.00000
  ---------------------------------------------------------------------------
  data required time             1.00000  0.00000              1.00000
  data arrival time              -0.00982 0.00049             -0.01128
  ---------------------------------------------------------------------------
  slack (MET)                     0.99018 0.00049              0.98872
```

Starting from PrimeTime M-2016.12, the output of `report_timing -variation` has two separate sections for "`Incr`" and "`Path`" with each section having three new columns.

The "`Corner`" column in "`Incr`" section displays the corner value, i.e. `Mean +/- N * Sensit` where N is the value of `timing_pocvm_report_sigma`. This column helps you to validate `timing_path` attributes (which are using corner value) with report_timing such as CRPR `common_path_pessimism`, `endpoint_setup_time_value` and `endpoint_hold_time_value`, etc.

"`Mean`" and "`Sensit`" column in "`Path`" section display cumulative (accumulated path arrival and sigma) value at that particular pin.

For "Mean" column, it is simple addition of all mean stage delays up to that point, i.e. `mean_1st_stage + mean_2nd_stage + ……. + mean_previous_stage + mean_current_stage`

For "Sensit" column, it is simple RSS (Root Sum Square) of all sigma stage delay up to that point, i.e. `sqrt(sensit_1st_stage^2 + sensit_2nd_stage^2+ …. + sensit_previous_stage^2 + sensit_current_stage^2)`

These two new columns help you to validate the path arrival up to that point. Path arrival is corner value `Mean +/- N * Sensit` where N is the value of `timing_pocvm_report_sigma`.

Figure 14c shows an example of the output of `report_timing -variation` command using PrimeTime M-2016.12.

Figure 14c The output of report_timing –variation changes in M-2016.12

```
pt_shell> report_timing -variation -delay_type max_rise
  Startpoint: in1 (input port clocked by clk)
  Endpoint: out1 (output port clocked by clk)
  Path Group: clk
  Path Type: max
  Sigma: 3.0

                                          -------- Incr ----------------  -------- Path ----------
 Point                                  Mean     Sensit  Corner   Value    Mean     Sensit     Value
 --------------------------------------------------------------------------------------------------
 clock clk (rise edge)                  0.00                      0.00     0.00                 0.00
 clock network delay (ideal)            0.00     0.00     0.00    0.00     0.00     0.00        0.00
 input external delay                   0.00                      0.00     0.00                 0.00 r
 in1 (in)                               0.00     0.00     0.00    0.00     0.00     0.00        0.00 r
 b1/Z (NR3)                             1.00     3.00    10.00   10.00     1.00     3.00 H     10.00 r
 statistical graph pessimism            3.00    -2.24             0.00     4.00     2.00       10.00
 b3/Z (NR3)                             1.00     2.00     7.00    3.49     5.00     2.83 H     13.49 r
 statistical graph pessimism            5.00    -2.58             0.00    10.00     1.16       13.49
 out1 (out)                             0.00     0.00     0.00    0.00    10.00     1.16       13.49 r
 data arrival time                                                                              13.49

 clock clk (rise edge)                  2.00                      2.00     2.00                 2.00
 clock network delay (ideal)            0.00     0.00     0.00    0.00     0.00     0.00        2.00
 clock reconvergence pessimism          0.00     0.00     0.00    0.00     0.00     0.00        2.00
 output external delay                                             0.00                        2.00
 data required time                                                         2.00     0.00        2.00
 --------------------------------------------------------------------------------------------------
 data required time                                                         2.00     0.00        2.00
 data arrival time                                                        -10.00     1.16      -13.49
 --------------------------------------------------------------------------------------------------
 statistical adjustment                                            0.00                       -11.49
 slack (VIOLATED)                                                           -8.00     1.16      -11.49
```

In order to show the details how PrimeTime calculates the final derating factor from all derating factors applied in POCV analysis, use `report_delay_calculation -derate` command. Figure 15 shows the output of `report_delay_calculation -derate` when POCV slew-load table is applied.

Figure 15 The output of report_delay_calculation –derate command with POCV LVF

```
pt_shell> report_delay_calculation -from I/I -to I/ZN -derate
...


   Advanced driver-modeling used for rise and fall.
                              Rise        Fall
   ---------------------------------------------------------------------
   Input transition time  = 0.011600    0.011600  (in library unit)
   Effective capacitance  = 0.002000    0.002000  (in pF)
   Effective capacitance  = 0.002000    0.002000  (in library unit)
   Drive resistance       = 0.001000    0.001000  (in Kohm)
   Output transition time = 0.016620    0.011003  (in library unit)
   Cell delay             = 0.013041    0.010004  (in library unit)

   POCVM delay sigma       = 0.000652    0.000500
   POCVM coef scale factor = 1.000000    1.000000
   POCVM distance derate   = 1.000000    1.000000
   POCVM guardband         = 0.980000    0.980000
   Incremental derate      = 0.000000    0.000000
   Cell delay derated      = 0.012780    0.009804  (in library unit)
   Cell delay sigma        = 0.000639    0.000490  (in library unit)

   Cell  delay derated = "Cell delay" * ( "POCVM guardband" * "POCVM distance
derate" + "Incremental derate" )
   Cell  delay sigma   = "POCVM delay sigma" * ( "POCVM guardband" * "POCVM coef
scale factor" )
```

As you can see in Figure 15, `report_delay_calculation –derate` command displays the equations how the cell delay and cell delay variation are derated by applying POCV guard-banding, POCV distance-based derating, POCV LVF, POCV coefficient scaling factor and incremental derate.

How Slack Sensitivity is Calculated

Below are equations showing how slack sensitivity (or sigma slack) is derived from sigma of arrival time, required time, and clock reconvergence pessimism removal (CRPR).

```
sigma_slack = sqrt(sigma_arrival^2  +/-  sigma_required^2)
```

(+) if `sigma_required` is positive value

(-) if `sigma_required` is negative value

where

```
sigma_required^2 = sigma_capture_path^2 – sigma_crpr^2
```

where

```
sigma_crpr^2 = sigma_common_launch_path^2 + sigma_common_capture_path^2
```

Sigma of a path (`sigma_capture_path, sigma_common_launch_path` or `sigma_common_capture_path`) is calculated using RSS (Root Sum Square) of all cell delays along the path.

```
sigma_path = sqrt(sensit_cell_A^2 + sensit_cell_B^2 + sensit_cell_C^2
+ .....)
```

POCV LVF Accuracy

POCV LVF accuracy can be verified by performing the correlation with Monte-Carlo HSPICE. Figure 16 shows a typical path delay correlation between nominal (i.e. no POCV) path delay and Monte-Carlo HSPICE simulation.

Figure 16 The correlation of nominal (no POCV) path delay vs. HSPICE simulation

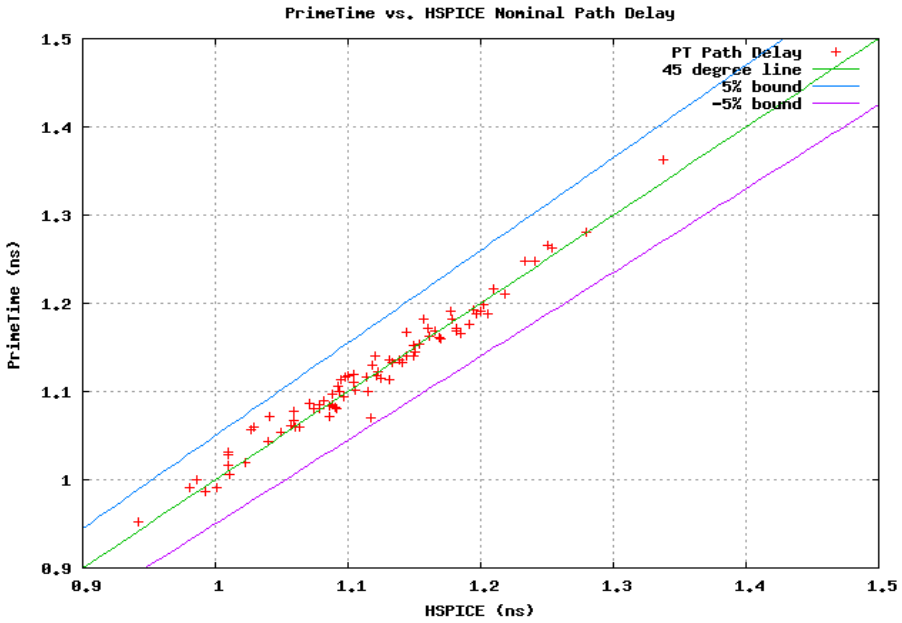
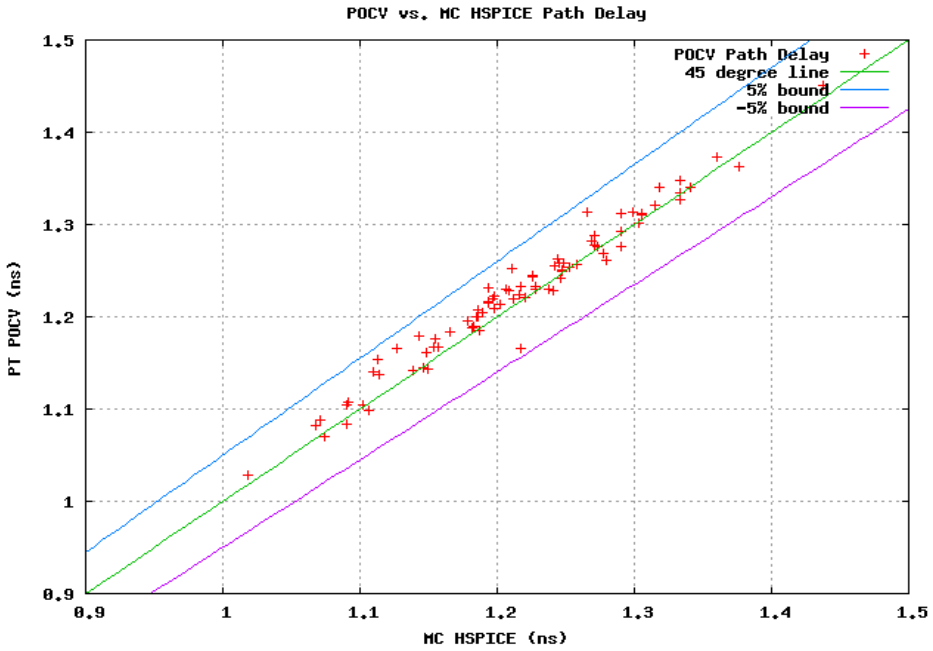


Figure 17 shows the path delay correlation between POCV LVF and Monte-Carlo HSPICE simulation on the same set of paths in Figure 16. The new path delay is calculated as `nominal delay +/- 3*sigma`. As you can see, after POCV analysis the path delay correlation is still good. POCV LVF path delay is bounded within +/-5% compared to Monte-Carlo HSPICE simulation.

Figure 17 The correlation of POCV LVF path delay vs. Monte-Carlo HSPICE simulation



POCV LVF – Monte Carlo Correlation

Monte-Carlo (MC) SPICE simulation can be run on path arrivals and its results can be compared to the results from PrimeTime POCV. After you have selected a set of paths for POCV-MC correlation, it is recommended to first run PrimeTime without POCV and compare with HSPICE results. Once that correlation is established, then you can move forward with POCV-MC correlation.

PrimeTime Without SI – HSPICE Correlation

Here is the recommended procedure to perform PrimeTime non-SI correlation with SPICE.

1. Run PrimeTime without SI analysis and select a set of paths for correlation.

   It is recommended to select register-to-register paths for the correlation to reduce Monte Carlo runtime for POCV correlation.

2. Use the `write_spice_deck` command in PrimeTime to generate SPICE decks for these selected paths.
3. Run HSPICE and collect path arrival times for all selected paths.
4. Run PrimeTime and collect path arrival times for all selected paths.
5. Compare path arrivals between PrimeTime and HSPICE simulation.

   Typically, the path arrival correlation is within +/-3%.

It is recommended to use the flow and scripts given in the following SolvNet article for PrimeTime – HSPICE correlation.

Automating PrimeTime and HSPICE Timing Path Correlation

PrimeTime POCV LVF – MC HSPICE Correlation

After you are done with PrimeTime – HSPICE correlation without SI, you can start performing PrimeTime POCV versus MC HSPICE correlation using the same set or subset of the selected paths in the delay correlation. If you don't have POCV LVF for the library cells, you can use POCV single-coefficient data. However, for better accuracy, POCV LVF should be used for PrimeTime-MC HSPICE correlation.

Here is the recommended procedure to perform POCV to MC SPICE correlation.

1. Select the same set or subset of the selected paths in PrimeTime-HSPICE correlation step.

   It is recommended to use register-to-register paths for PrimeTime POCV -MC HSPICE correlation.

2. Run PrimeTime analysis with SI and POCV disabled and use the `write_spice_deck` command in PrimeTime to generate SPICE decks for these selected paths.

   Use the `-user_measures` option in `write_spice_deck` to add the path delay measure statement in the SPICE deck. It is needed to collect path arrival mean and sigma data later in MC HSPICE run.

   Here is an example how to add the measure statement in the SPICE deck using the `-user_measures` option:

```
pt_shell> write_spice_deck [get_timing_path -from Uff1/CK \
                    -through U1/I -to Uff2/D] -user_measures \
          {{delay Uff1/CK U1/I stage_1_delay} {delay Uff1/CK Uff2/D path_delay}}
```

3. Modify the SPICE deck to add "`sweep monte = N`" in the "`.tran `" statement of the existing SPICE decks. N is number of MC samples. The number of MC samples has a big impact on HSPICE run time. The larger N is, the longer the run time. It is recommended to run MC with at least 5000 samples.

   Here is an example how to modify SPICE deck.

   Before modification:

```
* transient analysis
.tran 0.001ns 100ns
```

After modification:

```
* transient analysis
.tran 0.001ns 100ns sweep monte=10000
```

4. Run MC HSPICE simulation and collect path delay mean and sigma data.

   You can collect path delay data from that HSPICE mt0 file and process it to get mean and sigma from MC samples. For better accuracy, you should collect separate sigma values for early and late from MC samples in mt0 file. Sigma late is based on the difference between the nominal path delay and 3-sigma quantile value of 99.865%. Sigma early data is based on the difference between nominal path delay and 3-sigma quantile value of 0.135%.

```
set sort_samples [lsort -real $monte_carlo_samples]
set number_of_samples [llength $sort_samples]
set late_index [expr round(0.99865 * $number_of_samples) -1]
set early_index [expr round(0.00135 * $number_of_samples) -1]
set quantile_late [lindex $sort_samples $late_index]
set quantile_early [lindex $sort_samples $early_index]
set sigma_late [expr ($quantile_late - $nominal_delay)/3]
set sigma_early [expr ($nominal_delay - $quantile_early)/3]
```

5. Run PrimeTime POCV analysis and collect path delay mean and sigma data for each selected path.

   You can take advantage of POCV attributes on the path to collect path arrival mean and sigma data.

   Here is an example showing how to collect mean and sigma data for a path in PrimeTime POCV.

```
pt_shell> set path [get_timing_path –from … -to ….]
pt_shell> get_attribute [get_attribute $path variation_arrival] mean
1.990180
pt_shell> get_attribute [get_attribute $path variation_arrival] std_dev
0.054
```

6. Compare path arrival data between PrimeTime POCV and MC HSPICE simulation.

   For the max analysis, the path arrival is mean +3 sigma.

   For the min analysis, the path arrival is mean – 3 sigma.

   The correlation between POCV and MC HSPICE simulation should be within +/-5%.

POCV Transition Variation

PrimeTime supports POCV Liberty Variation Format (LVF) for transition variation. Similar to cell delay variation, the transition variation data is represented as a slew-load table per delay timing arc in liberty variation format. It improves the accuracy of POCV analysis by taking into consideration transition variation for each cell. In addition, the impact of transition variation on performance and capacity is small compared with only delay variation enabled.

For the 16nm technology node and smaller, it is recommended to use POCV LVF with both cell delay variation and transition variation to improve the accuracy of POCV analysis.

LVF Format for Transition Variation

You can load the library with POCV LVF into PrimeTime before linking the design. The unit of POCV LVF data is the time unit of the library. The LVF data is automatically read in during loading of the library.

Figure 18 shows an example of a POCV slew-load table in the library. The `index_1` is output load (fF) and `index_2` is input transition (ns). At output load 0.1296fF and input transition 0.0108ns, the corresponding early sigma is 0.0005608ns.

Figure 18 Example of POCV LVF table for Transition Variation

```
ocv_sigma_rise_transition ("delay_template_7x7") {
  sigma type : "early";
  index 1("0.0088000, 0.0264000, 0.0608000, 0.1296000, 0.2672000, 0.5424000, 1.0936000");
  index 2("0.0010000, 0.0024000, 0.0052000, 0.0108000, 0.0221000, 0.0445000, 0.0895000");
  values("0.0004474, 0.0011155, 0.0014607, 0.0027899, 0.0060622, 0.0327056, 0.0199022", \
  "0.0002205, 0.0003167, 0.0013133, 0.0050358, 0.0051558, 0.0122486, 0.0175923", \
  "0.0007443, 0.0005332, 0.0002091, 0.0016752, 0.0067215, 0.0256739, 0.0163855", \
  "0.0014063, 0.0034562, 0.0021763, 0.0005608, 0.0027979, 0.0135877, 0.0478228", \
  "0.0025631, 0.0048075, 0.0039450, 0.0001698, 0.0021382, 0.0068812, 0.0215443", \
  "0.0022192, 0.0110083, 0.0078389, 0.0034448, 0.0036182, 0.0046001, 0.0136651", \
  "0.0130970, 0.0099494, 0.0046566, 0.0036081, 0.0052415, 0.0588888, 0.0127493");
 }                              }

ocv_sigma_rise_transition ("delay_template_7x7") {
  sigma type : "late";
  index_1("0.0088000, 0.0264000, 0.0608000, 0.1296000, 0.2672000, 0.5424000, 1.0936000");
  index_2("0.0010000, 0.0024000, 0.0052000, 0.0108000, 0.0221000, 0.0445000, 0.0895000");
    values("0.000476, 0.000677, 0.001075, 0.001870, 0.003438, 0.006626, 0.012922", \
    "0.000651, 0.000901, 0.001303, 0.002081, 0.003678, 0.006818, 0.013144", \
    "0.000840, 0.001166, 0.001714, 0.002558, 0.004112, 0.007249, 0.013529", \
    "0.001115, 0.001520, 0.002193, 0.003317, 0.005087, 0.008153, 0.014445", \
    "0.001521, 0.002033, 0.002883, 0.004242, 0.006522, 0.010072, 0.016258", \
    "0.002155, 0.002793, 0.003853, 0.005563, 0.008424, 0.012955, 0.020171", \
    "0.003204, 0.003977, 0.005321, 0.007515, 0.010960, 0.016582, 0.025786");
}
```

For a detailed description of the POCV Library Variation Format (LVF) syntax, refer to the Appendix POCV Library Variation Format.

Enabling Transition Variation

Before enabling transition variation, you must enable POCV analysis:

```
pt_shell> set_app_var timing_pocvm_enable_analysis true
```

To enable transition variation in POCV analysis:

```
pt_shell> set_app_var timing_enable_slew_variation true
```

How Transition Variation Is Combined Into Cell Delay Variation and Output Transition

In order to estimate the impact of transition variation on cell delay variation, first PrimeTime calculates the variation in cell delay due to the transition variation at the input transition of the cell. This is called “delay sigma induced by input transition variation”.

```
            Dσ(Sσ) = (D(Snom+N*Sσ) – D(Snom))/N
  where
 Dσ(Sσ)            additional variation on cell delay due to input transition
variation
 D(Snom)           cell delay @ mean slew
 D(Snom+N*Sσ)      cell delay @ mean_slew + N * slew_sigma
 N                 corner sigma timing_pocvm_corner_sigma
```

The combined sigma of cell delay sigma and additional sigma induced by input transition variation is calculated using a proprietary model which assumes cell delay and input transition are correlated.

Similarly, the impact of input transition variation on output transition is estimated the same way. First PrimeTime calculates the variation in output transition due to input transition variation. This is called as “output transition sigma induced by input transition variation”.

```
            Tσ(Sσ) = (T(Snom+N*Sσ) – T(Snom))/N
  where
 Tσ(Sσ)            additional variation on output transition due to input
transition variation
 T(Snom)           output transition @ mean slew
 T(Snom+N*Sσ)      output transition @ mean_slew + N * slew_sigma
 N                 corner sigma timing_pocvm_corner_sigma
```

The combined sigma of output transition sigma and additional sigma induced by input transition variation is then calculated using a proprietary model which assumes cell delay and input transition are correlated.

Reporting with Transition Variation

When transition variation is enabled in POCV analysis, the output of reporting commands `report_ocvm -type pocvm` and `report_delay_calculation -derate` show the transition variation accordingly.

For the `report_ocvm -type pocvm` command, like delay variation because the slew-load table is defined on library timing arcs, you need to specify the library timing arc in the `object_list` to make it work. The `report_ocvm` command displays all the slew-load tables, delay as well as transition variation, associated with the specified library timing arc. Figure 19 shows an example of the output of the `report_ocvm` command when LVF data for transition variation is available.

Figure 19 Example the output of report_ocvm –type pocvm command with transition variation

```
pt_shell> report_ocvm -type pocvm [get_lib_timing_arcs -from */INVD1/I -to */INVD1/ZN]

min rise table
Sense / Type: negative unate
Delay:

             | Slew
  Load       | 0.0010000  0.0024000  0.0052000  0.0108000  0.0221000  0.0445000  0.0895000
 ------------+-------------------------------------------------------------------------------
  0.0088000  | 0.0004760  0.0006770  0.0010750  0.0018700  0.0034380  0.0066260  0.0129220
  0.0264000  | 0.0006510  0.0009010  0.0013030  0.0020810  0.0036780  0.0068180  0.0131440
  0.0608000  | 0.0008400  0.0011660  0.0017140  0.0025580  0.0041120  0.0072490  0.0135290
  0.1296000  | 0.0011150  0.0015200  0.0021930  0.0033170  0.0050870  0.0081530  0.0144450
  0.2672000  | 0.0015210  0.0020330  0.0028830  0.0042420  0.0065220  0.0100720  0.0162580
  0.5424000  | 0.0021550  0.0027930  0.0038530  0.0055630  0.0084240  0.0129550  0.0201710
  1.0936000  | 0.0032040  0.0039770  0.0053210  0.0075150  0.0109600  0.0165820  0.0257860

Transition:

             | Load
  Slew       | 0.0010000  0.0024000  0.0052000  0.0108000  0.0221000  0.0445000  0.0895000
 ------------+-------------------------------------------------------------------------------
  0.0088000  | 0.0004474  0.0011155  0.0014607  0.0027899  0.0060622  0.0327056  0.0199022
  0.0264000  | 0.0002205  0.0003167  0.0013133  0.0050358  0.0051558  0.0122486  0.0175923
  0.0608000  | 0.0007443  0.0005332  0.0002091  0.0016752  0.0067215  0.0256739  0.0163855
  0.1296000  | 0.0014063  0.0034562  0.0021763  0.0005608  0.0027979  0.0135877  0.0478228
  0.2672000  | 0.0025631  0.0048075  0.0039450  0.0001698  0.0021382  0.0068812  0.0215443
  0.5424000  | 0.0022192  0.0110083  0.0078389  0.0034448  0.0036182  0.0046001  0.0136651
  1.0936000  | 0.0130970  0.0099494  0.0046566  0.0036081  0.0052415  0.0588888  0.0127493
```

The output of `report_delay_calculation -derate` also shows the data accordingly when transition variation is enabled. Figure 20 shows the output of `report_delay_calculation -derate` when transition variation is applied.

Figure 20 The output of report_delay_calculation –derate command with transition variation

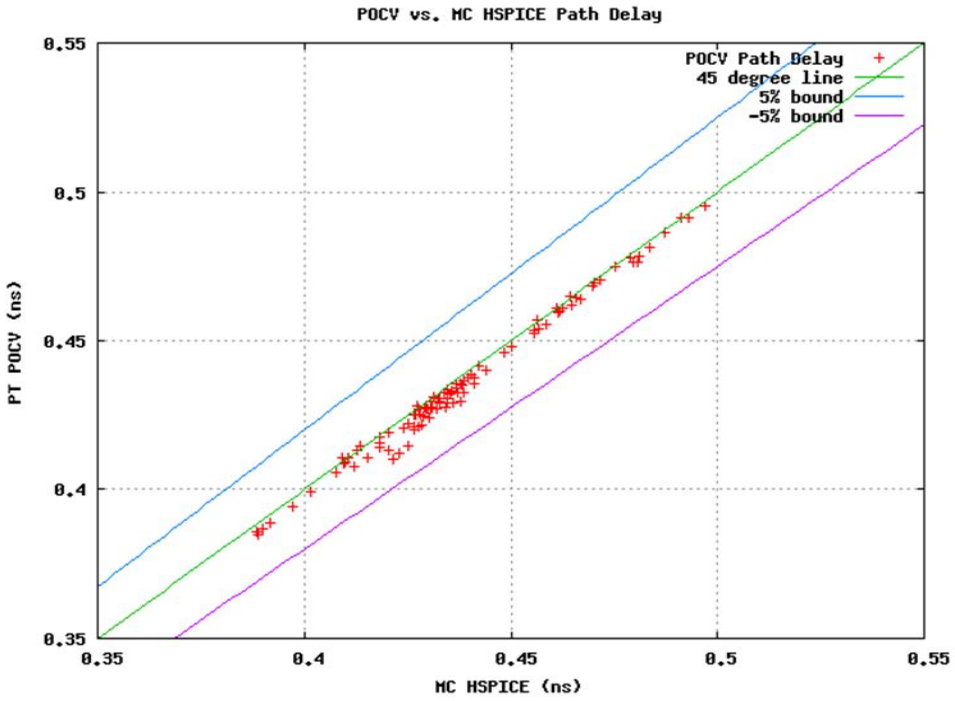
```
pt_shell> report_delay_calculation -from I/I -to I/ZN -derate
...
     Advanced driver-modeling used for rise and fall.

                              Rise        Fall
   ---------------------------------------------------------------
   Input transition time  = 0.014212    0.014164  (in library unit)
   Effective capacitance  = 0.000680    0.000679  (in pF)
   Effective capacitance  = 0.000680    0.000679  (in library unit)
   Output transition time = 0.018029    0.016408  (in library unit)
   Cell delay             = 0.032504    0.037570  (in library unit)

   POCVM delay sigma      = 0.001625    0.001878
   POCVM delay sigma induced by input transition variation        = 0.002957    0.003088
   POCVM combined delay sigma        = 0.004583    0.004967
   POCVM output transition sigma induced by input transition variation = 0.000189  0.000125
   POCVM combined output transition sigma        = 0.009204    0.008329
   POCVM input transition sigma        = 0.007106    0.007082
   POCVM coef scale factor = 1.030000    1.030000
   POCVM distance derate   = 1.045007    1.045007
   POCVM guardband         = 1.020000    1.020000
   Incremental derate      = 0.040000    0.040000
   Cell delay derated      = 0.035947    0.041548  (in library unit)
   Cell delay sigma        = 0.004815    0.005218  (in library unit)

  Cell  delay derated = "Cell delay" * ( "POCVM guardband" * "POCVM distance derate" +
"Incremental derate" )
   Cell  delay sigma   = "POCVM combined delay sigma" * ( "POCVM guardband" * "POCVM coef
scale factor" )
```

Accuracy with Transition Variation

Enabling transition variation helps reduce the optimism in POCV analysis. As a result, it improves POCV correlation with HSPICE Monte Carlo.

Figure 21 shows the path delay correlation between POCV and Monte-Carlo HSPICE simulation with transition variation enabled. The POCV path delay is bounded within +/- 5% compared to Monte-Carlo HSPICE simulation.

Figure 21 The correlation of POCV path delay with transition variation enabled vs. Monte-Carlo HSPICE simulation



POCV Constraint Variation

PrimeTime supports POCV Liberty Variation Format (LVF) for constraint variation. Similar to delay variation and transition variation, the constraint variation data is represented as a slew-load table per constraint timing arc in liberty variation format. When constraint variation is enabled, the constraint variation is taken into account in slack calculation.

PrimeTime now supports for LVF recovery and removal, non-sequential setup and hold, nochange setup and hold, and clock-gating check, as well as setup and hold

For 16nm technology node and smaller, it is recommended to use POCV LVF with constraint variation to improve the accuracy of POCV analysis.

LVF Format for Constraint Variation

The LVF data of constraint variation is automatically read in during loading of the POCV LVF library. Figure 22 shows an example of POCV LVF table with constraint variation in the library. The `index_1` is clock input transition (ns) and `index_2` is data input transition.

Figure 22 Example of POCV LVF table for Constraint Variation

```
timing () {
                                related_pin : "CK";
                                timing_type : "setup_rising";
                                rise constraint ("constraint template 3x3") {
                                        index 1("0.0049000, 0.0531000, 0.4384000");
                                        index 2("0.0049000, 0.0531000, 0.4384000");
                                        values("0.0175774, 0.0359865, 0.0956248", \
                                                "0.0030500, 0.0195200, 0.0772492", \
                                                "-0.0311726, -0.0186907, 0.0273438");
                                }
                               ocv_sigma_rise_constraint ("constraint_template_3x3") {
                                        index 1("0.0049000, 0.0531000, 0.4384000");
                                        index 2("0.0049000, 0.0531000, 0.4384000");
                                        values("0.007576, 0.012984, 0.025638", \
                                                "0.001040, 0.0195200, 0.02122", \
                                                "0.011726, 0.0086907, 0.007934");
                                }
```

For a detailed description of the POCV Library Variation Format (LVF) syntax, refer to the Appendix POCV Library Variation Format.

SiliconSmart also supports constraint variation characterization.

Enabling Constraint Variation

Before enabling constraint variation, you must enable POCV analysis:

```
pt_shell> set_app_var timing_pocvm_enable_analysis true
```

To enable constraint variation in POCV analysis:

```
pt_shell> set_app_var timing_enable_constraint_variation true
```

Reporting with Constraint Variation

When constraint variation is enabled in POCV analysis and LVF data for constraint variation is available in the library, the output of reporting commands `report_ocvm -type pocvm`, `report_timing -derate`, `report_timing -variation` and `report_delay_calculation` include the constraint variation data.

Starting from PrimeTime M-2016.12, constraint variation supports `pocvm_guardband`. Before version M-2016.12, you get an error message. `CMD-001`:

```
pt_shell> set_timing_derate -cell_check -pocvm_guardband -late 1.05
Error: Cannot specify '-pocvm_guardband' with '-cell_check'. (CMD-001)
```

Starting from PrimeTime L-2016.06-SP3 and later, constraint variation supports the `pocvm_coefficient_scale_factor` variable.

You can use `report_ocvm -type pocvm` command to print out LVF tables of a constraint arc. Figure 23 shows an example of the output of the `report_ocvm` command when LVF data for constraint variation is available. Constraint sigma is a function of clock input transition and data input transition.

Figure 23 Example the output of report_ocvm –type pocvm command with constraint variation

```
pt_shell> report_ocvm -type pocvm [get_lib_timing_arcs -from */DFF/CK -to */DFF/D]

setup fall table
Sense / Type: setup clk rise
Constraint:
             | Slew
  Slew       | 0.0049000  0.0531000  0.4384000
 ------------+---------------------------------
  0.0049000  | 0.0156240  0.0320963  0.1073436
  0.0531000  | 0.0067125  0.0097309  0.0850481
  0.4384000  | 0.0741415  0.0596409  0.0078125

hold rise table
Sense / Type: hold_clk_rise
Constraint:
             | Slew
  Slew       | 0.0049000  0.0531000  0.4384000
 ------------+---------------------------------
  0.0049000  | 0.0000000  0.0125652  0.0370311
  0.0531000  | 0.0125723  0.0019511  0.0225594
  0.4384000  | 0.0467976  0.0361933  0.0078432
```

The output of `report_timing -derate` reports constraint variation when it is enabled. Figure 24 shows the output of `report_timing -derate` when constraint variation is applied.

Figure 24 The output of report_timing –derate command with constraint variation

```
pt_shell> report_timing -derate -nosplit -delay min

  Point                          Derate    Mean    Sensit  Incr       Path
  ---------------------------------------------------------------------------
  clock CLK (rise edge)                                     0.000      0.000
  clock network delay (propagated)                          0.104      0.104
  Uff1/CP (DFD4BWP)                        0.000   0.000    0.000      0.104 r
  Uff1/Q (DFD4BWP)               0.822     0.077   0.006    0.072 &    0.176 f
  buf1/I (BUFFD1BWP)             0.960     0.000   0.000    0.000 &    0.176 f
  buf1/Z (BUFFD1BWP)             0.822     0.030   0.002    0.029 &    0.205 f
  buf2/I (BUFFD1BWP)             0.960     0.000   0.000    0.000 &    0.205 f
  buf2/Z (BUFFD1BWP)             0.822     0.032   0.003    0.030 &    0.235 f
  Uff2/D (DFD4BWP)               0.960     0.000   0.000    0.000 &    0.235 f
  data arrival time                                                     0.235
  clock CLK (rise edge)                                     0.000      0.000
  clock network delay (propagated)                          0.203      0.203
  clock reconvergence pessimism           -0.039  -0.009  -0.052      0.150
  clock uncertainty                                         0.100      0.250
  Uff2/CP (DFD4BWP)                                                     0.250 r
  library hold time              1.000     0.016   0.016    0.060      0.310
  data required time                       0.267   0.016               0.310
  ---------------------------------------------------------------------------
  data required time                       0.267   0.016               0.310
  data arrival time                       -0.247   0.008              -0.235
  ---------------------------------------------------------------------------
  slack (VIOLATED)                        -0.020   0.018              -0.075
```

Similarly, the output of `report_timing -variation` reports constraint variation when enabled. Figure 25 shows the output of `report_timing -variation` with constraint variation.

Figure 25 The output of report_timing –variation command with constraint variation

```
pt_shell> report_timing -variation -nosplit -delay min

  Point                                    Mean    Sensit  Incr       Path
  ---------------------------------------------------------------------------
  clock CLK (rise edge)                                     0.000      0.000
  clock network delay (propagated)                          0.104      0.104
  Uff1/CP (DFD4BWP)                        0.000   0.000    0.000      0.104 r
  Uff1/Q (DFD4BWP)                         0.077   0.006    0.072 &    0.176 f
  buf1/Z (BUFFD1BWP)                       0.030   0.002    0.029 &    0.205 f
  buf2/Z (BUFFD1BWP)                       0.032   0.003    0.031 &    0.235 f
  Uff2/D (DFD4BWP)                         0.000   0.000    0.000 &    0.235 f
  data arrival time                                                     0.235
  clock CLK (rise edge)                                     0.000      0.000
  clock network delay (propagated)                          0.203      0.203
  clock reconvergence pessimism           -0.039  -0.009  -0.052      0.150
  clock uncertainty                                         0.100      0.250
  Uff2/CP (DFD4BWP)                                                     0.250 r
  library hold time                        0.016   0.016    0.060      0.310
  data required time                       0.267   0.016               0.310
  ---------------------------------------------------------------------------
  data required time                       0.267   0.016               0.310
  data arrival time                       -0.247   0.008              -0.235
  ---------------------------------------------------------------------------
  slack (VIOLATED)                        -0.020   0.018              -0.075
```

The output of `report_delay_calculation` is also modified accordingly when constraint variation is enabled. It shows how constraint sigma is extracted from the LVF table. Figure 26 shows the output of `report_delay_calculation` when constraint variation is applied.

Figure 26 The output of report_delay_calculation command with constraint variation

```
pt_shell> report_delay_calculation -from FF1/CK -to FF1/D
...

   Fall Delay Sigma
   cell delay sigma = 0.0163316
     Table is indexed by
      (X) related_pin_transition = 0.0193678
      (Y) constrained pin transition = 0.0164327
     Relevant portion of lookup table:
                   (X)  0.0049      (X)  0.0531
  (Y)  0.0049      (Z)  0.0137      (Z)  0.0321
  (Y)  0.0531      (Z)  0.0011      (Z)  0.0215
     Z = A + B*X + C*Y + D*X*Y
     A =  0.0131           B =  0.3776
     C = -0.2651           D =  0.8605
  Z = 0.0163316
  scaling result for operating conditions
  multiplying by 1 gives 0.0163316

  Cell Delay Sigma
  fall:  0.0163316
```

POCV and Latches

Traditional latch borrowing calculation is different depending on whether the latch is borrowing, or violating. If the latch is borrowing, the arrival launched forward from the latch D pin depends on the incoming arrival to the latch. If the latch is violating, the launch forward from the latch D pin is based on the close edge of the latch.

In POCV analysis, all timing quantities are statistical and contain variation information. As a result, PrimeTime must consider that the variation for the incoming and close edge at the latch may be different. Switching between these two modes (i.e. launching forward the incoming arrival vs launching forward the close edge) would cause a slack discontinuity for endpoints downstream of the latch.

To remove this discontinuity, starting from version M-2016.12, when the path to the latch is violating, the launch forward from the latch is modeled as a function of both the closing edge and the arrival. When it happens, the report shows a line “smoothed adjustment to close edge” in the timing borrow section of a violating advanced latch or “close edge smoothing adjustment” in the timing borrow section of a violating regular latch.

Figure 27 shows the output of `report_timing -variation` of an advanced latch where the latch is causing a violation.

Figure 27 The output of report_timing -variation using advanced latch analysis

```
pt_shell> report_timing -variation -nosplit

  Startpoint: f_launch (rising edge-triggered flip-flop clocked by clk)
  Endpoint: f_capture (rising edge-triggered flip-flop clocked by clk)
  Last common pin: b_clk1/Z
  Path Group: clk
  Path Type: max
  Sigma: 3.0

                                          ------- Incr -------------  ------- Path ---------
  Point                                    Mean   Sensit Corner Value  Mean   Sensit    Value
  --------------------------------------------------------------------------------------------
  clock clk (rise edge)                    0.00                 0.00   0.00   0.00     0.00
  clock network delay (propagated)         2.62   0.22   3.27   3.27   2.62   0.22     3.27
  f_launch/CP (FD1)                        0.00   0.00   0.00   0.00   2.62   0.22     3.27 r
  f_launch/Q (FD1)                         1.67   0.17   2.18   1.84   4.30   0.27     5.12 r
  b1/Z (NR3)                               0.57   0.06   0.75   0.59   4.87   0.28     5.71 r
  b2/Z (NR3)                              41.00   4.10  53.30  52.49  45.87   4.11 H  58.20 r
  l1/D (LD1)                               0.00   0.00   0.00   0.00  45.87   4.11    58.20 r
  --------------------------------------------------------------------------------------------
  transparency window #1 (missed)

  clock clk (fall edge)                   10.00                10.00  10.00   0.00    10.00
  clock latency                            2.26   0.18   2.79   1.72  12.26   0.18    11.72
  clock reconvergence pessimism            0.40  -0.27  -0.41   1.54  12.66  -0.20    13.26
  transparency open edge                                              12.66  -0.20    13.26

  clock clk (rise edge)                   20.00                20.00  20.00   0.00    20.00
  clock latency                            2.80   0.20   3.41   2.20  22.80   0.20    22.20
  clock reconvergence pessimism            0.40  -0.27  -0.41   1.54  23.20  -0.18    23.74
  library setup time                      -0.40   0.00  -0.40  -0.40  22.80  -0.18    23.34
  transparency close edge                                             22.80  -0.18    23.34

  l1/D (LD1)                                                          45.87   4.11    58.20
  smoothed adjustment to close edge      -22.53  -4.11 -34.86 -34.86  23.34   0.00    23.34
  --------------------------------------------------------------------------------------------
  l1/D (LD1)                               0.00   0.00   0.00   0.00  23.34   0.00    23.34 r
  l1/Q (LD1)                              -0.00  -0.00  -0.00   0.00  23.34  -0.00 H  23.34 r
  b3/Z (NR3)                               0.57   0.06   0.75   0.74  23.91   0.06    24.09 r
  b4/Z (NR3)                               4.00   0.40   5.20   5.04  27.91   0.40 H  29.13 r
  f_capture/D (FD1)                        0.00   0.00   0.00   0.00  27.91   0.40    29.13 r
  data arrival time                                                   27.91   0.40    29.13

  clock clk (rise edge)                   20.00                20.00  20.00   0.00    20.00
  clock network delay (propagated)         2.22   0.18   1.69   1.69  22.22   0.18    21.69
  clock reconvergence pessimism            0.40  -0.27   1.21   1.54  22.62  -0.20    23.23
  f_capture/CP (FD1)                       0.00   0.00   0.00   0.00  22.62  -0.20    23.23
  library setup time                      -0.80                -0.80  21.82  -0.20    22.43
  data required time                                                  21.82  -0.20    22.43
  --------------------------------------------------------------------------------------------
  data required time                                                  21.82  -0.20    22.43
  data arrival time                                                  -27.91   0.40   -29.13
  --------------------------------------------------------------------------------------------
  statistical adjustment                                 -0.45                    -7.14
  slack (VIOLATED)                                                    -6.09   0.35    -7.14
```

Figure 28 shows the output of `report_timing -variation` of a regular latch where the latch is causing a violation.

Figure 28 The output of report_timing -variation for a regular latch

```
pt_shell> report_timing -variation -nosplit

  Startpoint: f_launch (rising edge-triggered flip-flop clocked by clk)
  Endpoint: l1 (positive level-sensitive latch clocked by clk')
  Last common pin: b_clk1/Z
  Path Group: clk
  Path Type: max
  Sigma: 3.0

                                          -------- Incr ---------------   -------- Path ----------
  Point                                     Mean    Sensit  Corner  Value   Mean    Sensit     Value
  ---------------------------------------------------------------------------------------------------
  clock clk (rise edge)                    0.000                   0.000   0.000   0.000     0.000
  clock network delay (propagated)         2.624   0.216   3.274   3.274   2.624   0.216     3.274
  f_launch/CP (FD1)                        0.000   0.000   0.000   0.000   2.624   0.216     3.274 r
  f_launch/Q (FD1) <-                      1.579   0.158   2.053   1.734   4.204   0.268     5.007 f
  b1/Z (NR3)                               1.116   0.112   1.451   1.183   5.320   0.290     6.190 f
  b2/Z (NR3)                              41.000   4.100  53.300  52.460  46.320   4.110 H  58.650 f
  l1/D (LD1)                               0.000   0.000   0.000   0.000  46.320   4.110    58.650 f
  data arrival time                                                       46.320   4.110    58.650

  clock clk' (rise edge)                  10.000                  10.000  10.000   0.000    10.000
  clock network delay (propagated)         2.256   0.179   1.719   1.719  12.256   0.179    11.719
  clock reconvergence pessimism            0.400  -0.270   1.211   1.544  12.656  -0.203    13.263
  l1/G (LD1)                               0.000   0.000   0.000   0.000  12.656  -0.203    13.263
  time borrowed from endpoint             10.148   0.095   9.862  10.076  22.804  -0.179    23.340
  data required time                                                      22.804  -0.179    23.340
  ---------------------------------------------------------------------------------------------------
  data required time                                                      22.804  -0.179    23.340
  data arrival time                                                      -46.320   4.110   -58.650
  ---------------------------------------------------------------------------------------------------
  statistical adjustment                                        -0.524                    -35.835
  slack (VIOLATED)                                                       -23.516   4.106   -35.835

  Time Borrowing Information
  ---------------------------------------------------------------------------------------------------
  clk' nominal pulse width                                                                   10.000
  clock latency difference                                                 0.548   0.095     0.262
  library setup time                                                      -0.400   0.000    -0.400
  ---------------------------------------------------------------------------------------------------
  max time borrow                                                         10.148   0.095     9.862
  ---------------------------------------------------------------------------------------------------
  actual time borrow                                                      10.148  -0.095     9.862
  close edge smoothing adjustment                                          0.536  -0.179     0.000
  statistical borrow adjustment                                                              0.215
  ---------------------------------------------------------------------------------------------------
  time given to startpoint                                                10.684  -0.203    10.076
```

When not using advanced latch analysis, or when using advanced latch analysis but the latch has been identified as a loop breaker or path breaker latch (see the command `report_latch_loop_groups`), in addition to the "`close edge smoothing adjustment`" line, there is a "`statistical borrow adjustment`" line. This line is similar to the "`statistical adjustment`" line above the "`slack`" line in that it compensates for the way statistical quantities are added and causes the `Value` column to sum properly.

For more details about regular latches and advanced latches, refer to the *PrimeTime User Guide*.

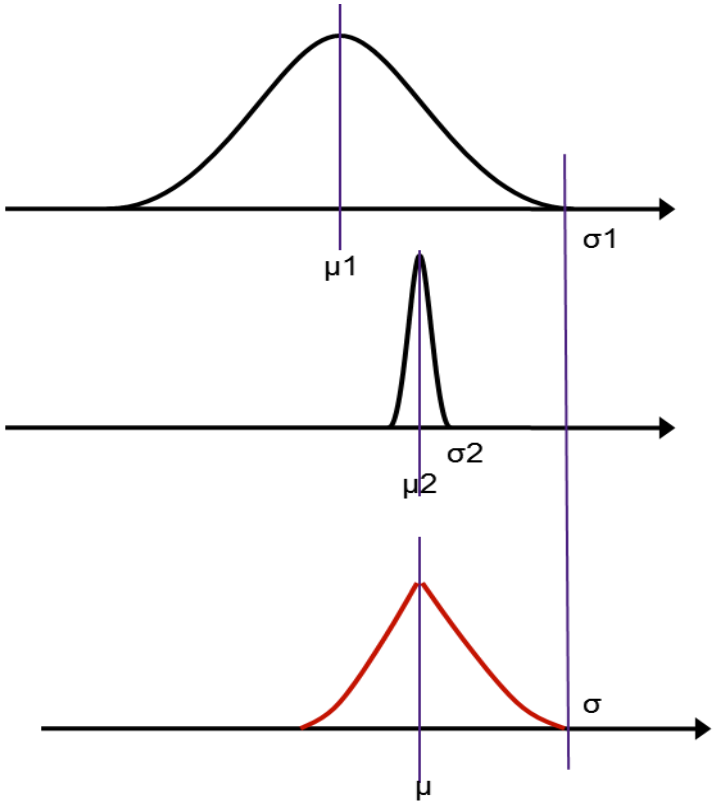
Statistical Graph Pessimism

With traditional (non-POCV) static timing analysis (STA), the maximum of two numbers A and B is equal to either A or B (or both). In POCV STA, the quantities under consideration are not single numbers but statistical distributions specified by mean and

standard deviation. In the case of POCV, the maximum of two distributions A and B is a constructed distribution M where M has a mean value of the maximum of the mean values of A and B, and the k-sigma corner value of the maximum of the k-sigma corners of A and B, as illustrated in Figure 29. This ensures that k-sigma pessimism is preserved.

As a consequence, the POCV graph arrival computed by the `update_timing` command at a given point is not necessarily identical to the worst-case statistical sum of delays to that point. The difference between the graph arrival and the statistical sum-of-delays is called statistical graph pessimism.

Figure 29 Statistical MAX operation



$\mu = MAX(\mu 1, \mu 2)$

σ=(MAX(μ1+k*σ1, μ2+k*σ2) - μ)/k

where k = `timing_pocvm_corner_sigma` (default is 3)

Due to statistical MAX operation in POCV analysis

- Graph pin slack attribute not found even with very high `-nworst` in `report_timing`
- GBA (graph-based analysis) path slack not the worst at `-nworst` 1 versus `-nworst` > 1

Starting from PrimeTime M-2016.12, `report_timing -variation` displays `statistical graph pessimism` inline at points in a GBA timing path where it is incurred (where the statistical max operation leads to the introduction of pessimism). The `statistical graph pessimism` information includes both the mean and the standard deviation.

The new rows indicating the graph pessimism at these points are highlighted in yellow in Figure 30.

Figure 30 Statistical Graph Pessimism

```
pt_shell> report_timing -variation -delay_type max_rise
  Startpoint: in1 (input port clocked by clk)
  Endpoint: out1 (output port clocked by clk)
  Path Group: clk
  Path Type: max
  Sigma: 3.0

                                          -------- Incr ----------------  -------- Path ----------
 Point                                     Mean     Sensit  Corner   Value   Mean     Sensit     Value
 --------------------------------------------------------------------------------------------------
 clock clk (rise edge)                     0.00                      0.00    0.00                0.00
 clock network delay (ideal)               0.00     0.00     0.00    0.00    0.00     0.00       0.00
 input external delay                      0.00                      0.00    0.00                0.00 r
 in1 (in)                                  0.00     0.00     0.00    0.00    0.00     0.00       0.00 r
 b1/Z (NR3)                                1.00     3.00    10.00   10.00    1.00     3.00 H    10.00 r
 statistical graph pessimism               3.00    -2.24             0.00    4.00     2.00      10.00
 b3/Z (NR3)                                1.00     2.00     7.00    3.49    5.00     2.83 H    13.49 r
 statistical graph pessimism               5.00    -2.58             0.00   10.00     1.16      13.49
 out1 (out)                                0.00     0.00     0.00    0.00   10.00     1.16      13.49 r
 data arrival time                                                                              13.49

 clock clk (rise edge)                     2.00                      2.00    2.00                2.00
 clock network delay (ideal)               0.00     0.00     0.00    0.00    0.00     0.00       2.00
 clock reconvergence pessimism             0.00     0.00     0.00    0.00    0.00     0.00       2.00
 output external delay                                               0.00                        2.00
 data required time                                                          2.00     0.00       2.00
 --------------------------------------------------------------------------------------------------
 data required time                                                          2.00     0.00       2.00
 data arrival time                                                         -10.00     1.16     -13.49
 --------------------------------------------------------------------------------------------------
 statistical adjustment                                              0.00                      -11.49
 slack (VIOLATED)                                                              -8.00     1.16     -11.49
```

With `statistical graph pessimism` added into `report_timing` GBA

- Slack from `report_timing -nworst 1` (vs. `-nworst > 1`) will be the worst slack
- GBA `report_timing` slack will match endpoint pin slack attribute

Please note that with statistical graph pessimism, in some cases you may see `report_timing -from / -through -to` *`endpoint`* does not match `report_timing -to` *`endpoint`*. A timing report of a subgraph to an endpoint (i.e. `report_timing -from / -through -to` *`endpoint`*) can potentially show less graph merging pessimism than a timing report to an endpoint (i.e. `report_timing -to` *`endpoint`*). As a result, in POCV analysis we can expect that `report_timing -to` *`endpoint`* will bound (be more pessimistic than), and not necessarily match, `report_timing -from / -through -to` *`endpoint`*.

POCV and SI Analysis

POCV analysis is only applied on non-SI cell delay. However, POCV can indirectly change crosstalk delta delay due to the difference in timing window alignment.

- POCV affects arrival window of victim and aggressors
- As a result, POCV can indirectly affect crosstalk delta delay

POCV and Voltage Scaling

For a POCV side file and voltage scaling, you can specify the POCV side file at a specific voltage using keyword “voltage”. The POCV coefficient at a voltage value between two POCV side files is linearly interpolated.

For example, you have two POCV side files at 0.7V and 0.5V, as follows:

POCV tables at 0.7V	POCV tables at 0.5V
version : 4.0 ocvm_type : pocvm object_type: lib_cell voltage : 0.7 rf_type : rise fall delay_type : cell derate_type: early object_spec: lib28nm/invx* coefficient: 0.04	version : 4.0 ocvm_type : pocvm object_type: lib_cell voltage : 0.5 rf_type : rise fall delay_type : cell derate_type: early object_spec: lib28nm/invx* coefficient: 0.08

At 0.6V, POCV coefficient is linearly interpolated as 0.06.

For LVF and voltage scaling, you need LVF libraries at different voltages. PrimeTime automatically performs voltage scaling on LVF data using scaling libraries defined by `define_scaling_lib_group` command. LVF voltage scaling uses linear interpolation.

ETM with POCV Analysis

When POCV analysis is enabled, the `extract_model` command automatically generates LVF delay tables in the extracted timing model (ETM). If constraint variation is enabled by setting `timing_enable_constraint_variation` to `true`, it also generates LVF constraint tables in the ETM.

Starting from 2015.06-SP3-2, `extract_model` can generate different types of POCV data in the ETM using the variable `extract_model_create_variation_tables`.

`extract_model_create_variation_tables` has the following settings:

- `auto` (the default) – `extract_model` command generates LVF delay tables in the ETM. It also generates LVF constraint tables if `timing_enable_constraint_variation` variable is set to `true`.
- `delay_only` – `extract_model` command generates only LVF delay tables in the ETM, not LVF constraint tables. Instead, constraint variation is embedded in nominal constraint tables.
- `none` – `extract_model` command does not generate LVF delay tables nor LVF constraint tables in the ETM. Instead, delay variation and constraint variation rare embedded in nominal delay tables and nominal constraint tables accordingly.

LVF delay tables specify delay variation as a function of output load and input slew, as specified by `ocv_sigma_cell_rise` and `ocv_sigma_cell_fall` timing types in Liberty Variation Format (LVF). LVF constraint tables specify constraint variation as a function of input slew of constraint pin and input slew of related pin, as specified by `ocv_sigma_rise_constraint` and `ocv_sigma_fall_constraint` timing types in Liberty Variation Format (LVF).

For details on how to generate an ETM, refer to the *PrimeTime User Guide*.

Running `check_library` from Library Compiler

The `check_library` from Library Compiler supports LVF checking. It is recommended to run `check_library` to check the quality of LVF data in your libraries before running POCV analysis (refer to the *Library Compiler User Guide* for details on how to run `check_library`).

You can use `check_library` to check which cells are missing cell delay variation, transition variation, or constraint variation. You also can check which cells have a large ratio between sigma and nominal as well as between sigma early and sigma late.

Below is an example showing how to run `check_library` to check an LVF library. In this example, the check for missing constraint sigma is skipped due to the library having no LVF data for constraint variation. It also checks any cells having a ratio of sigma over nominal larger than 0.5 and ratio of sigma late over sigma early greater than 3.

```
read_db lib1.db

set_check_library_options -analyze {lvf} -group \
 {!ocv_sigma_*_constraint} -criteria {sigma_to_nominal_ratio>0.5 \
 sigma_to_sigma_ratio>3} -report_format {nosplit}

report_check_library_options

check_library
```

Conclusion

POCV utilizes statistical analysis to address both random and systematic variation using POCV coefficient/LVF and distance-based derating tables. It helps to reduce the pessimism in graph-based analysis by not depending on path depth calculations as in AOCV. As a result, it reduces the overall turn-around time and chip area increase during ECO fixing.

PrimeTime supports POCV LVF syntax. The users have the flexibility to feed in POCV data while reading the library with POCV LVF or as a separate side file with POCV single-coefficient data. For better accuracy in timing result, POCV LVF is recommended.

For users who are familiar with the AOCV flow, moving to POCV flow is quite simple and straightforward. To generate POCV data for the library cells, Silicon Smart, a characterization tool from Synopsys, can help you to complete that task quickly.

Appendix

This appendix contains the following sections:

- Example POCV Timing Analysis Script
- POCV Side File Format for Single Coefficient
- POCV Library Variation Format (LVF)

Example POCV Timing Analysis Script

Following is an example of the run script for POCV analysis using the side files with POCV single-coefficient data.

```
###############################################################################
#  Run POCV analysis using the side file with POCV single coefficient
###############################################################################

##  Analysis setup

lappend search_path ./
set link_path {* test.db}

read_verilog ./test.v
link
set_operating_conditions -analysis_type on_chip_variation

set read_parasitics_load_locations true
read_parasitics test.spef

create_clock -per 10 -wave {0 5} [get_ports CLK]
set_propagated_clock [all_clocks]
set_false_path -from [get_ports D]
set_false_path -to [get_ports Q]

set_app_var timing_crpr_threshold_ps 1
set timing_remove_clock_reconvergence_pessimism true
set timing_save_pin_arrival_and_slack true

## Set clock uncertainty
set_clock_uncertainty -setup 0.5 [get_clock *]
set_clock_uncertainty -hold 0.8 [get_clock *]

## set_timing_derate -increment
set_timing_derate -increment -late 0.04
set_timing_derate -increment -early -0.04

### Setup POCV
set_app_var timing_pocvm_enable_analysis true
set_app_var variation_report_timing_increment_format delay_variation

echo "Read POCV table"
read_ocvm pocv.table

echo "Read distanced-based derating table"
```

```
read_ocvm distance.table

### Report POCV
report_ocvm -type pocvm -list_not_annotated -coefficient -cell_delay
report_ocvm -type pocvm buf1

#### New POCV options in set_timing_derate
set_timing_derate -cell_delay -late  -clock -pocvm_guardband 1.02
set_timing_derate -cell_delay -early -clock -pocvm_guardband 0.98
set_timing_derate -cell_delay -late  -data -pocvm_guardband 1.02
set_timing_derate -cell_delay -early -data -pocvm_guardband 0.98
set_timing_derate -cell_delay -late -pocvm_coefficient_scale_factor 1.03
set_timing_derate -cell_delay -early -pocvm_coefficient_scale_factor 0.97

## Update_timing
update_timing

### Reporting
set path [get_timing_path -fall_to Uff2/D -delay min]
report_timing $path
report_timing $path -pba path
report_timing $path -derate

### Query new POCV attributes
get_attribute [get_attribute [get_pin Uff2/D] min_fall_variation_slack] mean
get_attribute [get_attribute [get_pin Uff2/D] min_fall_variation_slack] std_dev
get_attribute [get_attribute $path variation_slack] mean
get_attribute [get_attribute $path variation_slack] std_dev

### report_delay_calculation -derate
report_delay_calculation -from buf1/A -to buf1/Z -derate -min
report_delay_calculation -from buf1/A -to buf1/Z -derate
```

The following is an example of the run script using POCV LVF.

```
#############################################################################
#  Run POCV analysis using POCV LVF
#############################################################################

##  Analysis setup

lappend search_path ./
set link_path {* pocv_lvf.db}

read_verilog ./test.v
link
set_operating_conditions -analysis_type on_chip_variation

set read_parasitics_load_locations true
read_parasitics test.spef

create_clock -per 10 -wave {0 5} [get_ports CLK]
set_propagated_clock [all_clocks]
set_false_path -from [get_ports D]
set_false_path -to [get_ports Q]

set_app_var timing_crpr_threshold_ps 1
set timing_remove_clock_reconvergence_pessimism true
```

```
set timing_save_pin_arrival_and_slack true

## Set clock uncertainty
set_clock_uncertainty -setup 0.5 [get_clock *]
set_clock_uncertainty -hold 0.8 [get_clock *]

## set_timing_derate -increment
set_timing_derate -increment -late 0.04
set_timing_derate -increment -early -0.04

### Setup POCV
set_app_var timing_pocvm_enable_analysis true
set_app_var timing_enable_slew_variation true
set_app_var timing_enable_constraint_variation true
set_app_var variation_report_timing_increment_format delay_variation

echo "Read distanced-based derating table"
read_ocvm distance.table

### Report POCV
report_ocvm -type pocvm
report_ocvm -type pocvm [get_lib_timing_arc -from */buf1/A -to */buf1/Z]

#### New POCV options in set_timing_derate
set_timing_derate -cell_delay -late  -clock -pocvm_guardband 1.02
set_timing_derate -cell_delay -early -clock -pocvm_guardband 0.98
set_timing_derate -cell_delay -late  -data -pocvm_guardband 1.02
set_timing_derate -cell_delay -early -data -pocvm_guardband 0.98
set_timing_derate -cell_delay -late -pocvm_coefficient_scale_factor 1.03
set_timing_derate -cell_delay -early -pocvm_coefficient_scale_factor 0.97

## Update_timing
update_timing

### Reporting
set path [get_timing_path -fall_to Uff2/D -delay min]
report_timing $path
report_timing $path -pba path
report_timing $path -derate

### Query new POCV attributes
get_attribute [get_attribute [get_pin Uff2/D] min_fall_variation_slack] mean
get_attribute [get_attribute [get_pin Uff2/D] min_fall_variation_slack] std_dev
get_attribute [get_attribute $path variation_slack] mean
get_attribute [get_attribute $path variation_slack] std_dev

### report_delay_calculation -derate
report_delay_calculation -from buf1/A -to buf1/Z -derate -min
report_delay_calculation -from buf1/A -to buf1/Z -derate
```

POCV Side File Format for Single Coefficient

Without a library with POCV LVF data, you can use a side file to apply POCV single-coefficient data. The following table types are supported:

- Single POCV coefficient table
- One-dimensional distance tables

Syntax Definition

The syntax definitions for the POCV file are described in this section. You must specify each of these fields on a separate line.

- `version:` *`table_format_version`*

  The version number is used for backward compatibility purposes if the POCV file format changes in the future. You must specify the version of the POCV file on the first line of the POCV file. Valid version numbers are `4.0` or later.

- `group_name:` *`name`*

  This is an optional field which specifies the name of the POCV table group to which the table belongs.

- `object_type: design | cell| lib_cell`

  In the `object_type` definition, you can specify `design`, `cell`, or `lib_cell`. PrimeTime can annotate

  - Cell and net tables on the design using the `design` object type
  - Cell and net tables on hierarchical cells using the `cell` object type
  - Cell tables on library cells using the `lib_cell` object type

  Note that the derating tables cannot be annotated on leaf cells or nets. In other words, there is no leaf cell or net `object_type`.

- `rf_type: rise | fall| rise fall`

  In the `rf_type` definition, you must specify `rise`, `fall`, or `rise fall`. You cannot keep this field description empty.

- `delay_type: cell | net | cell net`

  In the `delay_type` definition, you must specify either `cell`, `net`, or `cell net`. You cannot keep this field description empty. PrimeTime can annotate:

  - `cell` and `net` tables on the design
  - `cell` and `net` tables on hierarchical cells
  - `cell` tables on library cells

- `derate_type: early | late`

  In the `derate_type` definition, you can specify either `early` or `late`. You cannot specify both.

- `object_spec: [patterns] [-filter expression]`

  In the `object_spec` definition, the optional *`patterns`* argument specifies the object name and an expression to evaluate based on the attributes of the object.

  You can use regular expression matching for the *`patterns`* argument. It is the same as the `regexp` Tcl command that can be used for design-object-gathering commands, such as the `get_cells`, `get_lib_cells`, and `get_designs` commands.

  You can use any of the options of the related object-gathering command in the *`patterns`* argument. For example, if the `object_type` is `lib_cell`, any of the arguments of the related `get_lib_cells` command could be used in the *`patterns`* argument.

- `voltage: floating_number`

  The optional `voltage` field supports multi-voltage POCV derating. The field can be only applied to `lib_cell` or `design` object types.

- `path_type: data | clock | data clock`

  This is an optional field that specifies whether the whole table applies to clock paths only, data paths only, or both types of paths.

- `distance: list_of_floating_numbers`

  Use this field to specify the path distance as a set of N floating-point values, where N can be zero. SPEF does not specify units for coordinate locations. PrimeTime assumes that the locations in the SPEF file are in micrometers (µm) and, to improve accuracy, converts them to nanometers (nm). Therefore, use nanometers for the distance field coordinates in the AOCV tables.

- `table: array_of_N_floating_numbers`

  The derating factors are specified in a table. This table is a set of N floating-point values.

- Comment line

  To indicate comments in the AOCV file, use double slashes (`//`) at the start of the line.

POCV Side File Example

The following example specifies a late POCV coefficient of 5% to the library cell BUFX1:

```
version: 4.0
ocvm_type: pocvm
object_type: lib_cell
rf_type: rise fall
delay_type: cell
derate_type: late
object_spec: */BUFX1
coefficient: 0.05
```

The following example specifies an early POCV derating distance-based table for the whole design, which applies to all cells:

```
version: 4.0
ocvm_type: pocvm
object_type: design
rf_type: rise fall
delay_type: cell
derate_type: early
object_spec:
distance: 100 200 300 400
table: 0.87 0.93 0.95 0.96
```

POCV Library Variation Format (LVF)

The following is an example which shows the complete syntax of POCV Library Variation Format (LVF).

```
library (<name>) {

lu_table_template(2D_lu_template) {
variable_1: input_net_transition;
variable_2: total_output_net_capacitance;
index_1 ("1,2,3");
index_2 ("1,2,3");
  }

lu_table_template(2D_lu_constraint_template) {
variable_1: related_pin_transition;
variable_2: constrained_pin_transition;
index_1 ("1,2,3");
index_2 ("1,2,3");
  }

… …
….
pin/bus/bundle(<name>) {
  direction: inout/output;
      timing() {
  … …
  related_pin : "CLKIN" ;
  timing_type : rising_edge ;
   ocv_sigma_cell_rise( 2D_lu_template ){
   index_1 ( "0.320000, 0.480000, 0.640000");
   index_2 ( "0.019098, 0.259098, 0.509098");
   values (  "0.100863, 0.114242, 0.126399",\
             "0.093746, 0.097281, 0.099437",\
             "0.083662, 0.087340, 0.089494");
   }
   ocv_sigma_rise_transition( 2D_lu_template ){
   index_1 ( "0.320000, 0.480000, 0.640000");
   index_2 ( "0.019098, 0.259098, 0.509098");
   values (  "0.100863, 0.114242, 0.126399",\
             "0.093746, 0.097281, 0.099437",\
             "0.083662, 0.087340, 0.089494");
   }
   ocv_sigma_cell_fall( 2D_lu_template ){
   index_1 ( "0.320000, 0.480000, 0.640000");
   index_2 ( "0.019098, 0.259098, 0.509098");
   values (  "0.100863, 0.114242, 0.126399",\
             "0.093746, 0.097281, 0.099437",\
             "0.083662, 0.087340, 0.089494");
   }
   ocv_sigma_fall_transition( 2D_lu_template ){
   index_1 ( "0.320000, 0.480000, 0.640000");
   index_2 ( "0.019098, 0.259098, 0.509098");
   values (  "0.100863, 0.114242, 0.126399",\
             "0.093746, 0.097281, 0.099437",\
             "0.083662, 0.087340, 0.089494");
   }

… …
…..
```

```
      } /* end of timing */
   }

pin/bus/bundle(<name>) {
  direction: input/inout;
      timing() {
   ocv_sigma_rise_constraint( 2D_lu_constraint_template ){
     index_1 ( "0.320000, 0.480000, 0.640000");
     index_2 ( "0.320000, 0.480000, 0.640000");
     values ( "0.089270, 0.080001, 0.073321");
   }
   ocv_sigma_fall_constraint( 2D_lu_constraint_template ){
     index_1 ( "0.320000, 0.480000, 0.640000");
     index_2 ( "0.320000, 0.480000, 0.640000");
     values ( "0.089270, 0.080001, 0.073321");
   }
  … …
 ….
      } /* end of timing */
      … …
      ….
} /* end of pin */
… …
….
….
} /* end of cell */
… …
….
} /* end of library */
```