cadence®

Virtuoso Variety Reference Manual

Product Version 16.1
March 2017

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

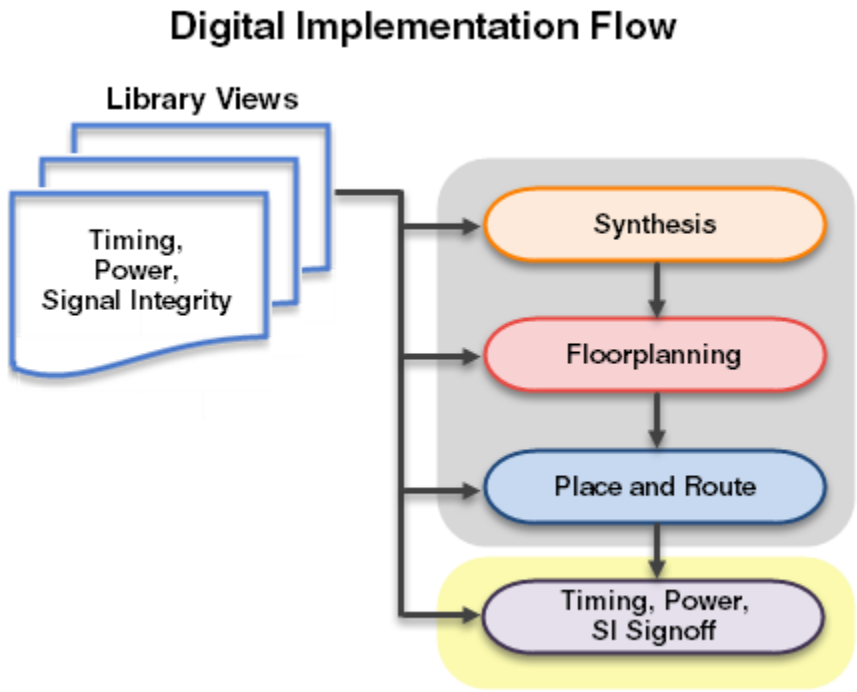
3
Variety Commands 31

Preface

Introduction to Characterization

The Role and Importance of Libraries

Creation of electrical views is a prerequisite for any digital design flow. The electrical information stored in the library views is used throughout design implementation from logic synthesis, through design optimization to final signoff verification. Accurate library view creation is essential to ensure close correlation between the design intent and the final silicon.

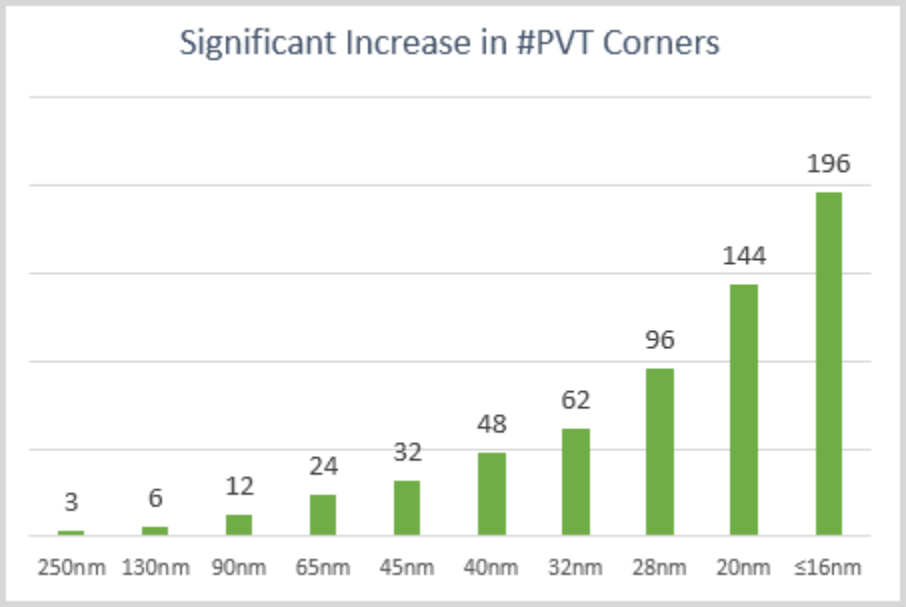


A Growing Problem

In nanometer geometries (65nm or below), the required number of library views is growing dramatically because of issues related to power leakage and process variation. To minimize

power leakage at deep submicron nodes, we see process variations such as LVT, RVT, and HVT (low/regular/high voltage) being utilized. For example, to manage power at 65nm, it is common to have library cells with two or three different threshold values (high threshold to reduce leakage power, low thresholds to improve performance), and to use two or more on-chip supply voltages. In this scenario, the number of views needed for 65nm will be six times greater than what is needed for 130nm.

The figure below shows the growing trend that requires PVT corners to accurately model the circuit behavior:

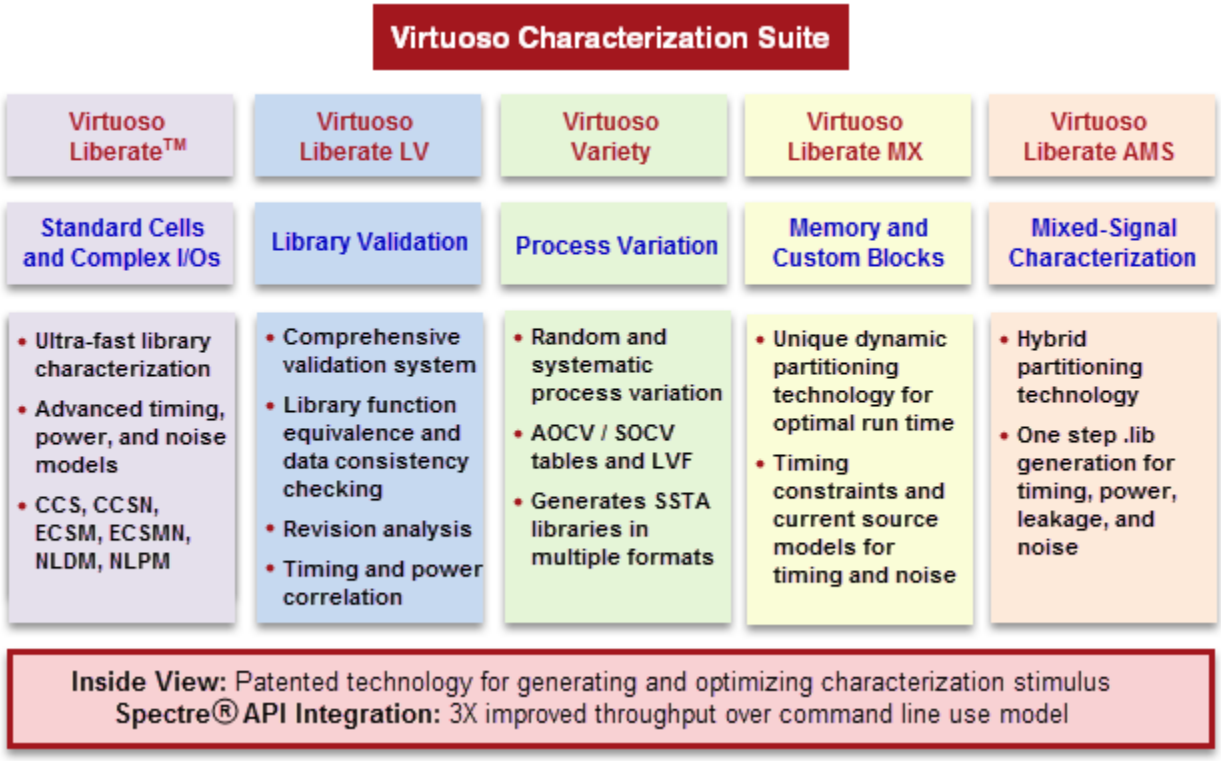


In addition, library views require more advanced models like:

- Current source models CCS and ECSM
- Statistical models – AOCV/SOCV/LVF
- Netlist extraction at various temperatures for Nanometer Process Nodes
- Support multiple foundries to assure flexibility for yield issues
- Support for many more functional designs – 1000+ STD cell, I/O, custom datapath, memory and Analog IP

Virtuoso Characterization Suite

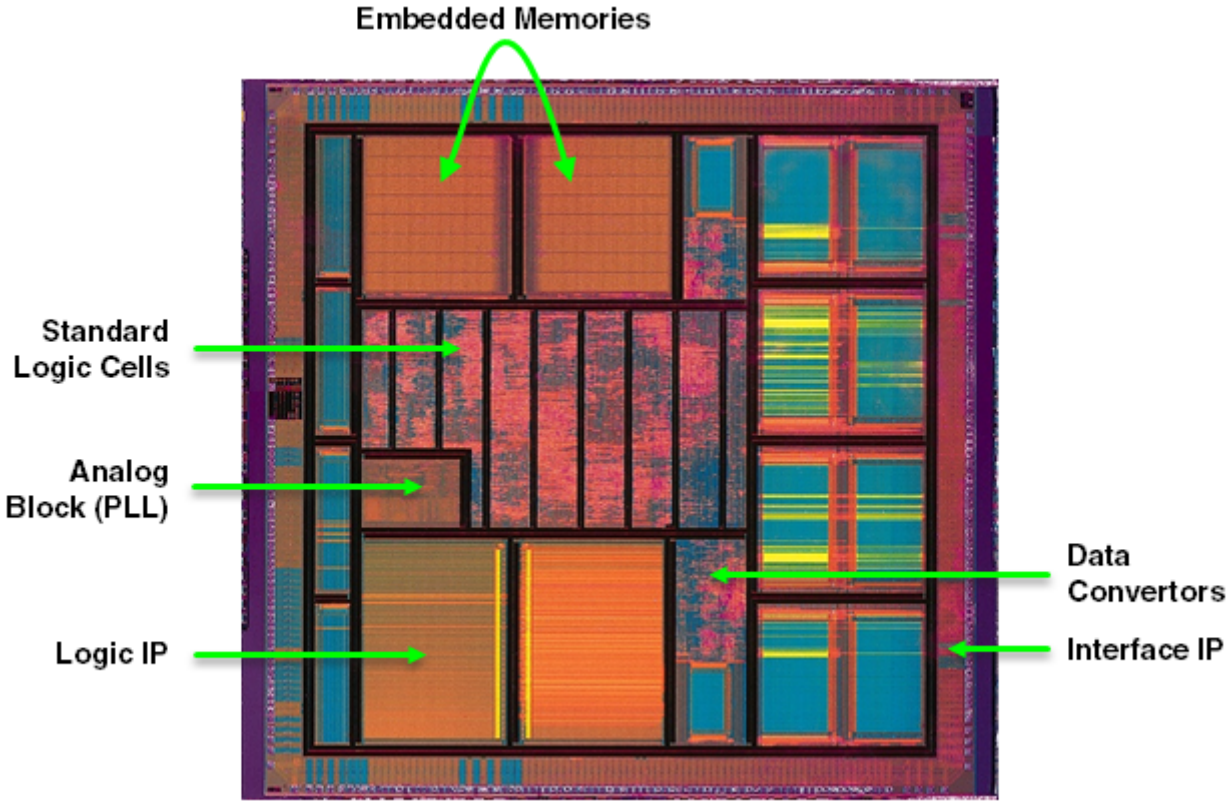
To address all the challenges, Cadence offers Virtuoso® Characterization Suite that covers the complete portfolio of characterization solutions given below:



The Virtuoso Characterization Suite intends to provide highly efficient and automated electrical view creation and validation for all IP blocks that include the following:

- Logic and I/O cells (GPIO, PCI, SSTL, PECL, and so on)
- Embedded Memory (SRAM, ROM, Register files, CAM, and so on)
- Custom digital blocks (custom cells, datapath, cores, and so on)

- Interface IP and analog blocks (USB, Serdes, DDR, and so on)



System Requirements

Liberate, Variety, Liberate MX, Liberate LV, and Liberate AMS run exclusively on Linux operating system. The following table lists the supported platforms:

Architecture	Development OS	Supported Environments
x86_64 (32/64)	RHEL 5.5	RHEL 6
		SLES10
		SLES11

For detailed information about the requirements, see Computing Platforms.

Software and Licensing Requirements

- LIBERATE 15.1

  The following table lists the required server and client product numbers for each product in the Virtuoso Characterization Suite:

Product Name	Server Product Number	Client Product Number
Liberate	ALT110	ALT111
Variety	ALT210	ALT211
Liberate MX	ALT410	ALT411
Liberate LV	ALT610	ALT611
Liberate AMS	ALT810	ALT811 or ALT812

- MMSIM 15.1

Product Name	Product Number
Spectre XPS	91600 or 90004
Spectre APS	3500 (restricted for characterization), 91050, or 90004

About This Manual

The *Virtuoso Variety Reference Manual* describes the Cadence® Virtuoso® Variety tool. The manual includes opening chapters that describe what Variety does and how to get started with the tool. Later chapters discuss the commands and variables that can be used with Variety.

Audience Profile

This manual is aimed at developers and designers who want to work on process variation aware timing models. It assumes that you are familiar with:

- SPICE simulations
- Basic expected behavior of the design being used

Additional Documents for Reference

For information about known problems and solutions, see *Virtuoso Characterization Suite Known Problems and Solutions*.

For a list of new features in a release, see *Virtuoso Characterization Suite What's New*.

For information about other products in Virtuoso Characterization Suite, refer to the following manuals:

- *Virtuoso Liberate Reference Manual* describes the Liberate tool—an accurate, highly efficient and easy-to-use library characterizer that creates electrical views (timing, power, and signal integrity) in formats such as the Synopsys Liberty (`.lib`) format.
- *Virtuoso Liberate LV Reference Manual* describes the Liberate LV library validator—a tool that provides a collection of capabilities used to validate and verify the data consistency, accuracy, and completeness of cell libraries.
- *Virtuoso Liberate MX Reference Manual* describes Liberate MX—a tool that provides library creation capabilities to cover memory cores.
- *Virtuoso Liberate AMS Reference Manual* describes Liberate AMS—a tool that provides library creation capabilities for Analog Mixed Signal (AMS) macro blocks.
- *Virtuoso Liberate API Reference Manual* describes a Tcl interface that allows access to the Liberate characterized Library DataBase (LDB).

Rapid Adoption Kits

Cadence provides Rapid Adoption Kits that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

Typographic and Syntax Conventions

This section describes the typographic and syntax conventions used in this manual.

`literal`	Non-italic words indicate keywords that you must enter literally. These keywords represent command or variable names.
`argument`	Words in italics indicate text that you must replace with an appropriate value.
`< >`	Angle brackets indicate text that you must replace with a single appropriate value. When used with vertical bars, they enclose a list of choices from which you must choose one.
`|`	Vertical bars separate a choice of values. They take precedence over any other character.
`-`	Hyphens denote arguments of commands or variables. Usually arguments denoted in this way are optional but, as noted in the syntax, some are required. The hyphen is part of the name and must be included when the argument is used.
`{ }`	Braces indicate values that must be denoted as a list. When used with vertical bars, braces enclose a set of values from which you must choose one or more. When you specify a list, the values must be enclosed by either quotation marks or braces. For example, `{val1 val2 val3}` and `"val1 val2 val3"` are legal lists.

Some argument are positional and must be used in the order they are shown. Any positional arguments that are used must be given after any arguments denoted with hyphens.

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

  Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit: https://www.cadence.com/support

- Log on to Cadence Online Support

Customers with a maintenance contract with Cadence can obtain the latest information about various tools at: https://support.cadence.com

Feedback about Documentation

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for
- Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support Product Manuals page, select the required product and submit your feedback by using the *Provide Feedback* box.

1

Introduction

This chapter gives an overview of the Variety process variation cell characterizer.

Statistical static timing analysis (SSTA) offers a number of advantages over traditional corner-based static timing analysis. Most notably, it provides a more realistic estimate of timing relative to actual silicon performance, typically reducing the worst-case timing margin by 10-15%. Armed with a better answer, designers can focus their optimization efforts on the paths that have the biggest impact on overall performance, rather than focusing solely on single extreme corners. The benefit of using SSTA is a higher-yielding design that can be taped out sooner.

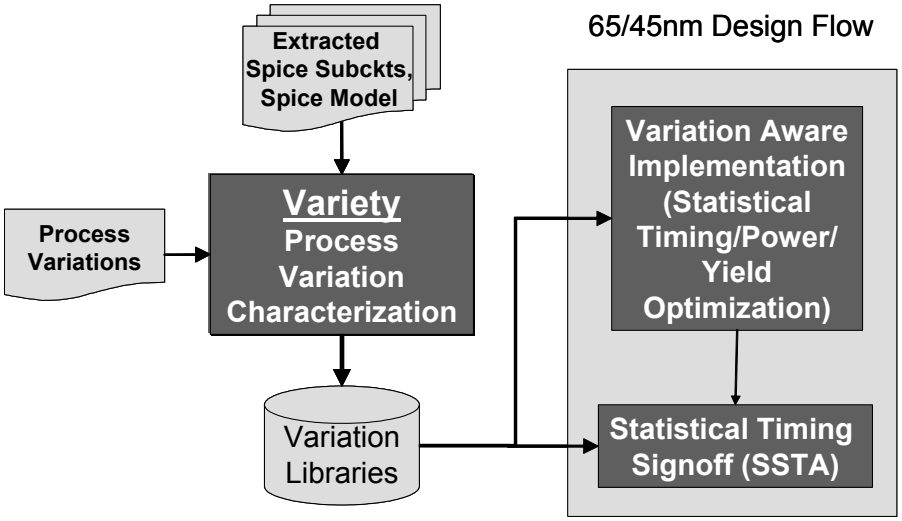


Figure 1-1 Variety in the Design Flow

To accurately predict variation, SSTA needs variation-aware timing models that account for both systematic process variations (such as, variations due to lithography) and random process variations (such as, variations due to doping).

Variety creates such models by pre-characterizing each cell under different loading and slew conditions for a given set of process parameter variations. Each process parameter set can

vary either systematically or randomly with the amount of variation based on actual process measurements.

Parameter variations can be treated as uncorrelated or correlated. Uncorrelated parameter sets are characterized independently while correlated parameter sets are characterized together. Typically, the key process parameters for SSTA analysis are transistor channel length (L) and threshold voltage (V_{th}), although Variety can model any process parameter, including abstract parameters created from principal component analysis (PCA).

For systematic variation, (also known as *inter-cell variation*) the process varies in the same direction by the same amount for each transistor inside a cell. Systematic variation can be used to model both on-chip and off-chip variation. For each set of systematic parameters every delay, transition, setup/hold, and pin capacitance table-entry is re-characterized. To specify a systematic process parameter set, use the `define_variation -type` option set to the `systematic` value.

For random variation, (also known as *intra-cell variation* or *mismatch*) the process variations apply to each transistor independently. Even for neighboring transistors inside the same cell, the variation may be different in direction and magnitude. As device geometries decrease in size, small imperfections in chip manufacturing mean that these random variations have an increasing impact. At 90nm, random variation accounts for up to 50% of the total process variation, and this is projected to increase as geometries continue to shrink. To characterize random variation efficiently, Variety deploys a number of techniques to avoid having to characterize every transistor uniquely for every table entry. The result of random characterization is the standard deviation of all of the individual effects from each transistor. To specify a random process parameter set, use the `define_variation -type` option set to the `random` value.

Variety models the impact of each parameter set as a non-linear sensitivity to the nominal values. The characterized variation data is stored in a central library database (LDB) from which various SSTA tool formats can be generated. Variety supports XT format (used by Extreme Design Automation), S-ECSM format (used by Cadence Design Systems) and CCS VA format (used by Synopsys Inc).

2

Getting Started with Variety

This chapter describes how to start using Cadence® Virtuoso® Variety. A systematic approach to tool setup is covered here with the intent to help new users of the tool. Once you are familiar with the tool, these can be refined.

Tool Installation and Setup

Installing Variety

To install Variety:

1. Familiarize yourself with the installation tools, InstallScape, and the license manager. To find guidance materials for these tools, see https://support.cadence.com.
2. To obtain the Variety software, see https://downloads.cadence.com.
3. Select the *LINUX* tab.
4. Select the appropriate release (for example, `LIBERATE161`).

   The first two numbers in the release name designate the year of the release and remaining numbers begin with one and increment with each additional release during that year. Therefore, `LIBERATE161` is the first LIBERATE base release of 2016.
5. Download and install the product.

   This step utilizes the tools, InstallScape and the license manager, that you learned about in step 1.
6. Use commands such as the following to include Variety in your software path.

   ```
   % setenv ALTOSHOME <install_dir>/<liberate_release_name>
   % set path=($path $ALTOSHOME/bin)
   ```
7. Set the following to include integrated Spectre in your executable path:

   ```
   % set path ($path $ALTOSHOME/tools.lnx86/spectre/bin)
   ```

Managing Licenses in Variety

Variety uses a server/client licensing scheme. A server license is used for invoking and monitoring the characterization run on the server machine while the client licenses are used for running simulations on the client machines and for any database post-processing. Each Variety server can access all the available client licenses. For example, with two server licenses and forty client licenses the following configurations are all valid:

- A single characterization run using 40 client processes.
- Two simultaneous characterization runs, each with 20 client processes.
- Two simultaneous characterization runs, 1 with 30 client processes, 1 with 10 client processes.

Important

Ensure that the license daemon (`cdslmd`) and the license server (`lmgrd`) have the same version and that this version is the same as that required for a release. For example, v11.11.1 is required for the Liberate 15.1 release. If a mismatch is detected, unexpected license behavior might be observed. For example, the license search path can be reset to `<none>` after a failed license check out request. This can result in incorrect license checking in process.

On a 64-bit license host, the 64-bit `cdslmd` and `lmgrd` must be used instead of the default 32-bit ones.

Waiting for a License

When you submit a Variety job, a request is made for a license. If you want the tool to wait until a license becomes available, set the following environment variable:

```
setenv ALTOS_QUEUE 1
```

Environment Variables for Controlling Licensing Checks

- ALTOS_LIC_MAX_TIMEOUT

  ```
  setenv ALTOS_LIC_MAX_TIMEOUT <value>
  ```

  where;

  value is duration in seconds.

  This shell variable specifies how long Liberate (both server and client) will wait to obtain a license.

For a server process, if the ALTOS _QUEUE variable is enabled, Liberate will attempt to check out 1 Server license. If the max timeout is reached, and no server license has been checked out, then Liberate will reset the timer and loop back to continue waiting for a license. For a client, when the max timeout is reached and at least one license was checked out, then the Liberate client will start to run with the licenses it has. No additional licenses are checked out.

- ALTOS_LIC_CHECK_ALT_TIMEOUT

  ```
  setenv ALTOS_LIC_CHECK_ALT_TIMEOUT <value>
  ```

  where;

  value is duration in seconds.

  Some Cadence characterization products can run using more than one product license. This variable controls both the server and client timeout before trying to check out an alternative license feature if there are any such licenses in the license pool.

System Libraries

Variety ships enabled with dynamically linked system libraries. To verify Variety is capable of running on your system, just try executing it. If Variety fails to start properly, you might have an old system with missing or incorrect system libraries. To check that your system libraries are compatible with Variety:

- Verify that the following libraries are included in `$ALTOSHOME/lib`, which is referenced automatically by the `liberate`, `variety`, and `lcplot` commands.

  ```
  libgcc_s-3.2.3-20040701.so.1
  libstdc++.so.5
  libtcl8.4.so
  libgcc_s.so.1
  libstdc++.so.5.0.3 libtcl.so
  ```

- Use `/usr/bin/ldd` to check library dependency, skip any missing library that did not come with our distribution and make sure all the libraries are accessible.

  ```
  ldd $ALTOSHOME/bin/lnx86_32_d/variety.exe
  ```

- Use `/usr/bin/objdump` to check GLIBC compatibility by examining the `Version References` section.

  ```
  objdump -p $ALTOSHOME/bin/lnx86_32_d/variety.exe
  ```

If it is confirmed there are Variety compatibility issues, try using statically linked libraries. To do this, edit each of the command scripts in the `$ALTOSHOME/bin` `liberate`, `variety`, and `lcplot` directories and change `ltype` from `d` to `s`. This changes the scripts to use binaries built using statically linked libraries.

Inputs Required for Variety Characterization

You need the following inputs to run Variety:

1. Extracted standard cell netlists in SPICE format.
2. Foundry device models in SPICE format.
3. A Variety command file in Tcl format that includes the amount of variation to model for each key parameter.

Extracted Cell Netlist

The transistors, diodes, resistors, capacitors, and extracted parasitic elements (RCs) that compose the cell are passed to Variety in SPICE format. Extracted SPICE netlists can be created directly from the cell layout by device and interconnect parameter-extraction tools. Standard Berkeley SPICE format and HSPICE® netlist formats are supported. Multiple cells can be specified in a single file or as a group of files. Each cell to be characterized must have a `.subckt` definition in the files passed to Variety. To specify the cell netlists, use the `read_spice` Tcl command.

```
read_spice {nand2x4.spi nor2x2.spi}
```

Device Models

The device models are supplied by the foundry and represent the electrical parameters of the target process. The device models include models from transistors (P and N channel), diodes, capacitors, and resistors. Most device model files include different parameters for different process corners, such as a typical corner, a fast corner, and a slow corner. To read a device model into Variety use the read_spice Tcl command.

```
read_spice {nand2x4.spi nor2x2.spi} -model models.spi
```

To specify the voltage and temperature to use for characterization use the `set_operating_condition` command:

```
set_operating_condition -voltage 1.2 -temp 25
```

While Variety can be used with any SPICE model, it is better to use a *statistical* SPICE model. Statistical SPICE models are designed to be used with Monte Carlo simulation and consequently are pre-programmed with defined parameter distributions. This makes it easy to vary parameters by sigma values rather than absolute values. In addition, statistical SPICE models can contain intermediate PCA parameters that can be varied independently without requiring any parameter correlation information.

Tcl Command File

Variety uses the Tcl scripting language to control the characterization process. The Tcl script is used to specify the cell netlists, SPICE models, and operating conditions. In addition, the Tcl script defines the range of data that the characterization is to be performed over, such as input slew and output loading conditions, and defines the different parameter variations that need to be modeled. Variety simulates and measures each cell using each of the specified input slews and loads and generates the appropriate delay tables, timing checks (setup, hold, etc.), and pin capacitance information. The Tcl commands available for controlling Variety are detailed in Chapter 3, “Variety Commands.”

A sample Tcl script for running Variety is shown below. This script characterizes the cells `NAND2x4`, `NOR2x2`, and `DFFX1` accounting for process variations in transistor length, width, threshold voltage, and oxide thickness.

```
# Define the templates for characterization
# Delay template for 3 input slews and 3 loads
define_template -type delay \
    -index_1 {0.025 0.1 0.25} \
    -index_2 {0.0010 0.015 0.100} \
delay_3x3

# Timing constraint template for 3 input slews
define_template -type constraint \
    -index_1 {0.025 0.1 0.25} \
    -index_2 {0.025 0.1 0.25} \
constraint_3x3

# Specify the PVT for this characterization run
set_operating_condition -voltage 1.2 -temp 25 -process TT

# Read in the SPICE subckts and models
read_spice {nand2x4.spi nor2x2.spi dffx1.spi models.spi}

# Define how to characterize each group of cells
define_cell \
    -input {A1 A2 D} \
    -output {Z Q QN} \
    -clock {CK} \
    -async {SN} \
    -delay delay_3x3 \
    -constraint constraint_3x3 \
    {NAND2X4 NOR2X2 DFFX1}

# Define random N & P Vth variation
define_variation -type random -pelgrom { \
    dvthn 3.0e-9 \
    dvthp 4.0e-9 \
} RANDOM_VTH

# Define correlated systematic length & width variation
define_variation -type systematic { \
    dxl 1.00e-9 \
    dxw 2.50e-9 \
} SYSTEMATIC_WL

# Define systematic oxide thickness variations
# that are uncorrelated between N & P
```

```
define_variation -type systematic { toxn 3.00e-9 } \
SYSTEMATIC_NTOX

define_variation -type systematic { toxp 3.00e-9 } \
SYSTEMATIC_PTOX

# Perform characterization and write out the library
char_variation -ccs -ecsm
write_ldb ssta.ldb
write_variation -extreme xt.lib
write_variation -ecsm ecsm.lib
write_variation -ccs ccs.lib
```

Variety can automatically create a list of template and cell definitions from an existing library in Liberty format. An example Tcl file for template creation is show below:

```
# Read in an existing library and create templates
read_library existing.lib
write_template variety_templates.tcl
```

The above creates a file called `variety_templates.tcl`. This file can be used in a subsequent Variety characterization run. For example:

```
# Read templates and cell definitions for characterization
source variety_templates.tcl

# Specify the PVT for this characterization run
set_operating_condition -voltage 1.2 -temp 25 -process TT

# Read in the SPICE subckts and models
read_spice {nand2x4.spi nor2x2.spi dffx1.spi} \
    -model models.spi

# Define correlated random N & P Vth variation
define_variation -type random -pelgrom { \
    dvthn 3.0e-9 \
    dvthp 4.0e-9 \
} RANDOM_VTH

# Define correlated systematic length & width variation
define_variation -type systematic { \
    dxl  1.5e-9 \
    dxw  2.5e-9 \
} SYSTEMATIC_WL

# Perform characterization and write out the library
char_variation
write_ldb ssta2.ldb
write_variation -extreme secsm.lib
```

Running Variety

Before using Variety, make sure that it is installed correctly and that all the necessary prerequisite data are available.

To use the 64-bit port of Variety, set the `ALTOS_64` environment variable prior to running the tool, as shown below:

```
% setenv ALTOS_64 1
```

To perform a characterization, type `variety` followed by the Tcl command file. A trial run of Variety can be performed as follows:

```
% cd $ALTOSHOME/examples/variety
% variety char.tcl
% vi example.lib
```

By default, Variety utilizes `stdout` and `stderr` for all messages and does not create a log file. However, to run Variety so that it uses a log file, you can use a command such as following:

```
% variety char.tcl |& tee char.log
```

Note: For the log file to be complete, both `stderr` and `stdout` must be captured.

3

Variety Commands

This chapter describes the Tcl commands that control library creation.

Note: The command options that are prefixed with a hyphen (-) are optional except where explicitly indicated.

a...	
aocv_add_attribute	aocv_read_ldb
aocv_check_monotonicity	aocv_set_cell_pin
aocv_define_temp_range	aocv_set_driver_receiver
aocv_derate_copy_cell	
c...	
char_variation	compare_library
check_delay_monotonicity	create_aocv_table
compare_ccs_nldm	
d...	
define_arc	define_max_transition
define_cell	define_pin_load
define_index	define_template
define_input_waveform	define_variation
define_leafcell	define_variation_average
define_map	define_variation_factor
define_max_capacitance_limit	define_variation_group
g...	
get_var	

p…	
printvars	
r…	
read_ldb	read_vdb
read_library	read_truth_table
read_spice	
s…	
select_index	set_network_port
set_client	set_operating_condition
set_conditional	set_pelgrom_equation
set_constraint	set_pin_capacitance
set_constraint_criteria	set_pin_gnd
set_default_group	set_pin_vdd
set_dependent_load	set_units
set_driver_cell	set_var
set_gnd	set_vdd
w…	
write_ldb	write_variation
write_socv	write_variation_table
write_template	write_vdb

aocv_add_attribute

Adds the specified attributes to each derate table.

Options

`-set`	Sets only the attribute. This will remove all previously added attributes.
`-cells {list}`	Adds the AOCV attributes to a list of cells. Default: All cells
`{attribute_list}`	Lists of attributes to add or set for each derate table.

aocv_check_monotonicity

Checks that the data in the advanced on-chip variation (AOCV) table is monotonic. With this command, you can also fix monotonicity problems.

Options

`-fix`	Fixes any monotonicity problems in the derate table.
`-max_early_derate <val>`	Specifies the maximum early derate allowed. Default: `1e20`
`-min_late_derate <val>`	Specifies the minimum late derate allowed. Default: `-1e20`

aocv_define_temp_range

Specifies the temperature range to be used for calculating the temperature derate table.

Options

{*temp1 temp2*}	Specifies the temperature range for temperature advanced on-chip variation (AOCV).
temp1	Specifies the low end of the range to be used for temperature AOCV.
temp2	Specifies the high end of the range to be used for temperature AOCV.

If a range is set, the *value* option from the define_variation command is used as the step value.

Example

In this example, Variety sweeps from temp1 up to temp2 with the step value, calculates the deltas, then uses the largest deltas to calculate the temperature derate table.

```
define_variation -margin -type systematic { temp value } temp
aocv_define_temp_range temp1 temp2
```

aocv_derate_copy_cell

Copies the derate table from the cell specified with -from to all the cells specified in the -to list.

Options

-from <*cell_name*>	Specifies the cell with the derate table that is to be copied.
-to {*cell_names*}	Specifies the list of cells to copy the derate table to.

aocv_read_ldb

Reads an LDB that was created during a previous create_aocv_table command execution. Variety uses the loaded data as a starting point and adds additional, uncharacterized arcs and cells to the existing data. You can also use this command for the following actions:

- Remove specific arc data and complete cell data from the LDB. This will result in the specified arcs and cells being recharacterized in the current create_aocv_table run.

- Restart a `create_aocv_table` run if something goes wrong in the first run, or if more cells need to be added to the AOCV LDB.

Note: The same cell with different `chain_length` is treated as a different cell and is run in a separate `create_aocv_table` command. The new cell data will be stored in the LDB with an extended name.

This command must be specified before the `create_aocv_table` command is run.

Options

`-remove_arc {cell1 pin1 related_pin1 ...}`

A list of arcs to remove from the LDB to ensure they are re-characterized. These should be specified in the following format:

```
{cell1 pin1 related_pin1 cell2 pin2
related_pin2 ...}
```

`-remove {`*`list of cells`*`}`

A list of cells to remove from the LDB.
Default: `(none)`

This option can be used to enable the re-characterization of the specifed list of cells. By default, Variety does not characterize any cell that has already been loaded from an LDB. If the cell is removed from the LDB during loading, then the cell is re-characterized.

This option overwrites the `-remove_arc` option if the same cell name is specified in both.

`<`*`filename`*`>` A library database file in LDB format.

aocv_set_cell_pin

Specifies a list of cells and their pins to select the arcs for Advanced On-Chip Variation (AOCV) modeling. This can improve run time when modeling AOCV used by the create_aocv_table and write_variation_table commands.

Options

`-cells {`*`cell_names`*`}`

Specifies a list of cells for AOCV.

`-exclude` Excludes the pins specified using the `-pins` and `-related` options while modeling AOCV for the specified `-cells`. The other pins in the cells that are not listed are modeled.

`-pins {`*`pin_names`*`}` Specifies the list of pins for AOCV.

`-related {`*`pin_names`*`}`

Specifies the list of related pins for AOCV.

aocv_set_driver_receiver

Specifies the information that is required to enable sequential cell advanced on-chip variation (AOCV) characterization.

Options

`-driver {`*`cell input output`*`}`

Specifies the name of the driver (sequential and level shifter) cell, and its input and output pins.

`-extra_driver {`*`cell input output`*`}`

Specifies a unique cell to be used as an *extra driver*. This option is useful when characterizing a level shifter to specify a different extra driver cell. If this option is not specified, then the command will default to the cell specified in the `-receiver` option.

An extra driver is a driver to be added before the stage where the measurements begin. Use the aocv_extra_driver variable to specify the number of extra driver cells to be used in series.

Important Points to Note:

- Set the `aocv_extra_driver` variable before using the `aocv_set_driver_receiver` command with this option; otherwise, this option will be ignored.

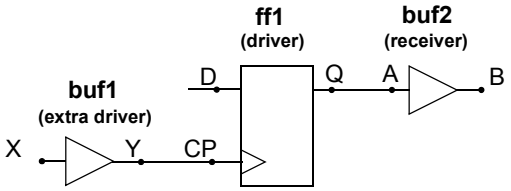
- This option is often used when characterizing cells that use different power domains on the input and the output, such as level shifters. The set_pin_vdd command should be used to specify the voltages of each input and output pin of the characterized cell.

`-receiver {`*`cell input output`*`}`

Specifies the name of the receiver cell, and its input and output pins.

Example

```
aocv_set_driver_receiver \
    -extra_driver {buf1 X Y} \
    -driver {ff1 CP Q} \
    -receiver {buf2 A B}
```



char_variation

Performs variation model characterization.

Arguments

`-auto_index`

Creates table indices for all constructs, overriding the values specified in the given templates. The number of entries for each index is taken from the appropriate predefined template. This feature uses the *`value`* of the `max_transition` variable to determine the range of output loads for each cell.

`-auto_max_capacitance`

Automatically computes the pin-based *max_capacitance* attribute by using the same method as the `-auto_index` option. This incurs the same increase in the run time as that incurred when the `-auto_index` option is enabled. If both `-auto_max_capacitance` and `-auto_index` are enabled at the same time, `-auto_index` takes precedence.

`-ccs`
Enables statistical characterization of CCS current waveforms and creates a library with CCS data. To write the data out as a CCS VA library, use the `write_variation -ccs` option. Default: Variety characterizes for waveform sensitivity, which is sufficient for ECSM and Extreme formats.

`-cells {`*`cell_names`*`}`
Instructs Variety to characterize only the cells named in the *`cell_names`* list. Each cell listed in a `define_cell` command is characterized, providing the SPICE `subckt` for that cell is defined in the netlists passed to the `read_spice` command. Default: Variety characterizes all the cells defined by `define_cell`.

When used with the `-exclude` option, Variety excludes the cells in the list from variation characterization.

`-ecsm`
Enables characterization of ECSM timing data. This option is enabled by default and is provided for script readability.

`-exclude`
Reverses the meaning of the `-cells` list, so that the specified list of cells are excluded from comparison.

`-extsim <`*`name`*`>`
Specifies an external SPICE simulator (for example, Spectre, Spectre APS, or Spectre XPS) to be used for systematic variations. The license for the external simulator must be available and the external simulator must be included in the search path. Using an external SPICE simulator typically increases the characterization run time by a factor of three or more as compared to the internal SKI simulator.

`-io`
Enables characterization of I/O cells without the use of *Inside View*. Using this option turns off automatic arc-determination and vector-generation, so, for I/O cells, each of the arcs and associated logic conditions must be expressed explicitly using `define_arc` commands.

`-monte`
: Instructs Variety to use Monte Carlo-based simulation for characterization of delay and the following constraint sigmas: `setup`, `hold`, `recovery`, `removal`, `non_seq_setup`, `non_seq_hold`, `nochange`, and `path_delta`. The `mpw` constraint is not supported.

  When `-monte` is used, the `char_variation -extsim` option must be set to `spectre` or `hspice`.

  For usage notes related to the `-monte` option, see Using Monte Carlo Characterization Flow.

`-monte_trials`
: Specifies the number of Monte Carlo trials to be performed by the external simulator. Default: `2000`

`-skip { constraint | delay | hold | hold_only | mpw | non_seq_hold | non_seq_setup | recovery | removal | setup | setup_only }`
: Disables characterization of specific categories of data. The list can can contain one or multiple categories to be skipped during characterization. Default: none (do not skip any data)

  `constraint`
  : Variety skips all constraint timing types: `setup_rising`, `setup_falling`, `non_seq_setup_rising`, `non_seq_setup_falling`, `recovery_rising`, `recovery_falling`, `hold_rising`, `hold_falling`, `non_seq_hold_rising`, `non_seq_hold_falling`, `removal_rising`, and `removal_falling`.

  `delay`
  : Variety skips the following delay types: `cell_rise`, `cell_fall`, `rise_transition`, and `fall_transition`.

  `hold`
  : Variety skips `hold_rising`, `hold_falling`, `non_seq_hold_rising`, `non_seq_hold_falling`, `removal_rising`, and `removal_falling`.

  `hold_only`
  : Variety skips `hold_rising` and `hold_falling`.

`mpw`	Variety skips all `mpw` timing characterization. This includes attributes and `mpw` tables.
`non_seq_setup`	Variety skips `non_seq_setup_rising` and `non_seq_setup_falling`.
`non_seq_hold`	Variety skips `non_seq_hold_rising` and `non_seq_hold_falling`.
`recovery`	Variety skips `recovery_rising` and `recovery_falling`.
`removal`	Variety skips `removal_rising` and `removal_falling`.
`setup`	Variety skips `setup_rising`, `setup_falling`, `non_seq_setup_rising`, `non_seq_setup_falling`, `recovery_rising`, and `recovery_falling`.
`setup_only`	Variety skips `setup_rising` and `setup_falling`.

`-server_thread <value>`

Specifies the number of threads (multithreaded CPUs) to be made available for the server to use. The extra threads are used to characterize the active driver (see set_driver_cell) waveforms. Default: use the same value as that of the `-thread` option.

Note: If the server is submitted to a queuing system, the `-server_thread` value and the number of threads requested from the queuing systems should be the same positive integer value (`!=0`); otherwise, the server can get overloaded.

`-skip_constraint_variation`

Disables characterization of variation impact on timing constraints such as setup and hold. Because the process of characterizing constraints contributes a significant portion of total run-time, using this option can greatly reduce characterization time.

`-skip_variation { constraint | hold | mpw | setup }`

	Skips characterization for the corresponding sensitivity of the specified skip type and sets the sensitivity for that specified skip type to 0. Default: nothing is skipped
`-thread <`*`number`*`>`	Specifies the maximum number of CPU threads to use on the current machine. Default: Variety automatically uses multiple threads based on the available CPUs. Running on two or more threads provides a significant reduction in characterization time.
`-trial`	Generates a *dummy* database that supports only NLDM format library data. This option instructs Variety to run the entire preprocessing without running the actual simulations. The database consists of NLDM format library data with proper structure, but dummy data values. All other output formats are disabled including CCS, CCSN, CCSP, ECSM, ECSMN, and ECSP. When combined with the write_ldb and write_variation commands, use of this option results in a library file that is structurally valid. This library can be used with commands such as write_template and `write_verilog`.
`-user_arcs_only`	Specifies to characterize only the user-specified arcs. When you specify this option, the *Inside View* of Variety skips the automatic addition of arcs. To use this option, you must provide all the required arcs using the `define_arc` command. This option is used to ensure that the `write_template` verbose flow matches the reference library structure more closely.

Only one `char_variation` command is permitted per Variety run.

Using Monte Carlo Characterization Flow

Use the `-monte` Boolean option with the char_variation command to instruct Variety to utilize the simulator native built-in Monte Carlo analysis. The resulting data is reported at `1` sigma and is available in the `.ldb` file. This `.ldb` file is used to generate Liberty files with LVF data. The compare_library command will compare a library created using sensitivity-based analysis with the one generated using `-monte` analysis. This provides a means to check the accuracy of the sensitivity algorithm to the Monte Carlo algorithm.

The Monte Carlo characterization mode is extremely run time heavy. However, you are advised that due to extremely long run times, using the Monte Carlo flow is not considered effective for characterizing a large full production library. Due to the long simulation run times, it might be necessary to adjust the settings of the packet_client_timeout and packet_client_timeout_action variables to ensure that the arc in question is not killed by Liberate. The `bsub -W` option can be used to override the LSF queuing system default allowed time for a job.

In addition, keep the following usage notes under consideration:

- The define_variation commands are not required when the `-monte` option is used with the `char_variation` command. If the `define_variation` command is specified, it will be ignored.
- The `char_variation -monte_trials` option specifies the number of Monte Carlo iterations. The `-monte_trials` value needs to be carefully chosen to represent enough sample points to provide accurate sigma.
- To get a two-sided early/late Monte Carlo distribution, the non_linear_random_variation variable must be set appropriately to enable non-linear characterization. By default, the `non_linear_random_variation` is one-sided (early and late are both the same).
- When using `-monte`, the process models should be in sync with your define_variation command in the sensitivity-based (not `-monte`) characterization flow. Any parameter differences could result in outliers. This is especially true for LVF- and AOCV-based formats that ignore the effect of systematic (also known as, *global*) variations. When correlating LVF and OCV data, load the models that enable only random (also known as, *local*) variations.
- An external simulator must be used (see `char_variation -extsim`). The currently supported external simulators are HSpice and Spectre.
- When using Spectre, the following variables can be used to tune the Monte Carlo-based simulation:
  - extsim_mc_type: This variable must be set to mismatch.
  - extsim_mc_append: This variable can be used to specify Monte Carlo directives to Spectre.
  - extsim_cmd_option: To decrease wall time, distribute each iteration (.alter) for an arc using `+mp=<number of alters>`. With this option, Spectre will automatically `bsub` each iteration and will collate and process the Monte Carlo data from each iteration. Spectre version 14.1 ISR15 or later must be used to enable the `+mp` option.

- Variety `-monte` automatically sets Spectre OPT1 windows based on estimation. The OPT1 settings can be overridden using the constraint_search_bound_estimation_mode and constraint_search_bound variables. For example:
- Use the extsim_monte_option variable to specify sampling settings when characterizing constraints. The different options that can be used are: `lds`, `lhs`, and `random`.

Spectre Example

The following statements enable Monte Carlo Latin hypercube sampling for `5000` sample points using Spectre:

```
set_var extsim_monte_option "sampling=lhs"
char_variation -monte -monte_trials 5000 -extsim spectre
```

Examples

```
# Automatically generate indices for characterization
char_variation -auto_index

# Perform statistical characterization for CCS char_variation -ccs
# Only characterize DFFX1 and INVX1
char_variation -cells {INVX1 DFFX1}

# Don't characterize NOR2X1
char_variation -cells -exclude NOR2X1

# Don't characterize constraint variation 1
char_variation -skip_constraint_variation
```

check_delay_monotonicity

Checks `cell_rise` and `cell_fall` delay data to ensure that all the entries are monotonically increasing with respect to output load.

Options

`-adjust <`*`value`*`>`	Specifies the amount to adjust the delay or transition when fixing non-monotonic data. Default: `1e-15` (seconds)
`-ecsm`	Checks ECSM waveform data.
`-exit`	Causes the `write_variation` command to exit if non-monotonic delay data (by load) is found. The warnings or errors are written to the screen and indicate the bad table entry, the values involved, and the arc type including the `when` condition. The `-exit` option overrides the `-fix` option.

`-fix`	Repairs any rise or fall transition and any delay monotonicity problems (with respect to output load) by making the non-monotonic table entry equal to the previous entry plus the `-adjust` value in seconds. The `-exit` option overrides the `-fix` option.
`-ocv`	Checks for OCV delay sensitivity if only the `-ocv` option is specified. If the `-transition` option is specified along with the `-ocv` option, the transition sensitivity is also checked.
`-slew`	Checks monotonicity with respect to input slew.
`-transition`	Checks rise and fall transition data.

The `check_delay_monotonicity` checks are performed as the library is being written out using the `write_variation` command.

The warnings and errors look like the following:

```
*Warning* (write_variation): Non-monotonic (by load) rise_transition values: (3,
4) 0.35 < 0.37 for DFFX1:CLK->Q
*Error* (write_variation): Non-monotonic (by load) cell_fall values: (2, 5) 0.254
< 0.257 for DFFX1:CLK->Q
```

Example

```
read_ldb my.ldb.gz
check_delay_monotonicity -ecsm -transition -fix
write_variation my.lib
```

compare_ccs_nldm

Compares the CCS data to the NLDM data in a single library and report differences that exceed the defined tolerances.

Options

`-absolute_average`	Reports absolute average. Default: report relative average
`-abstol <`*`value`*`>`	Defines absolute tolerance limits for the CCS vs. NDLM error for each comparison. Any comparison that exceeds both the `-abstol` and `-reltol` tolerances is considered an outlier and is reported. Default: `0.002 *` data_type_default_unit for each data type (delay and trans)

`-cells {`*`cell_names`*`}`	Specifies a list of cell names. Default: all cells
`-nworst <`*`number`*`>`	Specifies how many of the worst outliers to include in the summary for each data type. For each cell included in the summary, the worst absolute and relative outlier is reported. Default: 5
`-percent_max_diff`	Reports the percent error of the maximum difference. Default: report the max percentage difference
`-reltol <`*`value`*`>`	Defines relative tolerance limits for the CCS vs. NLDM error for each comparison. Any comparison that exceeds both the `-abstol` and `-reltol` tolerances is considered an outlier and is reported. Default: 0.02 for delay, 0.02 for trans
`-report <`*`filename`*`>`	Specifies the name of the output comparison file. Default: `<`*`library_name`*`>.cmp.txt`
`-verbose`	Generates a report showing every comparison, including those that did not exceed a tolerance. The output is written to the filename specified by the `-report` option.
`<`*`library_name`*`>`	(Required positional option) Specifies the library name.

Example

```
# Set relative tolerance to 1%, delay tolerance to 1ps
compare_ccs_nldm -reltol 0.01 -abstol "delay 1e-12" comp.lib
```

compare_library

Compares the comparison library against the reference library and report differences that exceed the defined tolerances.

Options

`-absolute_average`	Reports averages using absolute values. For example, assuming that one difference is `-3ps` and another is `5ps`, the calculation is done as following when `-absolute_average`:

is not used. *is* used.

$\frac{-3+5}{2} = \frac{2}{2} = 1$ $\frac{|-3|+5}{2} = \frac{8}{2} = 4$

`-abstol <`*`value`* | `{`*`type_and_value_list`*`}>`

Sets absolute tolerance differences for comparisons. Any comparison that exceeds both the `-abstol` and `-reltol` tolerances is considered an outlier and is reported. Default: `0.001` times the default unit for each data type. For example, if the `time_unit` is in `nS`, the `-abstol` for delay defaults to `0.001nS` or `1ps`.

This option accepts a single value or a paired list of type and value. Individual tolerances can be set for each different data type by assigning values to the following compare types:

`all`, `cap`, `ccs`, `ccs_cap`, `ccsn_dc`, `ccsn_vout`, `constraint`, `delay`, `ecsm`, `ecsm_cap`, `hyper`, `leakage`, `max_cap`, `max_trans`, `miller_cap`, `noise`, `power`, `siv`, `trans`, `timing`, `capacitance`, `voltage`, and `current`

If the option has only a single value, then the type for that value is assumed to be `all`. The `-abstol` value must be given standard units (not library units). For example, use `delay 5e=h12` to set the `-abstol` for delay to 5ps.

Set absolute difference tolerance. Default: `1e-3` of the default unit for the data type

`-cells {`*`cell_names`*`}`

Specifies a list of cell names to compare. Default: all cells are compared.

This option supports the use of a wildcard.

`-cell_map {`*`list`*`}`

Compares the specified pairs of `ref_lib` and `comp_lib` cells. Default: compare all matching cell names.

This option controls how the `compare_library` command chooses the cells that need to be compared. The following rules are considered to prioritise cell mapping:

- If the `-cells` and `-cell_map` options are specified together, each cell given in the `-cells` list is compared to the `-cell_map` list. If a cell in the `-cells` list maps to a valid pair in the `-cell_map` list, the mapped reference cell is compared with the comparison cell. One-to-many and many-to-one mapping of cells is allowed. Also, the comparison is done for all valid combinations. If a cell in the `-cells` list is not present in the `-cell_map` list, the comparison is done for the same cell in both the libraries.
- If only `-cell_map` is provided without `-cells`, each valid cell pair from the `-cell_map` list is compaired.
- If only `-cells` is provided without `-cell_map`, all cells in the `-cells` list that exist in both the libraries are compared.
- If both `-cells` and `-cell_map` are not provided, all cells that are present in the reference and comparison libraries are compared.

`-exact_match`

Compares arcs only when the logic (`when`) conditions are an exact match.

`-exclude`

Reverses the meaning of the `-cells` list, so that the specified list of cells are excluded from comparison.

`-format <txt | xls>`

Specifies the format for the output report. Default: `txt`

`htm`	Requests an HTML output format. The default directory name is `./html` and can be changed using the `-group` option. A one page comparison is generated for each cell group. Open the file `index.html` in a web browser to view the report.
`txt`	Requests a report formatted as standard text.

	`xls`	Requests a report in an output format that is suitable for import into Microsoft Excel.
`-group <`*`dirname`*`>`	Specifies the name of a directory to store cell comparisons for each cell group. `-group` requests a group-by-group comparison, storing the results in the given directory name. A cell group is determined by the `define_group` command or by the *cell_footprint* attribute. The comparison report for each group is stored in the file `<`*`dir_name`*`>/<`*`group_name`*`>.cmp.txt`. Default: all cells in a single report.	
`-gui <`*`filename`*`>`	Generates, and specifies a name for, an intermediate file that can be used for graphical comparisons of data with the `lcplot` utility.	
`-lcplot`	Uses the `lcplot` utility to display the comparison results graphically. The `-gui` option is not required because a comparison data file called `<`*`library_lib`*`>.gui` is automatically created.	
`-lib <abs | rel>`	Requests an output report formatted like the `comp.lib`, where the values in the data table represent the absolute or relative differences between the two libraries. The output report is named `<`*`comp.lib`*`>_<abs | rel>.cmp`.	
	`abs`	Values represent the absolute differences between the two libraries.
	`rel`	Values represent the relative difference between the two libraries.
`-multiple_matches`	Reports the results of comparing all arcs that have functional overlap with a reference arc. Default: reports the table that gives the best match. Multiple arcs are shown in the output file as (N of M) after the `when :` line. For example: `when : !M1 Vs (!(M1)*!(M2))(1 of 2), Timing : combinational` **Note:** The `-exact_match` option overrides the `-multiple_matches` option.	

`-nldm_only`
Requests that only the NLDM data is compared. Comparison of the following data is ignored:

- CCS and ECSM timing
- Noise and power constructs

`-no_interpolation`
Disables the comparison of data groups that have different indices, that is, no interpolation occurs between index points. Default: if the index values are different, the comparison values are interpolated.

`-nworst <number>`
Specifies how many of the worst delay difference and worst percent difference outliers to include in the summary for each data type (delay or leakage, for example) of each cell. For each cell included in the summary, the worst absolute and relative outlier is reported. Default: 5

`-ocv_include_nominal`

Compares the `(nominal+sigma)` and `(nominal-sigma)` values.
Default: compare `ocv_sigma` values.

The Liberty Variation Format (LVF) `ocv_sigma_*` table values can be very small. Comparing these values directly can lead to a significant number of outliers. Use this option to include the nominal delay in the comparison. This will reduce the number of outliers.

`-padding`
Pads delays, transitions and constraints by ½ input slew and pads power by an additional $\frac{1}{2}CV^2$ (where `C`=output capacitance, `V`=Vdd for that pin) before comparison. The padding does not apply to hidden power because the output is not toggling.

Padding is useful when comparing very small or even negative delay values.

`-percent_max_diff`
Reports the percent of the maximum difference. Default: reports the maximum of the percent difference.

`-reltol <value | {type_and_value_list}>`

Sets percentage tolerance differences for comparisons. Any comparison that exceeds both the `-abstol` and `-reltol` tolerances is considered an outlier and is reported. Default: 0.01 (1%).

This option accepts a single value or a paired list of type and value. Individual tolerances can be set for each different data type by assigning values to the following compare types:

all, cap, ccs, ccs_cap, ccsn_dc, ccsn_vout, constraint, delay, ecsm, ecsm_cap, hyper, leakage, max_cap, max_trans, miller_cap, noise, power, siv, trans, timing, capacitance, voltage, current

If the option only has a single value, the type for that value is assumed to be `all`.

`-report <filename>` Specifies the filename to be used for the output file. Default: `<comp_lib>.cmp.txt`

`-skip {list}` Specifies a list of data comparison types to skip. Default: none (do not skip any types).

By default, `compare_library` compares all of the variation data in the library (except CCS VA format). To compare variation in CCS format each unique CCS library created per parameter should be compared. The `parameters` type can be used to skip comparison of variation data and only compare nominal data. Valid comparison types are:

cap, capacitance, ccs, ccs_cap, ccsn, clear, ccsp, constraint, current, delay, ecsm, ecsm_cap, hidden_power, hold, hyper, leakage, max_cap, max_trans, mpw, noise, nonseq_hold, nonseq_setup, ocv_const, ocv_delay, ocv_trans, parameters, power, preset, removal, recovery, setup, siv, timing, trans, tristate, voltage

For convenience, you can also request subsets of these types by specifying the following values:

value	**subset**
`capacitance`	`{cap ccs_cap ecsm_cap ecsm_cap_variation in_cap max_cap miller_cap}`

`constraint`	`{setup hold recovery removal mpw nonseq_setup nonseq_hold}`
`current`	`{ccsn_dc ccsp siv}`
`delay`	`{ccs clear delay delay_variation ecsm ecsm_variation time_const preset three_state three_state_enable three_state_disable tristate}`
`power`	`{leakage}`
`retain`	`{ccs_retain retain retain_trans}`
`ssta`	`{delay_variation ecsm_cap_variation ecsm_variation trans_variation parameters}`
`timing`	`{ccs clear delay delay_variation ecsm ecsm_variation time_const preset three_state three_state_enable three_state_disable tristate max_trans trans trans_variation ccs_retain retain retain_trans ccs_delay ccs_trans}`
`trans`	`{max_trans trans trans_variation}`
`voltage`	`{hyper noise ccsn_prop ccsn_vout}`

`-unmatched` Reports instances where entries in the reference library do not have equivalents in the comparison library.

`-upscale` When the data in two libraries for a particular arc has different data dimensions (for example: 7x1 versus 7x7), the data from the smaller table is scaled up to match the data size in the larger table.

`-verbose` Generates a report showing every comparison, including those that did not exceed a tolerance. The output is written to the `-report` *`filename`*, default <*`comp_lib`*>`.cmp.txt`. An overall comparison summary is also written to the standard output.

<*`ref_lib`*> (Required positional variable) Reference library.

<*`comp_lib`*> (Required positional variable) Comparison library.

The `compare_library` command compares the data found in the reference library (`ref_lib`) with the matching data found in the compare library (`comp_lib`) and reports the differences that exceed the defined tolerances. The report includes the comparison of attributes, capacitance, leakage, delay, transition, power, timing constraints, and comparison of advanced model data such as ECSM, CCS, Electromigration (EM), Liberty Variation Format (LVF), and Normalized Driver Waveform (NDW). For CCS, the current waveforms are converted to voltage waveforms and the comparisons performed using delay and slew thresholds rather than for each current measurement. If the table indices in the comparison library are different from the reference library, bi-linear interpolation will be used prior to performing the comparison. For CCSN, the following data types are supported: `ccsn_dc`, `ccsn_vout` and `miller_cap` (propagation tables are not yet implemented). For `ccsn_dc` and `ecsm`, five points of the DC current data are compared: the first point, the last point and three intermediate points.

When comparing libraries, the data entries must have equivalent conditions. Two entries are deemed equivalent if they have the same or overlapping logic conditions, related pins and data type. If you are comparing libraries with different cell names, use the `-define_map` command to map the names in the comparison library to the reference library. Note that all the pin names must match.

When comparing two libraries that have different index values, slew thresholds, and units, the values in the *`comp_lib`* are scaled accordingly before comparison. The following characters are used to indicate that some form of data manipulation has occurred before the comparison:

*	Data were scaled due to slew thresholds or units.
^	Input slews were interpolated.
~	Output loads were interpolated.
!	The indices were switched.
+	Both the *`ref_lib`* and *`comp_lib`* values were padded.

When comparing libraries that have different `when` conditions, the data groups that have overlapping conditions are compared. If the number of indices (dimensions) differs between two data groups, the data in the smaller dimension table is expanded to fit the larger dimension table. For example, if comparing delay data based only on input slew versus delay data based on slew and load, the 1-D slew table is expanded to a 2-D slew/load table by using the first value of the load indices from the 2-D table.

Examples

```
# Set all relative tolerances to 2%, constraint tolerance
# to 3%, Set absolute tolerance
# values for constraint, transition

compare_library \
    -reltol { all 0.02  constraint 0.03} \
    -abstol { constraint 5e-12 trans 5.0e-12 } \
    ref.lib comp.lib

# Compare only nominal data
compare_library -skip parameters ref.lib comp.lib

# Compare CCS data impact for parameter par1
compare_library ccs.nom.lib ccs.par1_P.lib
compare_library ccs.nom.lib ccs.par1_N.lib
```

Sample Output Report

```
*** LIBRARY comparison of comp.lib with ref.lib


Legend : * scaled, ! indices switched, ^ slews interpolated, ~ loads interpolated,
+ half slew padding


*** BEGIN INVX1 COMPARISON ***


INVX1 Delay Comparison in ns
+------+-------------------------+-------------+-------------+-------------+-----
-----+----------+----------+----------+
| Row #|                Pin Name |   Ref Value |  Comp Value |        Diff |   Diff
% |     Type |  Index_1 |  Index_2 |
+------+-------------------------+-------------+-------------+-------------+-----
-----+----------+----------+----------+
|     1|          INVX1:A->ON FR |    0.181790 |    0.171756 |   -0.010034 |   -
5.52% |    delay |    0.304 |    0.058 |
|     2|          INVX1:A->ON FR |    0.239880 |    0.227162 |   -0.012718 |   -
5.30% |    delay |    0.612 |    0.058 |
+------+-------------------------+-------------+-------------+-------------+-----
-----+----------+----------+----------+
|     3|          INVX1:A->ON RF |    0.149020 |    0.138183 |   -0.010837 |   -
7.27% |    delay |    0.612 |    0.058 |
+------+-------------------------+-------------+-------------+-------------+-----
-----+----------+----------+----------+


INVX1 Delay SUMMARY
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      Data Type |  Entries |     Avg Diff |    Avg Diff% |       Sigma% |     Max
Diff |    Max Diff% | Outliers |
```

```
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      delay(ns) |       98 |     -0.00166 |       -2.30% |        4.27% |     -
0.01272 |       -7.27% |        3 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
Worst delay outlier: Max Abs:     -0.01272, Row # :       2; Max Rel:      -7.27%,
Row # :        3


INVX1 Transition Comparison in ns
+------+-------------------------+-------------+-------------+-------------+-----
-----+----------+----------+----------+
| Row #|                Pin Name |   Ref Value |  Comp Value |        Diff |   Diff
% |     Type |  Index_1 |  Index_2 |
+------+-------------------------+-------------+-------------+-------------+-----
-----+----------+----------+----------+
|     1|          INVX1:A->ON FR |    0.219420 |    0.201528 |   -0.017892 |   -
8.15% |   rising |    0.004 |    0.058 |
|     2|          INVX1:A->ON FR |    0.219220 |    0.201360 |   -0.017860 |   -
8.15% |   rising |    0.013 |    0.058 |
|     3|          INVX1:A->ON FR |    0.219550 |    0.201632 |   -0.017918 |   -
8.16% |   rising |    0.032 |    0.058 |
|     4|          INVX1:A->ON FR |    0.219330 |    0.201455 |   -0.017875 |   -
8.15% |   rising |    0.072 |    0.058 |
|     5|          INVX1:A->ON FR |    0.219460 |    0.202950 |   -0.016510 |   -
7.52% |   rising |    0.148 |    0.058 |
|     6|          INVX1:A->ON FR |    0.238460 |    0.225337 |   -0.013123 |   -
5.50% |   rising |    0.304 |    0.058 |
|     7|          INVX1:A->ON FR |    0.316770 |    0.301474 |   -0.015296 |   -
4.83% |   rising |    0.612 |    0.058 |
+------+-------------------------+-------------+-------------+-------------+-----
-----+----------+----------+----------+


INVX1 Transition SUMMARY
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      Data Type |  Entries |     Avg Diff |    Avg Diff% |       Sigma% |     Max
Diff |    Max Diff% | Outliers |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      trans(ns) |       98 |     -0.00242 |       -2.18% |        2.94% |     -
0.01792 |       -8.16% |        7 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
Worst trans outlier: Max Abs:     -0.01792, Row # :       3; Max Rel:      -8.16%,
Row # :        3
```

```
*** END INVX1 COMPARISON ***


Overall LIBRARY SUMMARY
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      Data Type |  Entries |     Avg Diff |    Avg Diff% |       Sigma% |     Max
Diff |    Max Diff% | Outliers |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|    leakage(nW) |        2 |      0.00000 |       0.00% |       0.00% |     0.00000
|         0.00% |         0 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+


+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      Data Type |  Entries |     Avg Diff |    Avg Diff% |       Sigma% |     Max
Diff |    Max Diff% | Outliers |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|       cap(pf) |        2 |      0.00000 |       0.00% |       0.00% |     0.00000
|         0.00% |         0 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+


+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      Data Type |  Entries |     Avg Diff |    Avg Diff% |       Sigma% |     Max
Diff |    Max Diff% | Outliers |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      delay(ns) |       98 |     -0.00166 |       -2.30% |        4.27% |     -
0.01272 |       -7.27% |        3 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+



Worst delay outlier (one per cell):
+-----+------------------+--------------+----------+------------------+----------
----+----------+
|   # |             Cell |     Max Diff |    Row # |             Cell |    Max Diff%
|    Row # |
+-----+------------------+--------------+----------+------------------+----------
----+----------+
|   1 |            INVX1 |     -0.01272 |        2 |            INVX1 |       -
7.27% |        3 |
```

```
+-----+------------------+-------------+---------+-----------------+---------
----+----------+

+----------------+---------+-------------+-------------+-------------+-------
-------+-------------+----------+
|      Data Type |  Entries |     Avg Diff |    Avg Diff% |       Sigma% |     Max
Diff |    Max Diff% | Outliers |
+----------------+---------+-------------+-------------+-------------+-------
-------+-------------+----------+
|      trans(ns) |       98 |     -0.00242 |       -2.18% |        2.94% |     -
0.01792 |       -8.16% |        7 |
+----------------+---------+-------------+-------------+-------------+-------
-------+-------------+----------+

Worst trans outlier (one per cell):
+-----+------------------+-------------+---------+-----------------+---------
----+----------+
|   # |            Cell |     Max Diff |    Row # |            Cell |    Max Diff%
|    Row # |
+-----+------------------+-------------+---------+-----------------+---------
----+----------+
|   1 |             INVX1 |     -0.01792 |        3 |             INVX1 |       -
8.16% |        3 |
+-----+------------------+-------------+---------+-----------------+---------
----+----------+



+----------------+---------+-------------+-------------+-------------+-------
-------+-------------+----------+
|      Data Type |  Entries |     Avg Diff |    Avg Diff% |       Sigma% |     Max
Diff |    Max Diff% | Outliers |
+----------------+---------+-------------+-------------+-------------+-------
-------+-------------+----------+
|     power(pJ) |      98 |     0.00003 |       1.99% |       6.54% |     0.00000
|        0.00% |        0 |
+----------------+---------+-------------+-------------+-------------+-------
-------+-------------+----------+


+----------+---------------+---------------+----------+
|  Entries |     Avg Diff% |        Sigma% | Outliers |
+----------+---------------+---------------+----------+
|      298 |        -0.82% |         5.18% |       10 |
+----------+---------------+---------------+----------+

Sample Summary
Writing library comparison report to comp.txt
```

```
Comparing cell INVD1


Overall LIBRARY SUMMARY
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      Data Type |  Entries |     Avg Diff |    Avg Diff% |       Sigma% |     Max
Diff |    Max Diff% | Outliers |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|    leakage(nW) |        2 |      0.00000 |       0.00% |       0.00% |      0.00000
|         0.00% |         0 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|        cap(pf) |        2 |      0.00000 |       0.00% |       0.00% |      0.00000
|         0.00% |         0 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|       delay(ns) |       98 |      -0.00166 |       -2.30% |        4.27% |     -
0.01272 |       -7.27% |         3 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|       trans(ns) |       98 |      -0.00242 |       -2.18% |        2.94% |     -
0.01792 |       -8.16% |         7 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
| constraint(ns) |        0 |      0.00000 |       0.00% |       0.00% |      0.00000
|         0.00% |         0 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
|      power(pJ) |       98 |      0.00003 |       1.99% |       6.54% |      0.00000
|         0.00% |         0 |
+----------------+----------+--------------+--------------+--------------+-------
-------+--------------+----------+
+----------+---------------+---------------+----------+
|  Entries |     Avg Diff% |        Sigma% | Outliers |
+----------+---------------+---------------+----------+
|      298 |        -0.82% |         5.18% |       10 |
+----------+---------------+---------------+----------+


LIBRARY comparison report written to comp.txt
```

create_aocv_table

Performs characterization and generation of advanced on-chip variation (AOCV) derate tables for a specific existing library.

Options

`-cells {`*`cell_names`*`}`

Specifies a list of cells to use for generating the AOCV derate table. Default: use all cells

`-chain_length <`*`number`*`>`

Specifies the number of instances in the chain. Default: `1`

`-chain_output_list {`*`list`*`}`

Specifies a list of chain lengths, controlling the depth of derate data that is to be output into the AOCV table. The list should be monotonically increasing. If `-chain_output_list` is specified, but the `-chain_length` option is *not* specified, the `-chain_length` value defaults to the largest value in the `-chain_output_list`. Variety characterizes every value from a length of 1 to n (the largest `-chain_output_list` value). For example, the following controls the chain lengths that go into the AOCV table:

```
-chain_output_list { 1 5 10 20 30 40 50 60 }
```

This defaults to every chain length from `1` to the `-chain_length` value.

`-dir <`*`directory`*`>` Specifies the directory to store all the files that are created. Default: `AOCV`

`-exclude` Reverses the meaning of the `-cells` list, so that the specified list of cells are excluded from characterization.

`-extsim <hspice | ski | spectre>`

Specifies an external simulator to use. Default: `ski`

`hspice`	Uses the HSPICE simulator.
`ski`	Uses Spectre simulator with SKI flow.
`spectre`	Uses the Spectre simulator.

`-extsim_format <spice | spectre>`

Specifies the format of the netlist. Default: `spectre` (However, when `-extsim` is set to `hspice`, the default format is `spice`.)

`spice`	Uses the SPICE format.
`spectre`	Uses the Spectre format.

`-fanout <value>` Specifies the number of fanout cells to be added to the output of each cell in the simulation chain. Default: `0`

The cell used for fanout is either the current cell in the chain or the cell defined by the `-fanout_cell` option.

`-fanout_cell <cell_name>`

Specifies a cell to be added to the output of each cell in the simulation chain. Default: current cell

The specified cell must have only one input and one output.

`-fanout_type <cell | cin>`

Specifies whether an instantiated cell or an equivalent input pin capacitance is to be used for fanout. Default: `cell`

`cell`	Uses a cell for fanout.
`cin`	Uses an input pin capacitance for fanout.

`-format <cadence | synopsys>`

Specifies the output format. Default: `cadence`

`cadence`	Requests output in Cadence format.
`synopsys`	Requests output in Synopsys format.

`-lib <lib_name>` Specifies the Liberty library to use.

`-load <value | min | mid | max | half_max>`

Specifies a load value in library units, or with specifiers. Default: `mid`

value	Specifies a load value in library units.
`min`	Selects the minimum load from the load index.
`mid`	Selects the middle load from the load index.
`max`	Selects the maximum load from the load index.
`half_max`	Uses half of the maximum load from the load index.

`-load_index <index_point>`

Specifies an index point from the library. If `-load_index` is used, this overrides any `-load` specification. Default: `-1` (Use the `-load` specification.)

The index values start at `0`.

`-model <filename>` Specifies the SPICE model filename to be included in the SPICE netlist as `.inc <model>`.

`-nominal` In cases where mixed models are used, specifies that the run is to determine nominal delay values.

`-nominal_dir <dir_name>`

Specifies the directory where nominal delay information is stored.

`-populate_start <num>`

Sets the number of stages to simulate. Default: `5`

`-select_timing <min | max>`

Specifies the timing arc to use for run-time improvement. Default: ""

To guarantee the worst-case variation, Variety characterizes all arcs in a cell for variation effects. This can cause run-time issues for cells with multiple inputs affecting each output. Set this option to enable Variety to select the worst or best arc for each output from the nominal library to characterize for variation affects. This can significantly improve the variation characterization run time.

`-sequential_slew_from_receiver`

Specifies that the slew for the sequential cell AOCV is to be taken from the *receiver* cell. Default: the slew is taken from the *sequential* cell.

`-start_from <ldb | model_nom>`

Instructs Variety to use the LDB and generate a new AOCV table without rerunning characterization

ldb	Instructs Variety to use the LDB.
model_nom	Instructs Variety to restart from the characterized nominal model data.

`-slew <value | min | mid | max | half_max>`

Specifies a slew value in library units, or with specifiers. Default: `mid`

value	Specifies a slew value in library units.
`min`	Selects the minimum slew from the load index in the library.
`mid`	Selects the middle slew from the load index in the library.
`max`	Selects the maximum slew from the load index in the library.
`half-max`	Uses half of the maximum slew from the load index in the library.

`-slew_factor <`*value*`>`

Specifies a multiplier that determines the final slew by multiplying the `-slew` value or the slew from the `-slew_index`. Default: `1.0`

That is, the final slew = (`-slew_factor`) x (`-slew`) or (`-slew_factor`) x (slew pointed to by `-slew_index`).

`-slew_index <`*index_point*`>`

Specifies an index point from the library to be used for AOCV. If `-slew_index` is used, it overrides the specification of a `-slew`. Default: `-1` (Use `-slew` instead of `-slew_index`.)

Note: The index starts at `0`.

`-subckts {<`*file_name*`>}`

(Required) Specifies the list of files containing the SPICE subckts for each cell to be compared.

`-thread <`*number*`>` Specifies the number of different CPU threads to use. Default: use all available threads

`-wire_cap <`*value* `| min | mid | max`**>**

Sets the wire capacitance between cells to the min, mid, or max load from the delay table for the arc under test, or to a specific value (in Farads). Default: `min`

value	Sets the capacitance to the specified value.
`min`	Sets the capacitance to the min load from the delay table.

`mid`	Sets the capacitance to the mid load from the delay table.
`max`	Sets the capacitance to the max load from the delay table.

`-wire_cap_index <index_point>`

Specifies an index load value to use as a wire capacitance. If the specified index is not in the load range, uses the `-wire_cap` value instead. Default: `-1` (Use the value specified by `-wire_cap`.)

Note: The index starts at `0` (zero).

`-wire_cap_subtract_next_input`

Subtracts the next input capacitance from the `-wire_cap` value and uses that as a net wire cap.

`-wire_res <value>` Specifies the wire resistance in ohms for each side of a Pi network between cells. Default: `0.01` ohms

`<aocv_table_file>` (Required positional option) Specifies the filename to be used for the created AOCV table.

The circuit used for characterization can be a single cell instance or a series of cell instances of the length specified by `-chain_length`. Between each cell instance is a T network consisting of two resistors (with resistance specified by `-wire_res`) with an intermediate capacitance (specified by `-wire_cap`).

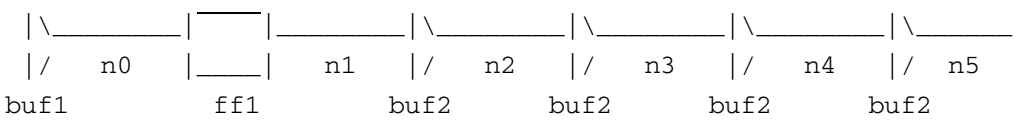
The output AOCV table file contains entries for each characterized cell that include the effects of random local variations. Each entry includes the derate factors for cell_rise and cell_fall as a function of the chain length of the cell's timing path.

It is also possible to generate AOCV derate tables for existing libraries that include SPICE global model variations relative to nominal. To use this feature, first generate a nominal derate table that does not include any global variations. Then use this nominal table to define the AOCV derate tables for random local variations. This two-step process is performed as follows:

```
create_aocv_table -nominal -dir "/path/nominal_results"
create_aocv_table -nominal_dir "/path/nominal_results" -dir "/path/aocv_tables"
```

The `-populate_start` option specifies the number of stages to simulate. All stages after the last simulated stage are populated mathematically. For example, in case of sequential cells and level shifters (all cells where it does not make sense to place them in series), the AOCV characterization circuit looks like:

```
aocv_set_driver_receiver \
    -extra_driver {buf1 X Y} \
    -driver    {ff1 CP Q} \
    -receiver   {buf2 A B}
```



Where,

`buf1` is an extra driver cell

`ff1` is the driver cell

`buf2` is the receiver cell

In this case, the first characterized stage is measured from the input to the driver cell (`n0`) to the input to the first receiver cell. If asked to characterize `5` stages (the default), the second through the fifth stages are built using the receiver cells. Each stage (up to the specified `populate_start` stage) is characterized from the input to the driver cell to the input of the receiver cell.

The five characterized stages are:

```
n0 to n1
n0 to n2
n0 to n3
n0 to n4
n0 to n5
```

All stages after `n5` will be populated automatically.

Examples

Example 1:

```
create_aocv_table -model $model_path -chain_length 8 -subckts $subckts \
    -lib "$test_path/inv.lib" aocv.tbl
```

Example 2:

For an existing library with global variations:

```
# FIRST RUN
create_aocv_table -cells $cells -model $model_path \
    -chain_length 12 -subckts $subckts -lib "$test_path/inv.lib" \
    -dir "nominal_results" ${cell}_aocv.tbl
```

```
# SECOND RUN
create_aocv_table -cells $cells -model $model_path \
    -chain_length 12 -subckts $subckts -lib "$test_path/inv.lib" \
    -nominal_dir "nominal_results" ${cell}_aocv.tbl
```

define_arc

Specifies a user-defined arc to override the Variety automatic arc determination. An arc represents library data between a given pin and a related pin. Typically, this command is only required for the characterization of I/O cells or very complex cells.

Options

`-constraint <"`*`function`*`">`

Specifies the logic condition applied to vectors but does not place the actual states in the library.

`-delay_threshold {<`*`in_rise`*`> <`*`in_fall`*`> <`*`out_rise`*`> <`*`out_fall`*`>}`

Defines a list of delay percentage measurement points (a ratio of VDD normalized to between 0 and 1) for the arc. Each option consists of a list of four values representing the input_rise_delay threshold, the input_fall_delay threshold, the output_rise_delay threshold, and the output_fall_delay threshold in that exact order. If not specified, then all delays are measured at the values defined by the `delay_inp_rise`, `delay_inp_fall`, `delay_out_rise`, and `delay_out_fall` variables.

`-dependent_load <`*`value`*`>`

Specifies the load to add to dependent side pins. Use this option to control the load applied to side outputs that impact the arc. Dependent loads specified with the `-dependent_load` option take precedence over those specified by the `set_dependent_load` command.

`-dual_dir <U | D | B>`

Specifies the switching direction of dual pin used to set load direction. The `-dual_dir` option is the equivalent of the `-load_dir` option as it defines the direction of the load circuitry to apply to the `-dual_pin` of this `define_arc` command.

	`U`	Sets a direction of up.
	`D`	Sets a direction of down.
	`B`	Sets a direction of both.

`-dual_pin <`*`name`*`>` Specifies the other pin in a pair of differential *output* pins.

`-extsim_deck_header` Allows to provide external simulator commands directly to the external simulator on an individual arc basis without needing Liberate to process or review them. This option is intended to be used when an external simulator is used (refer to the `-extsim` option of the `char_variation` command). It is a local arc-specific version of the `extsim_deck_header` variable. As Liberate does not parse the string specified by this option, ensure that the contents are valid and consistent with the arc simulation. The value string can contain the return character ("\n"). The value string is included in the top of simulation deck. For example:

```
define_arc -extsim_deck_header ".ic n128 0"
   -related_pin ck -pin Q ...
```

`-ignore` A flag that prevents simulation of all arcs originating from the related pin and ending at the pin. When this option is used, only the `-pin` and `-related_pin` options are required. All other options, including the `-vector` option, are ignored.

`-load_dir <U | D | B>`

Specifies whether the pullup resistance, the pulldown resistance, or both resistances should be applied to this arc.

`U`	Applies the pullup resistance.
`D`	Applies the pulldown resistance.
`B`	Applies both the pullup and pulldown resistances.

`-metric <delay | glitch>`

Specifies the metric to use for measuring timing constraints.

`delay`	Produces a violation when a delay change at the probed pin exceeds the `constraint_delay_degrade` variable.
`glitch`	Produces a violation when the glitch-peak at the probed pin exceeds the `constraint_glitch_peak` variable.

`-pin {`*`pins`*`}` (Required) Specifies a list of destination pins for the arc (typically output pins for combinatorial arcs, input pins for timing constraint or hidden power arcs).

`-pinlist {`*`list_of_pins`*`}`

List of pins corresponding to a vector.

`-pin_dir <R | F>` Specifies the transition direction of pins.

`R` Specifies a rising transition.

`F` Specifies a falling transition.

`-pin_load <`*`value`*`>` Specifies additional circuitry to be applied to all the destination pins of the `define_arc` command. The `-pin_load` *`value`* option refers to a template that defines the loading circuitry to be placed prior to the loading capacitance for the pin. The loading template must be pre-defined using the `define_pin_load` command. The additional circuitry can include pullup and pulldown resistances and series resistance.

`-probe <{`*`names`*`} | altos_internal>`

Used for timing constraints and defines the nodes to monitor when determining the constraint. It can be an external pin such as the `Q` pin in a flip-flop or an internal node name. Default: the pin defined by the `constraint_output_pin` variable is probed

`names` Specifies a list of names of nodes to monitor for constraints in sequential cells.

`altos_internal`

Instructs Variety to use the internal probe node when a constraint can be measured at both an internal node and an output pin.

`-related_pin {`*`pins`*`}` Specifies a list of related pins (typically input pins for combinatorial arcs, clock pins for timing constraint arcs) while the `-pin` option is a list of destination pins for the arc.

`-related_pin_dir <R | F>`

Transition direction of related pins.

`R` Specifies a rising transition.

`F` Specifies a falling transition.

`-slew_threshold {`*`values`*`}`

Specifies a list of slew percentage measurement points (a ratio of VDD normalized to between 0 and 1) for the arc. Each option consists of a list of four values. For `-slew_threshold` the values in the list represent the lower_rise_slew measurement threshold, the upper_rise_slew measurement threshold, the lower_fall_slew measurement threshold, and the upper_fall_slew measurement threshold in that exact order. If not specified, then all slews are measured at the values defined by the `measure_slew_lower_rise`, `measure_slew_upper_rise`, `measure_slew_lower_fall`, and `measure_slew_upper_fall` variables.

`-type <async |combinational | disable| edge | enable | hidden | hold | recovery | removal | setup>`

Specifies the type of arc. Default: `combinational`

`async` An `async` arc corresponds to a preset or clear transition.

`combinational`

The arc is a `combinational` path from input pins (`related_pin`) to output pins (`pin`) for combinational cells.

`disable` The `disable` type is used for specifying arcs that disable tri-state gates.

`edge` An `edge` arc between an input and an output pin is an edge-triggered transition.

`enable` The `enable` type is used for specifying arcs that enable tri-state gates.

`hidden` A `hidden` arc is an arc that does not cause an output transition and is used to simulate hidden power for that `pin`.

`hold` The arc is a timing constraint of type `hold` between data (`pin`) and a clock (`related_pin`) for sequential cells.

`recovery` The arc is a timing constraint of type `recovery` between data (`pin`) and a clock (`related_pin`) for sequential cells.

	`removal`	The arc is a timing constraint of type `removal` between data (`pin`) and a clock (`related_pin`) for sequential cells.
	`setup`	The arc is a timing constraint of type `setup` between data (`pin`) and a clock (`related_pin`) for sequential cells.
`-value <`*`value`*`>`	Use this to override the characterization and force a value for all entries into the data table for the specified arc. Default: use characterized values	
`-vector <"`*`stimulus`*`">`	Specifies the stimulus to simulate this arc. It is defined as a string of bits (digits) where each bit corresponds to one string in the pinlist. Each bit can have the values `R` (rising), `F` (falling), `X` (don't care), `1` (logic high), `0` (logic low). The order of the bits must correspond one-to-one to the order of pinlist defined by using the `-pinlist` option of the `define_cell` or `define_arc` command. Blank spaces are permitted in the vector for readability. The vector value for a pin must be logically consistent with the `when` and `constraint` options; else, the `define_arc` command is rejected. If a side input is specified as X, it is overridden by the state of the pin as specified in the `when` or `constraint` option. If there are busses in the pinlist, there should be one bit in the vector for each bus. The bit value is applied to all elements in a bus. If different logical values are required for each bit in the bus, the bus bits must be separately enumerated in the pinlist.	
`-when <"`*`function`*`">`	Specifies the logic conditions of the other pins of the cell to enable this arc using the Liberty `when` syntax. It corresponds to the Liberty `when` attribute.	
`{`*`cell_names`*`}`	(Required positional option) List of cells.	

The `define_arc` command can be applied to a single cell or a list of cell names. The template used for each arc defaults to the template defined for the cell unless a `define_index` command is specified for that particular arc.

Examples

```
# Define the IOCELL

define_cell \
-input {IN OEN } \
```

```
      -output {OUT} \
      -bidi {PAD} \
      -pinlist {IN OEN PAD OUT} \
      -delay delay_template_3x3 \
      -power power_template_3x3 \
      IOCELL

define_arc \
       -vector {XXRR} \
       -related_pin PAD \
       -pin OUT \
       IOCELL

define_arc \
       -vector {XXFF} \
       -related_pin PAD \
       -pin OUT \
       IOCELL

# Define additional loading for the PAD pin
define_pin_load \
        -pullup_voltage 3.3 \
        -pullup_resistance 1000 \
        -pulldown_resistance 1000 \
        -series_resistance 50 \
        load_template

define_arc \
       -vector {R0RX} \
       -pin_load load_template \
       -related_pin IN \
       -pin PAD \
       IOCELL

define_arc \
       -vector {F0FX} \
       -pin_load load_template \
       -related_pin IN \
       -pin PAD \
       IOCELL

define_arc \
       -type enable \
       -vector {1FRX} \
       -pin_load load_template \
       -related_pin OEN \
       -pin PAD \
       IOCELL

define_arc \
       -type enable \
       -vector {0FFX} \
       -pin_load load_template \
       -related_pin OEN \
       -pin PAD \
       IOCELL

define_arc \
       -type disable \
       -vector {0RRX} \
       -pin_load load_template \
       -related_pin OEN \
       -pin PAD \
       IOCELL
```

```
define_arc \
       -type disable \
       -vector {1RFX} \
       -pin_load load_template \
       -related_pin OEN \
       -pin PAD \
       IOCELL
define_arc \
    -pinlist { A B C[5:0] OUT } \
    -vector { R 0 1 F } \
    -related_pin A \
    -pin OUT \
    myCell
define_arc \
    -pinlist { A B C[5] C[4] C[3] C[2] C[1] C[0] OUT } \
    -vector { R 0 101110 F } \
    -related_pin A \
    -pin OUT \
    myCell
```

define_cell

Defines how a cell is to be characterized. Each cell can have a unique `define_cell` command or a `define_cell` command can be shared among a group of cells.

Options

`-async {`*`pin_names`*`}`	Specifies that the listed pins are asynchronous.
`-bidi {`*`pin_names`*`}`	Specifies that the listed pins are bi-directional.
`-clock {`*`pin_names`*`}`	Specifies that the listed pins are clocks.
`-constraint <`*`name`*`>`	Specifies a template, pre-defined using the `define_template` command, that characterizes timing constraints (setup, hold, recover, removal). The template defines the range of input slews to use for the data and clock signals.

`-delay <name>`

Specifies a template for delay tables, pre-defined using the `define_template` command, that enables timing models and the characterization of cell delay and output slew for the non-linear delay model (NLDM). The template defines the range of input slews and output loads to use.

Examples:

```
define_cell -input {A1 A2} -output {Z} -delay delay_3x3 \
{NAND2X4 NOR2X2}
```

```
define_cell -input {D} -output {Q QN} \
    -clock {CK} -async {SN} -delay delay_5x5 \
    -constraint constraint_3x3 \
{DFFX1}
```

`-ignore_input_for_autocap {pin_names}`

Specifies a list of input/bidi pins. Any timing arcs that originate at pins specified in the list are not considered for `-auto_index` and `-auto_max_capacitance` calculations.

`-ignore_output_for_autocap {pin_names}`

Specifies a list of input/bidi pins. This option accepts a list of bidi/output pins or an asterisk (`*`) to match all output pins. Any timing arcs that terminate at pins specified in this list are not considered for `-auto_index` and `-auto_max_capacitance` calculations.

`-input {pin_names}` Specifies that the listed pins are inputs.

`-output {pin_names}`

Specifies that the listed pins are outputs.

`-pinlist {pin_names}`

Specifies the pin-order list. This information is used by the `-vector` option of the `define_arc` command when specifying a user-defined timing arc. The pin list can contain internal pins as well as input, inout, and output pins.

`-scan {pin_names}` Specifies the names of scan-related pins that are to be removed. The `-scan`, `-scan_cell_postfix`, `-scan_cell_prefix`, `-scan_disable`, and `-scan_scale_power_factor` options are required when a dummy cell (one with the scan pins removed) is required.

`-scan_cell_postfix "`*`string`*`"`

Specifies a string to attach to the end of the dummy scan cell name when the scan pins are removed. If this option is specified, the library contains both the original cell and the scan dummy cell with the modified name.

`-scan_cell_prefix "`*`string`*`"`

Specifies a string to attach to the beginning of the dummy scan cell name when the scan pins are removed. If this option is specified, the library contains both the original cell and the scan dummy cell with the modified name.

`-scan_disable {<`*`pin value`*`>}`

Specifies a list of pin-value pairs that disables scan mode. If the application of the -`scan_disable` option results in two or more data groups with the same states, then the group data is merged according to the rules specified by the `default_timing` variable. (If the default group is turned off, then the group data is merged using the default setting of `max`.)

`-scan_scale_power_factor <`*`factor`*`>`

Specifies the scale applied to power in the scan dummy cell. Default: number_of_total_pins / number_of_non-scan_pins

`-when <"`*`function`*`">` Specifies user-defined cell level logic constraints using the Liberty format `when` syntax, constraining the tool's automatic vector generation for this cell.

`{`*`cell_names`*`}` (Required positional option) Specifies the list of cells to be characterized.

All pins of a cell must have a defined pin type. If a pin name or pin type does not apply to a particular cell it is ignored. For example, combinatorial cells such as NOR or NAND gates may not have clock or async pins so any definition for these pins is ignored. Likewise, if a pin name is specified but not used by a particular cell it is ignored by that cell. The same pin name cannot appear in multiple pin types within a single `define_cell` command. For example, if one cell has an input Y and another has an output Y then they must be defined uniquely with separate `define_cell` commands.

The `-constraint` and `-delay` options define which template to use for characterizing each library construct. If a template is specified, the appropriate construct is characterized for the given set of cells. If a template is omitted, this construct is not characterized.

define_index

Overrides the indices specified in the templates referenced by `define_cell`, or created using the `-auto_index` option of `char_variation`, for all the arcs between the `-related_pin` list and the `-pin` list for the given `-type`.

Options

`-index_1 {`*`indices`*`}` Specifies the indices to use as index_1. At least one of `-index_1` or `-index_2` must be specified.

`-index_2 {`*`indices`*`}` Specifies the indices to use as index_2. At least one of `-index_1` or `-index_2` must be specified.

`-pin {`*`pins`*`}` (Required) Specifies a list of pin names.

`-related_pin {`*`pins`*`}`

(Required) Specifies a list of related pin names.

`-type {constraint | delay}`

Specifies a list of data types.

`constraint`	Specifies that constraint data can be overridden.
`delay`	Specifies that delay data can be overridden.

`{`*`cell_names`*`}` (Required positional option) Specifies the list of cells to which the overrides are applied.

The size of the `index_1` and `index_2` lists must be equal to the equivalent template-type specified by `define_cell`. Multiple `define_index` commands can be used to specify different overrides for different arcs for the same set of cells, or for different cells.

Examples

```
define_template -type delay \
-index_1 {0.025 0.1 0.25} \
-index_2 {0.0010 0.015 0.100} delay_3x3

define_cell \
    -input {A1 A2} -output {Z} \
    -delay delay_3x3 {NAND2X4 NOR2X2}

# Define different output loads for A1 to Z arcs
define_index \
    -pin {Z} -related_pin {A1} \
    -type delay \
```

```
    -index_2 {0.0010 0.050 0.500} \
    {NAND2X4 NOR2X2}
```

define_input_waveform

Specifies a piece-wise linear waveform to drive the input during characterization.

Options

`-direction <rise | fall>`

(Required) Specifies the logical direction of a given input waveform.

All input transition values must be specified as PWL for any given arc. Do not specify only one index value as PWL and have the others default to another alternative input waveform method.

`rise`	Specifies a logical direction of *rise*.
`fall`	Specifies a logical direction of *fall*.

`-gnd_val <voltage>` (Required) Specifies the ground voltage value of the associated input pins in volts.

`-pinlist {<cell pin>}`

Specifies a list of cell-pin pairs to which the waveforms are applied.

The cells and pins can be specified using wildcards (`*`). User inputs for a cell-pin pair are searched for in the following order: cell:pin, cell:*, *:pin, and *:*.

`-pwl {<time voltage>}`

(Required) Specifies a list of time-voltage pairs in MKS units.

`-slew_index <slew_index>`

(Required) Specifies the slew from the index (in LDB/lib units) with which the PWL waveform is linked.

`-vdd_val <voltage>` (Required) Specifies the input voltage full rail swing for the associated input pins in volts.

This command must be specified before the `char_variation` command is run.

Example

```
define_input_waveform \
    -direction fall \
    -pwl  {  0.0 3.0 6.875e-11 0.0} \
    -vdd_val 3.0 \
    -gnd_val 0.0 \
    -slew_index 5.5e-2 \
    -pinlist { DFF1  D }
define_input_waveform \
    -direction rise \
    -pwl  {  0.0 0.0 6.875e-11  3.0 } \
    -vdd_val 3.0 \
    -gnd_val 0.0 \
    -slew_index 5.5e-2 \
    -pinlist { DFF1 D }
```

define_leafcell

Defines the level of hierarchy that resides at the bottom of a cell level netlist.

Options

`-area <"`*`string`*`">`	Specifies the name of the diode area parameter in the cell. Default: `'area'`

`-extsim_model`

Use this option to inform the simulator that some of the model files for the leafcells are loaded by `extsim_model_include` and `extsim_deck_header` variables, rather than by the `read_spice` command.

Using this option allows for partial include and partial use of the `read_spice` command. If this option is used, the leafcell being defined also needs to have the `extsim_deck_header` variable insert a `.inc '<path>/modelfile.inc'` to load a model (probably a Verilog model) for this cell.

If a leafcell *does not have* the `-extsim_model` option and the `extsim_model_include` variable is missing, the tool outputs an error requesting use of the `extsim_model_include` variable and quits.

If a leafcell *does have* the `-extsim_model` option, you can load model files for it by using either:

- The `extsim_model_include` variable.
- The `extsim_deck_header` variable.

All other device models can be loaded by using the `read_spice` command.

`-length <"string">`

Specifies the name of the mos length parameter in the cell. Default: `'l'`

`-multiple <"string">`

Specifies the name of the multiple mos parameter. Default: `'m'`

`-pin_position {list_of_pin_positions}`

(Required) Maps the pin positions in this device to the nodes in the model, specifying one number for each pin in the cell.

The first pin is designated by `0`, where `0` is associated with drain, `1` with gate, `2` with source, and `3` with bulk.

For example,

```
define_leafcell -type nmos -pin_position {0 1 2 3} nch
```

`-pj <"string">`

Provides the name of the pj diode parameter in the cell. Default: `'pj'`

`-scale <"value">` Provide the MOS parameter scale factor in the cell. Default: `1.0`

This scale factor is used only by the *Inside View* of Variety to determine device sizes, and is not applied to the device sizes in the simulation netlist.

`-type <nmos | pmos | diode | r | c | nmos_stk | pmos_stk>`

(Required) Specifies the type of the cell.

`nmos`	Specifies the cell as an NMOS semiconductor.
`pmos`	Specifies the cell as a PMOS semiconductor.
`diode`	Specifies the cell as a diode.
`r`	Specifies the cell as a resistor.
`c`	Specifies the cell as a capacitor.
`nmos_stk`	Specifies the cell as an nmos stack. This type supports 5 pin stacked NMOS transistors. For 7 pin stacked MOS, the extra 2 pins are internal pins. Note that the pin position for stacked MOS is: d g1 g2 s b.
`pmos_stk`	Specifies the cell as a pmos stack. This type supports 5 pin stacked PMOS transistors. For 7 pin stacked MOS, the extra 2 pins are internal pins. Note that the pin position for stacked MOS is: d g1 g2 s b.

`-width <"string">` Provides the name of the MOS width parameter in the cell. Default: `'w'`.

`{cell_names}` (Required positional option) Specifies the list of leafcell names.

Using the `define_leafcell` command allows the tool to correctly identify devices in the cell netlist even when the process model file cannot be parsed. This command can be used in combination with the `extsim_model_include` variable to enable external simulation with the process models and the compiled netlist. This command supports identification of mosfets, diodes, resistors, and capacitors.

Examples

```
# Define the cell NCH_MAC as a leafcell
define_leafcell \
```

```
    -type pmos \
    -pin_position { 0 1 2 3 } \
PCH_MAC
# Define the cell PCH_map as a leafcell.
# first node (gate) in netlist must be swapped with the
# second node (drain) to match drain/gate/source/bulk order
define_leafcell \
    -type pmos \
    -pin_position { 1 0 2 3 } \
    PCH_map
```

define_map

Defines a file for mapping cell names prior to writing out the library.

Options

<map_filename>	(Required positional option) Defines a file that maps the names of cells.

The `define_map` command defines a file for mapping cell names prior to writing out the template, library, Verilog, VITAL, or datasheet files. It can also be used to map cell names when doing a library comparison using the `compare_library` command. It also changes the cell name(s) returned by the following API functions: `ALAPI_inputs`, `ALAPI_outputs`, `ALAPI_inouts`, `ALAPI_internals`, `ALAPI_clocks`, `ALAPI_pinnames`, `ALAPI_name`, `ALAPI_cellnames`, and `ALAPI_cellgroups`.

If the specified file contains only cell name mapping, this command can be used before model generation (see `write_library`, `write_verilog`, and `write_vital`) and before the `write_template` command. However, if the specified file contains pin mapping, the `define_map` command must be specified before the `read_ldb` and `read_library` commands are run.

The specified file should contain separate lines of one of the following formats:

- *<original_cell_name> <new_cell_name>*
- *<original_cell_name:pin_name> <new_pin_name>*

Example

Define a mapping file before writing the library

```
read_ldb      my.ldb.gz
define_map    my_cell.map
```

```
write_library my_mapped.lib
```

The `map_filename` would contain the following information:

```
cell_1     cell1_new
cell_1:ck  CLK
```

Liberate maps the cell named `cell1` to `cell1_new` and the pin named `ck` in `cell_1` to `CLK`.

define_max_capacitance_limit

Sets a pin-specific maximum capacitance.

Options

<value>	(Required positional option) Specifies the maximum allowable capacitance (in Farads).
{*<cell pin>*}	(Required positional option) Specifies a list of cell-pin pairs.

This command has effect only when using the `-auto_index` option to `char_variation`. The values set by `define_max_capacitance_limit` override the calculated max_capacitance when the limit is exceeded. Multiple `define_max_capacitance_limit` commands can be specified. A wildcard (`*`) is supported for the cell name, to allow all cells to be referenced, but you cannot use a wildcard for the pin. Only cells with the given pin name are affected.

Example

```
# Set the max capacitance for pin Y of the cell AND1
define_max_capacitance_limit 100e-15 { AND Y }
```

define_max_transition

Sets a pin-specific maximum transition.

Options

<value>	(Required positional option) Specifies the maximum allowable transition time (in seconds).
{*<cell pin>*}	(Required positional option) Specifies a list of cell-pin pairs.

This command has effect only when using the `-auto_index` option to `char_variation`. The values set by `define_max_transition` override the global value set by the `max_transition` variable. Multiple `define_max_transition` commands can be specified. A wildcard (`*`) is supported to allow all cells to be referenced, but you cannot use a wildcard for the pin. Only cells with the given pin-name are affected.

Example

```
# Set the default maximum transition time
set_var max_transition 1e-9

# Set maximum transition time for some clock pins
define_max_transition 0.5e-9 {DFFX1 CK LTX1 LCK}
define_max_transition 0.75e-9 {GATER CLKIN * CLK}
char_variation -auto_index
```

define_pin_load

Defines additional loading that can be applied to a particular pin prior to the output load.

Options

`-pulldown_resistance <`*`value`*`>`

Specifies the pulldown load resistance.

`-pullup_resistance <`*`value`*`>`

Specifies the pullup load resistance.

`-pullup_voltage <`*`value`*`>`

Specifies the pullup voltage.

`-series_resistance <`*`value`*`>`

Specifies the series load resistance.

`<`*`pin_load_name`*`>` (Required positional option) Specifies the name of the pin load definition.

The loading of the pin can consist of a pullup voltage source via a pullup resistance, a series resistance, and a pulldown resistance. The load definition can be referenced by the `define_arc` command to specify additional loading to be applied to a specific arc. This can be particularly useful for characterizing I/O cells.

Example

```
# Define additional loading for a pin
define_pin_load \
    -pullup_voltage 3.3 \
    -pullup_resistance 4000 \
    -pulldown_resistance 4000 \
    -series_resistance 25 \
    pin_load_template
```

define_template

Defines a template to be used for characterization.

Options

`-index_1 {`*values*`}` Specifies a list of values to be used as the first index. All `-index_1` entries for all the library constructs should be monotonically increasing.

`-index_2 {`*values*`}` Specifies a list of values to be used as the second index. All `-index_2` entries for all the library constructs should be monotonically increasing.

`-index_3 {`*values*`}` Specifies a list of values to be used as the third index. All `-index_3` entries for all the library constructs should be monotonically increasing.

`-type {delay | constraint}`

(Required) Specifies the type of template being defined.

`constraint` The `constraint` template type can be used for timing constraint (setup, hold, removal, recovery) characterization. It requires both `-index_1` and `-index_2` to be specified where `-index_1` represents the range of input slews of the data signal and `-index_2` represents the range of input slews of the reference signal (clock, reset, etc.).

`delay`	The `delay` template type can be used for delay characterization using input slew and output load. It requires both `-index_1` and `-index_2` to be specified where `-index_1` represents the range of input slews and `-index_2` represents the range of output loads.

<template> (Required positional option) Specifies the name of the template being defined.

How each cell is to be characterized is defined by associating the defined template with the appropriate option of the `define_cell` command. Multiple `define_cell` commands can reference a single template.

Examples

```
# Delay template for 3 input slews, 3 output loads
define_template -type delay \
    -index_1 {0.025 0.1 0.25} \
    -index_2 {0.0010 0.015 0.100} delay_3x3
# Timing constraint template for 2 input slews
define_template -type constraint \
    -index_1 {0.025 0.25} \
    -index_2 {0.025 0.25} constraint_2x2
```

define_variation

Defines a set of SPICE model parameters to be varied and by how much.

Options

`-classification <global | local | random>`

Specifies the ecsm_parameter_classification for the s-ecsm format. Default: `local` is assigned to systematic variables and `random` is assigned to random variation parameters in the output library.

`global`	Applies to systematic variation.
`local`	Applies to systematic variation.
`random`	Applies to random variation.

`-margin`

Treats the *`value`* in the {*`parameter value`*} pair as an offset from nominal. Default: *`value`* is treated as an absolute value.

This option cannot be used with `-sigma` or `-pelgrom`.

For example, if the nominal for `par1` is `1`,

- The following defines random variation around `par1 = 2`.

  ```
  define_variation -type random { par1 2 } local1
  ```

  In this case, `2` is an absolute value that is set to `par1` (`par1 = 2`).

- The following defines random variation around `par1 = 1 + 2 = 3`.

  ```
  define_variation -type random -margin { par1 2 }
  local1
  ```

  In this case, `2` is a relative margin value that is added to `par1` (nominal value = `1`). The new `par1 = 1 + 2`.

`-pelgrom`

Uses the Pelgrom effect to model random variation, interpreting parameter values as *Pelgrom* coefficients.The amount of parameter variation is scaled inversely proportional to the square root of the transistor area.

A Pelgrom model is useful because larger transistors are less sensitive to random variation than small transistors.

`-sigma`

Treats the value in the {*`parameter value`*} pair as a sigma value. This option requires that Variety has read and flattened the SPICE models so the specified sigma value can be mapped into the real value specified in the gauss or agauss setting in the model file.

This option cannot be used with the `-margin` option.

`-type <systematic | random>`

Specifies the type of variation to be used. Default: `systematic`

To turn on temperature variation, see the description for {*`parameter value`*}.

	`systematic`	Specifies that all transistors within each cell are assumed to vary together, e.g. a 5% variation in transistor length is applied to all the devices before measuring the overall sensitivity to the change in length.
	`random`	Specifies that each transistor within a cell is assumed to vary independently and the impact of each transistor on the resulting delay, transition, pin-capacitance and timing constraints is calculated. After calculating the impact of each transistor, the overall random variation impact is the standard deviation of all of the contributions from each transistor.

`-unit <`*`string`*`>`

Specifies the unit of the parameter set, either *nm* for length or width parameters, *mV* for threshold voltage parameters, or *A* (angstrom) for oxide thickness. The parameter values are scaled according to the `-unit`. If no `-unit` is specified, the parameter values are assumed to be in MKS units and the output sensitivity values in the library are in library units. The default is *ns* for delay and *pf* for capacitance. The complete set of library units with 1, 10, or 100 prefix, are supported.

The `-unit` option can be specified only if the `-type` option is set to `systematic`.

{*parameter value*}	(Required positional option) Specifies a list of parameter-value pairs consisting of the name of a parameter to be varied and the value to be assigned to that parameter. The *value* overrides any value set in the SPICE models. To enable temperature variation, set *parameter* to the keyword `temp` and set <*name*> (the variation name) also to the keyword `temp`. For example: `define_variation -margin -type systematic {temp value} temp` Temperature variation is treated as systematic in AOCV and is only characterized when you specify the `-margin` option. Temperature ranges are supported by using the `aocv_define_temp_range` command. If a range is set, the *value* option is used as a step applied to the range, otherwise, *value* is treated as a plus-minus offset from the nominal temperature (see `set_operating_condition`).
<*name*>	(Required positional option) Specifies the name of the variation being defined.

You can specify multiple `define_variation` commands, giving each a unique name. Multiple model parameters can be given, each with an associated absolute variation value (typically one sigma variation, although any value is permitted). All parameters defined within the same `define_variation` command are assumed to be fully correlated. Each `define_variation` option set is deemed to be fully uncorrelated to any other `define_variation` set.

Variety can characterize the sensitivity of a cell to a variation in the supply voltage (both `vdd` and `gnd`). To enable supply voltage variation, provide a `define_variation` command with the following arguments:

- `-type` set to `systematic`
- `-margin`
- {*parameter* `voltage`}
- an appropriate variation name

To vary all positive-supply voltages simultaneously, use `"voltage"` for the parameter and the variation name. For ground supply variation, use the keyword `"ground"` instead of `"voltage"` in the `define_variation` command. To vary a specific supply, use the supply name itself for the parameter and variation name. The value is specified in volts, but can also be specified as a ratio of the supply. For more information about specifying the value as a

percent, refer to variation_voltage_variation_use_percent variable. An example for the same is provided below.

```
sset_var variation_voltage_variation_use_percent 1
define_variation -type systematic -margin { voltage 0.05 } voltage
define_variation -type systematic -margin {ground 0.01} ground
define_variation -type systematic -margin { vdd1 0.02 } vdd1
```

Note: For ground variation, if `variation_voltage_variation_use_percent` is set to `1`, "`0.1 * vdd`" will be applied to the ground variation. If `variation_voltage_variation_use_percent` is set to `0`, then `0.1 Volt` will be applied to the ground variation.

Examples

```
define_variation \
    -type systematic \
    -unit "nm" \
    {dxl 0.5 dxw 0.8} \
SYSTEMATIC_WL

define_variation \
    -type systematic \
    -unit "A" \
{toxn 3.0 toxp 3.0} \
SYSTEMATIC_TOX

define_variation \
    -type random \
    -pelgrom \
{dvthn 0.05 dvthp 0.05} \
RANDOM_VTH

define_variation \
    -type systematic \
    -sigma \
{a1 3} \
SYSTEMATIC_a1
```

define_variation_average

Specifies how a single sensitivity is calculated from negative and positive sensitivity.

Options

`-max_factor <value>`	Specifies a factor to be applied to the maximum of the negative or positive sensitivity (as determined by applying the `-random` or `-systematic` options). Default: 0.5

`-min_factor <value>`

Specifies a factor to be applied to the minimum of the negative or positive sensitivity (as determined by applying the `-random` or `-systematic` options).

Specifies the minimum factor. Default: 0.5

`-random <simple | abs | abs_follow_pos_sign | simple_pos_is_max | abs_sign_follow_pos_magnitude>`

Determines the min and max selection criteria for random variation. Default: (none)

`abs` Selects the min and max based on their absolute values.

`simple` Selects the min and max based on their values.

`abs_follow_pos_sign`

Takes the absolute value of the positive and negative values, then calculates the average, but applies the same sign from positive sensitivity to the average (that is, the average takes the sign from positive variation.)

`simple_pos_is_max`

Computes the average by using the following equation:

```
avg = pos * max_factor + neg * min_factor
```

Here,

`pos` = the variation computed when the value specified in the `define_variation` command is applied to the circuit.

`neg` = the variation computed when the negative of the value specified in the `define_variation` command is applied to the circuit.

`abs_sign_follow_pos_magnitude`

Computes the average by using the following equation:

```
avg = max( abs(pos), abs(neg) ) *
max_factor + min( abs(pos), abs(neg)) *
min_factor
```

Here, if `pos > neg`, then `avg = -avg`.

`-systematic <simple | abs | abs_follow_pos_sign | simple_pos_is_max | abs_sign_follow_pos_magnitude>`

Determines the min and max selection criteria for systematic variation. Default: (none)

`abs` Selects the min and max based on their absolute values.

`simple` Selects the min and max based on their values.

`abs_follow_pos_sign`

Take the absolute value of the positive and negative values, then calculate the average, but apply the same sign from positive sensitivity to the average (that is, the average takes the sign from positive variation.)

`simple_pos_is_max`

Computes the average by using the following equation:

```
avg = pos * max_factor + neg * min_factor
```

Here,

`pos` = the variation computed when the value specified in the `define_variation` command is applied to the circuit.

`neg` = the variation computed when the negative of the value specified in the `define_variation` command is applied to the circuit.

`abs_sign_follow_pos_magnitude`

Computes the average by using the following equation:

```
avg = max( abs(pos), abs(neg) ) *
max_factor + min( abs(pos), abs(neg)) *
min_factor
```

Here, if `pos > neg`, then `avg = -avg`.

For example, given the following values, the calculations for `simple`, `abs`, and `abs_follow_pos_sign` are shown below:

```
-min_factor 0.2
-max_factor 0.8

negative sensitivity = 5
positive sensitivity = -10;
```

simple:

```
5 > -10, min = -10, max = 5
avg = min * min_factor + max * max_factor = -10 * 0.2 + 5 * 0.8 = 2
```

abs:

```
5 < 10, min = 5, max = 10
avg = min * min_factor + max * max_factor = 5 * 0.2 + 10 * 0.8 = 12
```

abs_follow_pos_sign:

```
5 < 10, min = 5, max= 10
avg = min * min_factor + max * max_factor= 5 * 0.2 + 10 * 0.8 = 12, since
positive sensitivity = -10 < 0, avg = -12 (follow the sign)
```

define_variation_factor

Defines a different characterization value for each type of arc.

Options

`{list_type_pairs}` Specifies a list of type-factor pairs where the type should be one of the following: `all`, `delay`, `constraint`, `setup`, `recovery`, `non_seq_setup`, `hold`, `removal`, `non_seq_hold`, `nochange`, and `mpw`.

Note: When `all` is specified as the type, it overrides only those types that are not mentioned explicitly in the list.

The define_variation command specifies the value for characterizing the sensitivity for a process parameter. This value is normally applied to all arc types. It is possible to characterize

each arc type at different sigma values. Use the `define_variation_factor` command to specify the sigma for characterizing each arc type. Each arc type specified in this command is characterized at the `define_variation` value times the factor specified in this command.

Note: When using the `define_variation_factor` command, the `define_variation` commands should be specified with values equivalent to `1` sigma and the variation_sigma variable should be set to `1`.

This command must be specified before the `char_variation` command is run.

Example

```
# Characterize delay at 3.75 times the value specified in the define_variation
commands.
#  Characterize hold at 3 times the value specified in the define_variation
commands.
# Characterize nochange at 1 times the value specified in the define_variation
commands.
# Characterize all other arcs at 2 times the value specified in the define_variation
commands.
define_variation_factor {delay 3.75 hold 3 constraint 2 nochange 1}
```

define_variation_group

Defines a variation group composed of a list of parameter variations.

Options

`-correlation {`*`correlation_values`*`}`

Specifies a matrix of variation values. Default: `1` on the diagonal, `0` for all other values

There must be enough values to fully populate the matrix. The number of values is determined by taking the square of the number of names in the list of parameter variations. The matrix values are checked for consistency, that is, value(x,y) must be equal to value(y,x).

`-sigma_factor <`*`value | ListOfValues`*`>`

Specifies a multiplication weighting factor to be applied to each variation. Use this option to shift the variation values from the characterized sigma. For example, if the `define_variation` is set for characterizing `1` sigma, you can use this option to adjust the variation to `3` sigma by using a value of 3.

If a single value is provided it is applied to all variations. If there is a single value for each variation, they will be applied to each variation in the order as specified in the variation list.

Examples:

```
# The variations are  characterized at 3 sigma.
# Adjust all variations to 1 sigma.
define_variation_group -sigma_factor " [expr 1.0/3]" {
tox dell delw } local_variation_group
# Specify a unique scaling factor to be applied to each
# variation. The variations are characterized at 1 sigma.
# Adjust tox to 3 sigma, dl to 3.5 sigma and dw to 3.25
# sigma
define_variation_group -sigma_factor { 3 3.5 3.25 } { tox
dl dw } myVarGroup
```

`{`*`varNames`*`}` (Required positional option) List of parameter variations, each defined by a `define_variation` command.

`<`*`name`*`>` (Required positional option) Specifies a name for the variation group.

This allows combining of un-correlated variations into a single set. This grouping is performed as a post-processing step after characterization and only impacts the creation of the SSTA model. This can be used, for example, to model effects that are uncorrelated with a single chip but are correlated across all chips, that is, global variation. The sensitivity values that are stored in the output library are the standard deviation of all the sensitivities from each *`varName`* parameter set.

Examples

Example #1:

```
define_variation_group \
    -correlation{ \
     1.0  0.5 -0.1 \
     0.5  1.0  0.2 \
    -0.1  0.2  1.0 } \
{TOX \
  XL \
```

```
  XW  } \
GLOBAL_VARIATION
```

In the following example, 3 process variations, TOX, XL and XW, are merged into a single process variation called GLOBAL_VARIATION, according to the given correlation matrix. In this particular example, TOX and XL are correlated with a coefficient of 0.5, TOX and XW are correlated with a coefficient of -0.1, and XL and XW are correlated with a coefficient of 0.2.

Example #2:

```
define_variation \
    -type systematic \
    {dxl 0.5 dxw 0.8} \
SYSTEMATIC_WL

define_variation \
    -type systematic \
    -unit 1e-9 \
{toxn 3.0 toxp 3.0} \
SYSTEMATIC_TOX

# Characterize for WL and TOX systematic variation
char_variation

# Define global variation group

define_variation_group { \
SYSTEMATIC_WL \
SYSTEMATIC_TOX \
} GLOBAL_VARIATION

# Write a library with SYSTEMATIC_WL, SYSTEMATIC_TOX
# and GLOBAL_VARIATION
write_variation -ecsm ssta.lib
```

get_var

Returns the current value of a Variety variable, whether the default value or a value set using the `set_var` command.

Options

<variable_name>	(Required positional option) Specifies the name of a Variety variable for which you want to determine the value. You can generate a list of Variety variables by using the `printvars` command.

Example

```
# Get the value of default_timing
get_var default_timing
```

printvars

Lists the current values of all of Variety's command variables. This command does not have any options.

Example

```
# List Variety's variables
printvars
conditional_constraint = 0
constraint_delay_degrade = 0.1
constraint_glitch_peak = 0.1
constraint_glitch_hold = 0
…
slew_lower_rise = 0.3
slew_upper_rise = 0.7
slew_lower_fall = 0.3
slew_upper_fall = 0.7
spice_delimiter = /.
```

read_ldb

Reads an existing library database (LDB) created by the `write_ldb` command.

Options

`-remove {`*`list_of_cells`*`}`

Specifies a list of cells to remove from the library database. Default: `none`

This option can be used to enable the re-characterization of a list of cells. By default, Variety does not characterize any cell that has already been loaded from an LDB. If the cell is removed from the LDB during loading, then the cell is re-characterized.

<ldb_name> (Required positional option) Specifies a library database file in LDB format or a directory name containing LDB format files (bundle mode).

The *ldb_name* can be compressed (ending in `.gz`) or uncompressed. If the uncompressed file or directory does not exist, `read_ldb` checks for a compressed version of the file.

Bundle mode is enabled when multiple LDB files are saved into a directory. In this mode, *ldb_name* can be the name of the directory.

The library database can be later used for formatting the library data for a particular SSTA tool.

Variety can also use an existing LDB to recover from any characterization run that did not complete successfully. As each cell is characterized, it is saved to an LDB in the current directory named `altos.ldb.<#>.gz`. This temporary LDB can subsequently be read by Variety (using `read_ldb`) to complete the characterization. Any cells defined by `define_cell` that do not exist in the recovered LDB are subsequently characterized. A complete LDB that contains all the cells can then be saved using `write_ldb`.

Examples

```
# Read an ldb to generate additional formats
read_ldb tt_all.ldb.gz

# Write a variation library for Cadence
write_variation -ecsm  secsm.lib

# Recover from an incomplete characterization
read_ldb altos.ldb.123.gz
char_variation
write_ldb tt_all.ldb
```

read_library

Creates a Tcl template from an existing Liberty library. You can then use the Tcl template file with Variety.

Note: Use this command together with the `write_template` command.

Options

{lib_filenames} (Required positional option) Specifies a list of Liberty format library files to read in.

Example

```
# Generate a template for Variety
read_library tt.lib
write_template -auto_index variety_tt_template.tcl
```

read_spice

Reads in the SPICE netlists of cells along with the device models.

Important

It is recommended that you do not mix the netlist formats. For example, do not mix an HSPICE netlist with a Spectre model file or a Spectre netlist with an HSPICE model file. The different simulators may follow different parsing rules that can lead to problems during circuit flattening.

The `read_spice` command can read files that have been compressed using `gzip`. Specify the filename with or without the `.gz` suffix.

Options

`-format {hspice | spectre}`
: SPICE netlist format (Default: `hspice`)

`{`*`spice_netlist`*`}`
: (Required positional option) List of files with extracted circuit netlists in SPICE format, including models.

The models should be included in the list of SPICE files. The SPICE netlist and model formats supported by Variety are as follows:

- HSPICE netlist, Level=49, Level=53, and Level=54 models
- Spice3 netlist, BSIM3, and BSIM4 models
  - PSP

Examples

```
# Read in a group of SPICE cell netlists
read_spice {90nm_cmos.spi nand2x2.spi nor2x2.spi inv2x4.spi}
# Read in a group of SPICE cell netlists
set cells {nand2x2 nor2x2 inv2x4}
set spice_netlists {}
set csz [llength $cells]
```

```
for {set c 0} {$c < $csz} {incr c 1} {
    set cell [lindex $cells $c]
    lappend spice_netlists subckts/$cell.spi
}
read_spice $spice_netlists -model 90nm_cmos.spi
```

read_vdb

Reads in a list of VDB files previously created by the `write_vdb` command.

Options

`{<filename>}`	(Required positional option) Specifies a list of VDB files to load.

Use the `read_vdb` command in a Tcl command file before the `char_variation` command, ensuring that the Tcl file contains the same setup that was in place when the VDB file was created.

Use the `write_vdb`/`read_vdb` flow to speed up analysis by storing processed vectors and by enforcing a specific structure across multiple PVT corners.

read_truth_table

Reads in and validate truth table files.

Options

`{<filenames>}`	(Required positional option) Specifies a list of Truth Table files that contain one or more truth tables to load.

Use the `write_template` command with the `-truth_table` and the `-auto_index` options to output a template file built from the loaded truth table file data. The template produced in this way has the following characteristics:

- The `define_pin_load` section is added when there is a tri-state bi-directional pin in the cell and one or more of the following attributes—*pull-up voltage*, *pull-up resistance*, *pull-down resistance*, or *serial-resistance*—is defined in the truth table.
- The `-pin_load` and `-load_dir` options are not added to the `define_arc` command. If required, you must add these options manually.

Note: For pull-up and pull-down pins, only pin capacitance is characterized. The `-pin_load` and `-load_dir` options should be added to the `define_arc` commands in the template to specify these values.

Examples

Example input truth table:

```
# CMOS Tri-State Output Pad with Schmitt Trigger Input and Pull-Up,
# High-V Tolerant
* PULLUP_RES=1000
* PULLDOWN_RES=1000
* SERIES_RES=25
* PULLUP_VOLT=3.3
* CELL=PDIO12
* TABLE= OEN I PAD @ PAD C
          0  0  -  @  0  0
          0  1  -  @  1  1
          1  X  0  @  -  0
          1  X  1  @  -  1
          1  X  -  @  Z  1
* TABLE_END
```

Example output template:

```
define_cell \
    -input {OEN I} \
    -output {C} \
    -bidi {PAD} \
    -pinlist {OEN I PAD C} \
    -delay delay_template_7x7 \
    -power power_template_7x7 \
    PDIO12

define_pin_load \
    -pullup_voltage 3.3 \
    -pullup_resistance 1000 \
    -pulldown_resistance 1000 \
    -series_resistance 25 \
    pin_load_template_PDIO12

# OEN -> PAD

define_arc \
    -type enable \
    -vector "F0FX" \
    -related_pin OEN \
    -pin PAD \
    PDIO12

define_arc \
    -type enable \
    -vector "F1RX" \
    -related_pin OEN \
    -pin PAD \
    PDIO12

define_arc \
    -type disable \
    -vector "RXRX" \
```

```
    -related_pin OEN \
    -pin PAD \
    PDIO12

define_arc \
    -type disable \
    -vector "RXFX" \
    -related_pin OEN \
    -pin PAD \
    PDIO12

# I -> PAD

define_arc \
    -vector "0FFX" \
    -related_pin I \
    -pin PAD \
    PDIO12

define_arc \
    -vector "0RRX" \
    -related_pin I \
    -pin PAD \
    PDIO12

# OEN -> C

define_arc \
    -vector "F0XF" \
    -related_pin OEN \
    -pin C \
    PDIO12

define_arc \
    -vector "F1XR" \
    -related_pin OEN \
    -pin C \
    PDIO12

define_arc \
    -vector "RX0F" \
    -related_pin OEN \
    -pin C \
    PDIO12

define_arc \
    -vector "RX1R" \
    -related_pin OEN \
    -pin C \
    PDIO12

define_arc \
    -vector "RXXR" \
    -related_pin OEN \
    -pin C \
    PDIO12

# PAD -> C

define_arc \
    -vector "1XFF" \
    -related_pin PAD \
    -pin C \
    PDIO12

define_arc \
    -vector "1XRR" \
```

```
    -related_pin PAD \
    -pin C \
    PDIO12
set cells { \
    PDIO12 \
}
```

select_index

Specifies the template index values to be used for simulation.

Options

`-style <value>` Specifies the method for selecting the template index values. The supported values are:

`"index"`	Uses the settings of the `-index_1` and `-index_2` options.
`"1x1"`	Uses the first index value. This is equivalent to `"-index_1 {1} -index_2 {1}"`.
`"2x2"`	Uses the first and last index values. For an index with seven values, this is equivalent to: `"-index_1 {1 7} -index_2 {1 7}"`.
`"3x3"`	Uses the first, middle, and last index values. For an index with seven values, this is equivalent to: `"-index_1 {1 4 7} -index_2 {1 4 7}"`.
`"mid"`	Uses the middle index value. For an index with seven values, this is equivalent to: `"-index_1 {4} -index_2 {4}"`.

`-index_1 { list } -index_2 { list }`

Specifies which index value positions specified by a define_template or define_index command are characterized. The position begins counting with `1`. Default: all indexes (indexes are not reduced).

The supported value is a list of specific positions within curly braces, such as `{ 2 3 5 }`.

`-cells { list }` Specifies a list of cells. Default: all cells

`-type { list }` Specifies a list of data types. Default: all supported data types

The supported data types are: `delay`, `power`, `constraint`, and `mpw`.

This command must be specified before the `char_variation` command is run.

set_client

Defines a machine or a queue to be used for distributed library characterization.

Options

`-dir <directory_name>{%N%U%P%S}`

(Required) Defines a directory on the client machine to use as a temporary workspace for simulation jobs performed on that machine. The tool creates the directory if it does not exist. You can incorporate the following objects into the name to create unique scratch directories for each individual validation run.

`%N`	Inserts the client number.
`%U`	Inserts the user name.
`%P`	Inserts the Variety server process id.
`%S`	Inserts the server name.

`-n <number_of_clients>`

Specifies that the tool is to submit jobs to this number of clients via the specified queue name.

When you use this option, all file names within the Tcl file must be full pathnames and the full pathname for the Tcl file must be specified when running the tool.

`<machine_or_queue_name>`

(Required positional option) Specifies the name of a client machine or a queue name.

As an alternative approach, you can instruct Variety to perform distributed processing by explicitly defining the names of each of the client machines. To specify multiple machines, use multiple `set_client` commands. The network port number to be used can also be set using

the `set_network_port` command. For more details on distributed parallel processing see Chapter 5, “Parallel processing.”

Examples

```
# Set 20 machines for use with the LSF queue
set_client  -dir /tmp/variety_%N -n 20 variety_lsf

# Or, explicitly set the client machines without a queue
set_client -dir /tmp/scratch/%U_%S_%P linux1
set_client -dir /tmp/scratch/%U_%S_%P linux2
```

set_conditional

Disables conditional arcs for delay and constraint groups for a list of cells.

Options

`-cells {`*`cell_names`*`}`

(Required) Specifies the list of cells affected by the command.

`-off`

Disables conditional arcs for the specified types of data. If the `-off` option is not used, the `set_conditional` command has no effect.

`-type {delay | const}`

Type of data for which conditional arcs are disabled. Default: `{delay const}`

Examples

```
# Turn off conditional delay and constraint arcs on ao32     # aO33 cells
set_conditional -off -type {delay const} -cells {ao32 ao33}
```

set_constraint

Adds margin to the constraint type when a library or datasheet is output.

Options

`-margin <value>`
Specifies the margin to add to timing constraints. Default: `0.0` seconds

The margin is applied before checking the value against the `-max` and `-min` limits. Margins are not cumulative.

`-max <value>`
Specifies the maximum constraint value. Default: `1e20`

`-min <value>`
Specifies the minimum constraint value. Default: `-1e20`

`-min_recrem <value>`
Specifies a minimum value (in seconds) that the sum of recovery+removal must exceed. If the minimum is not met, the removal values are adjusted to meet the minimum. Default: no checking

`-min_setuphold <value>`

Specifies a minimum value (in seconds) that the sum of setup+hold must exceed. If the minimum is not met, the hold values are adjusted to meet the minimum. Default: no checking

`-min_warning <0 | 1 | 2>`

Specifies the warning level when `-min_recrem` or `-min_setuphold` are not met. Default: 1

`0`	Issues no warnings.
`1`	Issues one warning per table.
`2`	Issues one warning per table value.

`-type {hold | mpw | recovery | removal | setup}`

Specifies the timing constraint type to which the criteria are applied. Default: the `-margin` value is applied to all constraints

`hold`	Specifies the *hold* type.
`mpw`	Specifies the minimum pulse width (MPW) type.
`recovery`	Specifies the *recovery* type.
`removal`	Specifies the *removal* type.
`setup`	Specifies the *setup* type.

If there are multiple `set_constraint` commands, the last command overrides earlier ones.

The margins and limits defined by `set_constraint` apply to nominal constraint values only. They do not apply to parameter variations.

Example

```
# Add 20ps margin to all constraints, 50ps to hold,
# enable warnings if sum is less than 0pS

set_constraint -margin 20e-12
set_constraint -type hold -margin 50e-12
set_constraint -min_recrem 0 -min_setuphold 0 -min_warning 2
```

set_constraint_criteria

Sets global and cell-specific setup and hold constraint criteria.

Options

`-cells {`*`cell_names`*`}`

Provides a list of cells that the specified criteria affect. Default: the specified criteria are applied globally

`-delay_degrade <`*`degrade_val`*`>`

Specifies the delay degradation relative tolerance—the maximum amount of nominal delay degradation permitted before an arriving signal is deemed to fail a timing constraint. The `<`*`degrade_val`*`>` is a value from 0.0 to 1.0. This option overrides the global variable constraint_delay_degrade when the `-cells` option is not used.

`-delay_degrade_abstol <`*`degrade_abs`*`>`

Specifies the delay degradation absolute tolerance—the minimum nominal delay degradation value permitted (in seconds). The maximum of the `-delay_degrade` percentage of the clock-to-output-delay or data-to-output-delay and the `-delay_degrade_abstol` is used as the delay degradation criteria. This option overrides the global variable constraint_delay_degrade_abstol when the `-cells` option is not used.

`-glitch_peak <`*`peak_val`*`>`

Specifies the maximum size of logic glitch permitted on the constraint output pin before an arriving signal is deemed to fail a timing constraint. This option overrides the global variable constraint_glitch_peak when the `-cells` option is not used.

`-metric <list>`

Specifies the constraint metric type. The supported values are: `delay`, `glitch`, `slew`, `width`, and `path_delta`.

Use this option for setting the timing constraint measurement criteria. The supported values are: `delay`, `glitch`, `width`, and `path_delta`. If `-metric` is specified both in `set_constraint_criteria` and `define_arc` commands, the `define_arc` setting takes precedence. If `-metric` is not specified for an arc, the usual *Inside View* decision process applies.

`-probe {probes}`

Specifies a list of nodes that can include one or more constraint criteria after any probe in the list.

`-slew_degrade <slew_degrade_val>`

Includes slew degradation when determining timing constraints. By setting the slew criteria, both delay degradation and slew degradation are checked and the first criteria to fail determines the setup and hold values. The `<slew_degrade_val>` is a value from 0.0 to 1.0 that represents the percentage of slew degradation and is measured using the values defined by the `measure_slew_lower_rise`, `measure_slew_upper_rise`, `measure_slew_lower_fall`, and `measure_slew_upper_fall` variables.

`-type <hold | recovery | removal | setup>`

Specifies the constraint type to which the criteria are applied. This option is ignored if the `-cells` option is not specified.

`hold`	Specifies the *hold* type.
`recovery`	Specifies the *recovery* type.
`removal`	Specifies the *removal* type.
`setup`	Specifies the *setup* type.

When this command is used to set global criteria, the following variables should not be set using the `set_var` command:

- constraint_delay_degrade
- constraint_delay_degrade_abstol
- constraint_glitch_peak
- constraint_slew_degrade

If both a variable and the `set_constraint_criteria` command (in global mode) are used, then the last command run sets the global criteria.

The `-probe` option supports the following two use models:

```
-probe { list of probe nodes }

-probe { probeNode1 crit1 value1 < crit2 value2 > ... probeNode2 crit1 value1
< crit2 value2 > ... }
```

Examples:

```
set_constraint_criteria -probe {a b}


set_constraint_criteria -probe {a -metric glitch -glitch_peak 0.2 b
-delay_degrade 0.1}
```

The set of supported criteria and their meaning is the same as the `set_constraint_criteria` command currently supports.

Specifying a criterion in the `-probe` option, such as `-delay_degrade`, modifies the criteria but does NOT enforces the metric. To enforce the metric, it must be specified explicitly using the `-metric` option; otherwise, Liberate chooses which metric should be applied.

The criteria specified for each probe overrides the criteria set by using `set_constraint_criteria` even if it is not specified in the same command. Also, if some criteria is specified with both `define_arc` and `set_constraint_criteria` commands, the criteria specified with the `define_arc` command takes precedence. One useful subtlety is that if a per-probe criterion is specified in a list of probe nodes and the `-probe` list is modified by a later `set_constraint_criteria` or `define_arc` command, the per-probe criteria still applies to probes in the new list, unless explicitly overridden. This allows the criterion to be specified once for all potential probe nodes and the probe node selection to be specified per-arc or per-cell.

This command must be specified before the `char_variation` command is run.

Example

```
# Set the hold criteria to 10% glitch for clock_gater
set_constraint_criteria \
    -type hold \
```

```
    -glitch_peak 0.1 \
    -cells clock_gater
# Set the setup criteria to 10% delay degradation and
# 50% slew degradation for the clock_gater
set_constraint_criteria \
    -type setup \
    -delay_degrade 0.1 \
    -slew_degrade 0.5 \
    -cells clock_gater
# set the global criteria
set_constraint_criteria  \
    -delay_degrade 0.15 \
    -delay_degrade_abstol 10e-12 \
    -glitch_peak 0.5 \
    -slew_degrade 0.5
```

set_default_group

Specifies the criteria for creating the default group. This command can be used to specify the global criteria for all cells or to specify the criteria for specific cells and arcs.

Options

`-cells {`*`cell_names`*`}`

Specifies a list of cells that the specified options affect. Default: This command applies to all cells globally.

`-criteria {<delay | cap | leakage> <off | min | avg | max>}`

Specifies type-value pairs that determine how the values in the default group data tables are selected. This option replaces the `default_timing` variable and if used globally overrides the value of that variable. The available types are:

`delay`	Specifies a type of timing delay, which accepts the values `off`, `min`, and `max`. Default: `max`
`cap`	Specifies a type of `cap`, which accepts the values `min`, `avg`, and `max`. Default: `max`
`leakage`	Specifies a type of leakage, which accepts the values `min`, `avg`, and `max`. Default: `avg`

`-method {<`*`group_type`*`> <`*`selection_method`*`>}`

Specifies the method used to select the data used to create the default group from multiple state-dependent groups.

Default: Finds the delay, power, and constraint table that has the worst value (reviewed bitwise) and uses that complete table (in its entirety) in the default group.

group_type Specifies the group, either `default` or `const`, that you want to apply the method to. The `default` value creates a default group and the `const` value creates a constraint group.

selection_method

Specifies the selection method, either `bitwise` or `table`, to be applied to the timing (delay/transition), cap, and constraint tables. The transition table in the default group follows the timing table selection.

`-pin {`*list*`}` Specifies a list of cell pins. Default: `"*"` (all cell bidirectional/output pins)

`-pin_dir <`*string*`>` Specifies the direction of the pins. The supported values are: `r`, `R`, `f`, `F`, `b`, and `B`. Default: `B`

`-related_pin {`*list*`}`

Specifies a list of related pins. Default: `"*"` (all cell input/bidi pins)

`-type <`*string*`>` Specifies the type of arc. The valid values are: `delay`, `constraint`, `leakage`, and `power`. Default: `""` (do not apply a type)

`-unateness <merge | separate>`

Controls whether *positive_unate* and *negative_unate* timing groups are merged into a single *non_unate* default timing group. Default: `merge`

This option applies only to default timing groups and has no effect on other data groups.

Note: If all *timing_sense* attributes are identical, then the original timing sense remains unchanged during merging. This option replaces the `default_unateness` variable and, if used globally, overwrites the value of that variable.

	`merge`	If both *positive_unate* and *negative_unate* timing groups exist, they are merged into a single *non_unate* default group.
	`separate`	Keeps *positive_unate* and *negative_unate* timing groups from being merged in the default group.
`-when <`*`string`*`>`		Specifies the state of the arc. Default: `""` (do not apply a state)

This command is used to choose which characterized state-dependent arc should be selected for the default group. To choose a specific arc, you must provide one or more of the following options that correspond to exactly one characterized arc for the list of cells: `-when`, `-type`, `-pin`, `-pin_dir`, `-related_pin`, and `-cells`. The `-when` and `-type` options are required. The other options are optional because they have reasonable default values. The `-cell`, `-pin`, and `-related_pin` options accept a wildcard.

Multiple occurrences of this command can be issued. If this command is issued more than once for the same cell, the last command issued for the cell overrides any previous settings for that cell.

Note: The parameters that are replaced by this command are used internally when no list of cells is provided and, if currently accessible, are accessible through Tcl. As these variables might not be supported in the future, Cadence strongly recommends using this command instead of the global variables.

This functionality does not support the ability to read in a library database (LDB), modify the default group settings, and write out a modified library.

Example 1

```
set_default_group \
    -method { default bitwise const table } \
    -unateness merge \
    -criteria { timing max cap avg } \
    -cells { inv nr2 }
```

Example 2

```
set_default_group \
    -type leakage \
    -when "!(A1)" \
    -cells "XOR3D1BWP"
```

Example 3

```
set_default_group \
    -type delay \
    -pin "Z" \
    -related_pin "A" \
    -when "A2 & A3"
```

set_dependent_load

Specifies a load to add to the specified cell:pin when the specified cell:pin is a dependent output. A dependent output is an output port of the cell that is in the path, but is not the monitored output of the timing arc being characterized.

Options

`-cells {`*`cell_names`*`}`

Specifies a list of cells that the specified options affect. Default: all cells

`-pinlist {`*`list`*`}` Specifies a list of pin names. Default: all pins

`<`*`load_value`*`>` (Required positional variable) Specifies the load to add to dependent pins, in Farads. Default: use the same load as the active output for input to output arcs

The dependent load is used for all characterizations such as delay, constraint, minimum pulse width, and so on.

Example

```
set_dependent_load -cells {fdrs15} -pin {X1} 5e-15
```

set_driver_cell

Defines an active pre-driver cell that ensures that the characterized delay and output slew values more realistically model the typical on-chip behavior by applying a non-linear PWL waveform on the input pin.

Options

`-accuracy_mode <0 | 1>`

Determines whether an algorithm is enabled for generating waveforms while observing the slew behavior. Default: `0`

The recommended setting of this option is `1`.

`0`	Does not enable an algorithm for generating waveforms while observing the slew behavior.
`1`	Enables an algorithm for generating waveforms while observing the slew behavior.

`-char_pin <`*`pin`*`>` If the driver cell has more than one output pin, you can use this option to specify the primary output pin where slew matching is performed. The transition is measured on the specified pin. If the `-char_pin` option is specified, then the `-pin_map` and `-pinlist` options are also required.

`-input_transition <`*`value`*`>`

Specifies the input transition time, in seconds. Default: `5e-12`

`-instantiate` Use this option to allow the instantiation of driver cells for constant side input pins during characterization. This option causes this driver cell to be used for the specified cell/pin in the SPICE deck when the specified pin is a side pin and is static. The use of this functionality can result in a significant (~20%) run time penalty.

`-pin_map {<`*`driver_pin`*`> <`*`cell_pin`*`>}`

Use this option to specify the driver cell pin that drives each pin in the `-pinlist`. The `-pin_map` maps by position to the `-pinlist` pins with the first `-pin_map` cell corresponding to the first `pinlist` pair, etc.

`-pinlist {<`*`cell`*`> <`*`pin`*`>}`

The `-pinlist` option specifies a list of pin pairs between driver cell output pins and characterization cell input pins.If a `-pinlist` is given then the driver cell is used for only the specific cell and pin pairs in the `-pinlist`. The cell names can be wild-carded with (`*`).

`<`*`driver_cell`*`>` (Required positional option) Specifies the name of the driver cell.

In nanometer technologies, it is common to have non-linear signal transitions. Using an active pre-driver cell ensures that the characterized delay and output slew values more realistically model the typical on-chip behavior by applying a non-linear PWL waveform on the input pin. A good choice for a pre-driver is to use a strong buffer cell.

When characterizing CCS data, Synopsys recommends using a CCS predriver waveform instead of an active driver cell. For more information about this, see the variable predriver_waveform.

By default, Liberate uses a linear ramp as the input waveform during characterization. The `set_driver_cell` command defines an active pre-driver cell to be used instead of the linear waveform. This pre-driver cell is driven at it's input by a linear ramp defined by the `-input_transition` option. Liberate determines the loading on the output of the pre-driver such that the output transition of the driver cell as measured on the `-char_pin <pin>` are equivalent to the input transitions specified in the `define_template` or `define_index` commands when measured at the values defined by the `measure_slew_lower_rise`, `measure_slew_upper_rise`, `measure_slew_lower_fall`, and `measure_slew_upper_fall` variables.

Variety supports an active driver that can simultaneously drive multiple inputs to a characterization cell. This capability allows multiple inputs to include delay offsets between related signals such as CK and CKN.

You can specify multiple `set_driver_cell` commands if necessary.

This `set_driver_cell` command must be specified before the `char_variation` command is run.

Examples

```
# Set the default pre-driver with a 10ps input ramp
set_driver_cell -input_transition 10e-12 bufx16

# Set the default pre-driver for all CLK pins, GATER:CLKIN

set_driver_cell \
    -input_transition 10e-12 \
    -pinlist {* CLK GATER CLKIN } \
    clkbufx4

# connect driver cell output X to inputs CK and SE on cell DFF1, and
# connect driver cell output Y to inputs CKN and SEN on cell DFF1.

set_driver_cell \
    -input_transition 6e-12 \
    -char_pin X \
    -pinlist { X CK X SE Y CKN Y SEN } \
    -pin_map { DFF1 DFF1 DFF1 DFF1 } \
    active_driver_2

# Set the active driver mode for more accurate generation of
# driver waveforms
```

```
set_driver_cell -input_transition 3e-12 \
    -accuracy_mode 1 \
    GL_CKBUFX14
```

set_gnd

Defines the names of ground nets. You can specify multiple `set_gnd` commands if necessary.

Options

`-cells`	Specifies a list of cells that the specified options affect.
`-name_map <value>`	Specifies the name that this supply is called in the output `.lib` file. If the `-name_map` option is specified, the `-cells` option must also be specified; otherwise, Variety reports an error. Maps the supply pin to a different named supply.

`-type <backup | deepnwell | deeppwell | internal | primary | pwell | well>`	Specifies the power supply type. Default: `primary`
	`backup`
	`deepnwell`
	`deeppwell`
	`internal`
	`primary`
	`pwell`
	`well`

`gnd_net <name>`	(Required positional option) Specifies the name of the ground supply net.
`voltage <value>`	(Required positional option) Specifies the ground value (in volts).

Variety automatically identifies the net names `0`, `GND`, and `VSS` (case insensitive) as ground supplies and sets them to zero volts. Use the `set_gnd` command to set them to different values.

set_network_port

Defines an explicit network port number to be used for distributed library creation.

Options

`<port_number>`	(Required positional variable) Specifies the network port number.

By default, Variety searches for an available port. To specify the machines to use for parallel processing, use multiple `set_client` commands. For more details on distributed parallel processing, see "Parallel processing" on page 243.

Examples

```
# Set the network port on the host machine
set_network_port 20000

# Set the client machines to use for parallel processing
set_client -dir /tmp/variety linux1
set_client -dir /tmp/variety linux2
```

set_operating_condition

Defines the process corner, temperature, and default voltage to be used for library creation.

Options

`-process <name>`	(Required) Specifies the name for the process corner. This name should correspond to a `.LIB` name in the SPICE models.
`-temp <value>`	(Required) Specifies the temperature to use for the characterization in °Celsius.
`-voltage <value>`	(Required) Specifies the default power supply voltage in volts.

The specified voltage is assigned to any VDD pin name. To specify additional power or ground-supply nets and their appropriate values, use the `set_vdd` and `set_gnd` commands.

Examples

```
# Characterize using typical process, 25°C, 1.2 Volts
set_operating_condition -process TT \
    -temp 25 \
    -voltage 1.2
```

set_pelgrom_equation

Overrides the default Pelgrom equation.

Options

`-parameter <`*`name`*`>` Specifies the name of a SPICE parameter or model name that has been loaded. Default: none

`-equation "`*`equation`*`"`

Specifies the Pelgrom equation to use. Default: none

The equation can consist only of L, W, numbers, Tcl variables, and functions that are accepted by SPICE. This command is effective only when the `define_variation -pelgrom` option is provided, in which case the `define_variation` parameter value is replaced by the Pelgrom equation provided by this command. If the `define_variation -pelgrom` option is provided, but `set_pelgrom_equation` is not defined for a given transistor model, the default Pelgrom equation is used. The default equation is the *`value`* specified in `define_variation` divided by the square root of L*W.

Example

```
set An 2.1e-10
set Ap 1.2e-10
set B 0.001
set C 0.002
set_pelgrom_equation -parameter mn_lvt -equation "($An /
sqrt(L*W)) + $B"
set_pelgrom_equation -parameter mp_lvt -equation "($Ap /
    sqrt(L*W)) + $C"
```

set_pin_capacitance

Specifies how the *simple capacitance*, *rise_capacitance*, and *fall_capacitance* attributes for each pin are determined.

Options

`-direction <min | avg | max>`

Specifies how the capacitance value is selected from the rise_capacitance and fall_capacitance values developed as a result of applying the `-table` option. Default: `max`

`min`	Uses the minimum value.
`avg`	Uses the average value.
`max`	Uses the maximum value.

`-state <min | avg | max>`

Specifies which values to use from all the tables among each logic state (`when` condition). Default: `max`

`min`	Uses the minimum value from all the tables.
`avg`	Uses the average value from all the tables.
`max`	Uses the maximum value from all the tables.

`-table <min | avg | max>`

Specifies which entries to select in the consolidated table developed as a result of applying the `-state` option. From these selections, a single value for rise_capacitance and a single value for fall_capacitance is extracted. Default: `max`

`min`	Uses the minimum values from the consolidated table.
`avg`	Uses the average values from the consolidated table.
`max`	Uses the maximum values from the consolidated table.

To understand how this command works, think of the options being applied in sequence:

1. `-state` option
2. `-table` option
3. `-direction` option

The selection emerging from the first step is evaluated by the `-table` option, and the resulting selection is then evaluated according to the `-direction` option. The third step determines the final capacitance value.

This command can be used independently from characterization, only impacting the attributes output by `write_variation` as the characterization database (LDB) contains all the rise/fall state-dependent capacitance tables for each pin.

Example

```
# Set how the pin-capacitance attributes are determined
set_pin_capacitance -state max -table avg -direction min
```

set_pin_gnd

Associates a pin of a cell with a particular ground supply voltage.

Options

`-add_supply`
: Creates and adds `-supply_name <`*`name`*`>` to the cell ground list.

`-supply_name <`*`name`*`>`
: (Required) Specifies the name of the supply that drives this pin.

  This is useful when a level shifter is being characterized with a particular PVT where both the input voltage and the output voltage are the same (but are different for other PVTs). By specifying the supply_name, this avoids matching an incorrect supply to the input pin. In addition, it also fixes CCSN stage generation for level shifters so that Liberate not only considers the input/output voltages of a timing arc when deciding if an arc stage is to be used, but it also checks to see if the input/output shares the same voltage supplies before using the arc based constructs.

  When the `-supply_name` option is used and the `voltage_map` command is set to `1`, the specified `-supply_name` is output in *related_power_pin/ related_ground_pin* format. If `voltage_map` is set to `2`, the *input*/*output_signal_level* attributes are used instead.

<cell_name>	(Required positional option) Specifies the name of the cell. Wildcards are not supported.
<pin_name>	(Required positional option) Specifies the name of the pin.
<gnd_value>	(Required positional option) Specifies the ground supply value.

This command is useful for setting ground supplies on cells that have multiple power connections, such as level shifters. Typically, `set_pin_gnd` is used in conjunction with `set_pin_vdd`.

set_pin_vdd

Associates a pin of a cell with a particular supply voltage.

Options

`-add_supply`	Indicates that the supply specified with `-supply_name` is to be added to the cell VDD list.
`-leakage_add_to_supply` *<name>*	Specifies the name of a supply to which all leakage for this port should be added. This option should be used when gate leakage on an input pin is to be added to a power pin not controlling it. This is useful when characterizing level shifters.

`-supply_name` *<name>*	(Required) Specifies the name of the supply that drives this pin. This is useful when a level shifter is being characterized with a particular PVT where both the input voltage and the output voltage are the same (but are different for other PVTs). By specifying the supply name, you can avoid matching an incorrect supply to the input pin. In addition, it also fixes CCSN stage generation for level shifters so that Liberate not only considers the input/output voltages of a timing arc when deciding if an arc stage is to be used, but also checks to see if the input/output shares the same voltage supplies before using the arc-based constructs. When the `-supply_name` option is used and the `voltage_map` command is set to `1`, the specified `-supply_name` *<name>* is output with *related_power_pin* / *related_ground_pin* attributes. If `voltage_map` is set to `2`, the *input_signal_level* / *output_signal_level* attributes are used instead.
{*cell_names*}	(Required positional option) Specifies the name of the cell. Wildcards are not supported.
<pin_name>	(Required positional option) Specifies the name of the pin.
<vdd_value>	(Required positional option) Specifies the power supply value.

If there is more than one `set_pin_vdd` command for the same pin but with a different value or different VDD name, the latest instance of the command overrides the first. Furthermore, if one of the commands does not specify the `-supply_name` and `-add_supply` options, Liberate issues an error and exits.

This command must be specified before the `char_variation` command is run.

Examples

Example 1:

```
# Set the voltage swing on the input pin of a level shifter
set_pin_vdd -supply VDD3 level_shifter_3to1 A1 3.0
```

Example 2:

```
set_vdd VDD $VOLT1
set_vdd VDD1 $VOLT
set_pin_vdd -supply_name VDD1 -leakage_add_to_supply VDD \
LVLHLD1HVT   IN   $VOLT
```

In the above example, gate leakage currents on pin `IN` are multiplied by `$VOLT` (controlling `VDD1`) to get the leakage power that is added to the supply pin `VDD`.

set_units

Specifies the timing, capacitance, and leakage power units.

Options

`-capacitance <1pf | 100ff | 10ff | 1ff>`

Specify the capacitance units. Default: `1pf`

`-leakage_power <1mw | 1uw | 1nw | 1pw>`

Specify the leakage power units. Default: `1nw`

`-timing <1ns | 100ps | 10ps | 1ps>`

Specify the timing units. Default: `1ns`

Example

```
# Set the timing units to 1pS
set_units -timing 1ps
```

set_var

Sets the value of Variety-specific variables. The variables available for use in Variety are described in Chapter 4, “Variety Variables.”

Options

`-cells` List of cells. Default: all cells

`-pin {`*`pins`*`}` List of destination pins for the arc (typically, output pins for combinational arcs, input pins for timing constraint, or hidden power arcs). (REQUIRED)

`-pin_dir <R | F>` Transition direction of pin(s).

`-related_pin {`*`pins`*`}`

List of related pin names (typically input pins for combinational arcs, clock pins for timing constraint arcs).

`-related_pin_dir <R | F>`

Transition direction of related pin(s).

`-type < constraint | delay | delay and power | hold | leakage | mpw | nochange | power | recovery | removal | setup >`

Type of arc. (Default: *all types*)

<parameter> (Required positional option) Parameter name.

<value> (Required positional option) Parameter value.

The options `-cells`, `-type`, `-pin`, `-pin_dir`, `-related_pin`, and `-related_pin_dir` are used to specify local cell and arc specific variables and their corresponding values. All options are not valid for all parameters. If an option is not allowed, an error is issued and the setting is ignored. If an option is omitted, any value for that option is allowed. The options `-cells`, `-pin`, and `-related_pin` support the usage of the wildcards `*` and `?`. Some variables can only be set at a global level. If you specify local cell and arc variable that can only be applied globally, a warning is issued in the log file and `set_var` is ignored.

Examples

```
# Set the variable 'max-transition' to 1ns
set_var max_transition 1e-9
```

set_vdd

Defines the names of power supplies. You can specify multiple `set_vdd` commands if necessary.

Options

`-cells`	Specifies a list of cells that the specified options affect.
`-name_map <value>`	Use the `-name_map` option to specify the name that this supply is called in the output .lib file. If the `-name_map` option is specified, the `-cells` option must also be specified; otherwise Variety reports an error. Maps the supply pin to a different named supply.

`-type <backup | deepnwell | deeppwell | internal | primary | pwell | well>`	Specifies the power supply type. Default: `primary`
	
	`backup`
	
	`deepnwell`
	
	`deeppwell`
	
	`internal`
	
	`primary`
	
	`pwell`
	
	`well`

`gnd_net <name>`	(Required positional variable) Specifies the name of the power supply net.
`voltage <value>`	(Required positional variable) Specifies the voltage value (in volts).

Variety automatically identifies the net name `VDD` (case insensitive) as a power supply and sets it to the voltage specified by the command. Use the `set_vdd` command to set it to a different value.

Example

```
# Set VDD3 to 3 volts
set_vdd VDD3 3
set_gnd BULK_GND 0
```

write_ldb

Creates a library database in LDB format.

Options

<filename>	(Required positional option) Specifies a name for the library database file to be created.

The LDB can then be used in a later Variety session to create library data formatted for a particular statistical static timing analysis (SSTA) tool.

Variety automatically saves each cell, as it is characterized, to an LDB named `altos.ldb.<#>.gz` located in the current directory. The `write_ldb` command renames this file to the name specified by the `write_ldb` command.

Cadence recommends that the `write_ldb` command be executed immediately following the `char_variation` command and before any model creation commands such as `write_variation`. This practice ensures that there is a clean, unmodified copy of the LDB saved for future use. This is important because, for example, when user data is loaded with `write_variation -user_data`, the internal database is modified by the user data and any LDB subsequently saved contains those modifications.

Example

```
# Characterize the library
char_variation
# Save the library database to tt.ldb.gz
write_ldb tt.ldb
```

write_socv

Writes out an SOCV format file. This file can be loaded into a Static Timing Analysis tool to account for statistical variation effects.

Options

`-cells {`*`cell_names`*`}`

The `-cells` option controls which cells get written to the SOCV file. Default: all cells.

This option supports the use of a wildcard.

`-exclude` Reverses the meaning of the `-cells` list, so that the specified list of cells are excluded.

`-index1` Specifies the index1 selection. Default: `all` index1 from the library.

`-index2` Specifies the index2 selection. Default: `all` index2 from the library.

`-`*`filename`* Specifies the output library file name to write to the variation table. Default: `${libname}.socv`

`<`*`library_name`*`>` (Required positional option) Specifies the name of the output library.

`-type` Specifies to output only the specified types of timing. By default, both delay and constraint timing are output.

Following are the valid values for this option: `"delay"`, `"constraint"`, and `"all"`.

This command must be specified after a `char_variation` or `read_ldb` command is run.

Example

```
write_socv -filename mySOCV.tbl -libname slowCorner
```

write_template

Creates a Variety Tcl command file template by reading an existing library (`.lib`) or library database (LDB).

The Tcl command file is named based on the user-input provided for the `<`*`filename`*`>` option (`.tcl` is appended to the filename if it does not end in `.tcl`). The Tcl file includes all the necessary define_template and define_cell commands needed to run Variety. This function provides a convenient way to use an existing library's templates to create the Tcl file to characterize a new library.

Options

`-abstol <value>`	Sets the absolute tolerance for comparing templates. Default: `0.0`
`-auto_index`	Generates templates suitable for use with the `-auto_index` feature of `char_variation`. When `-auto_index` is used, all cells refer to a single delay and power template whose size is denoted by the `-index_delay` option (default `7x7`) and a single constraint template whose size is denoted by the `-index_const` string (default `3x3`). The values of the indices in the templates are used as scaling factors for the indices automatically determined from the minimum and maximum transition and minimum load where these are extracted from the input library by the `write_template` command. The `-unique` and `-define_index` options cannot be used with the `-auto_index` option.
`-cells {cell_names}`	Controls which cells get written to the template. Default: all cells This option supports the use of a wildcard.
`-define_index`	Generates a `define_index` command for each arc of a cell whose indices differ from the default indices for the cell. Default: `write_template` assumes all the data of the same type (delay, power, etc.) within a cell uses the same template. The `-define_index` option is useful when there are distinct indices for different paths within that cell, where the slew and loading conditions used for characterization are different. The `-define_index` option cannot be used with the `-auto_index` option.
`-exclude`	Reverses the meaning of the `-cells` list, so that the specified list of cells are excluded from the template.
`-group <number>`	If the `-group` option is specified, the Tcl file only contains cell definitions and templates for a `<number>` of cells per group. A group is created by either using the `define_group` command or by sharing the same footprint name. This option can be useful to generate a list of cells for a trial characterization run with a representative subset of the cells in the library. Default: `all`

`-index_const <NxM>`	Denotes the constraint template sizes. Default: `3x3`
`-index_delay <NxM>`	Denotes the delay template sizes. Default: `7x7`
`-input_supply_pin`	Outputs `set_pin_gnd` and `set_pin_vdd` commands into the template file. The original library must have pg_pin syntax with related power nodes in the pin group.
`-io`	Creates templates with `define_arc` commands for I/Os. This option is equivalent to the `-verbose` option.
`-skip {leakage}`	Disables the output of `define_leakage` commands into the template file. This option should be used only when *Inside View* is enabled. It cannot be used with the `char_variation -io` option, which disables *Inside View*. If there are no `define_leakage` commands loaded and *Inside View* is not enabled, the resulting library does not have any leakage states characterized.
`-sort_pinlist in_bi_ou`	Accepts only the value `in_bi_ou`. When this option is used, the `define_cell -pinlist` option in the template has the pins sorted as follows: "input bidi output".
`-truth_table`	Writes a template file out from the truth table data that has been loaded using the `read_truth_table` command. When this option is used, the `-auto_index` option must also be used. The following options are ignored when using this option: `-cells`, `-define_index`, `-exclude`, `-group`, `-io`, and `-unique`.
`-unique`	If the `-unique` option is used each cell has its own unique set of template definitions, otherwise cells share templates where the templates are identical. Two templates are deemed identical if they have the same type, the same number of indices, and each index is within `-abstol` (default 0.0) to each other. The `-unique` option cannot be used with the `-auto_index` option. **Note:** The `-unique` option always writes uniquified templates for each cell by adding a sequential numerical suffix. This option should not be combined with `-use_lu_table_name` because the resulting library will not match the original library.
`-use_lu_table_name`	Reuse the `lu_table names` from the original library in the Liberate `define_template` commands.

`-verbose`	Creates templates with `define_arc` commands for I/Os. This option is equivalent to the `-io` option.
`-when_as_vector`	Converts `when` conditions into a vector sequence rather than having a `-when` for each `define_arc`.
<*filename*>	(Required positional option) Specifies the file name for the Tcl template that is created.

A `read_library`, `read_ldb`, or `char_variation` command should be issued before using the `write_template` command.

When using the `-verbose` option, it is common to see invalid define_arc commands in the output template. This can occur if the `timing_sense` and/or unateness of an arc is not specified in the input library. Liberate cannot determine the real arcs; therefore, it outputs all combinations of `define_arc` commands for all possible cases from the `related_pin` to the output pin. For example, `define_arc` commands will cover all four cases of RR, RF, FF, FR for an arc where the `timing_sense` and unateness are absent. Two of the `define_arc` commands will be invalid and may cause an error condition during `char_variation` run depending on the value of `def_arc_msg_level`. To avoid having invalid `define_arc` commands in the template, the input library must contain complete `timing_sense` and unateness attributes. The template can be modified manually to remove the invalid arcs.

Example

```
read_library my.lib

# Output a Variety Tcl command file for auto_index
write_template -auto_index -index_delay 8x8 ai_template
```

write_variation

Outputs the variation model in the selected format.

Options

`-capacitance_only`	Omits the *rise_capacitance* and *fall_capacitance* attributes, leaving only a single *pin capacitance* attribute.
`-capacitance_range <0 | 1 | 2>`	Controls how the *rise_capacitance_range* and *fall_capacitance_range* attributes are output. Default: `1` `0` Omits the rise and fall capacitance ranges.

`1` Outputs the rise and fall range spanning from the minimum of the min_capacitance values to the maximum of the max_capacitance values. This method has been reported to cause timing issues in PrimeTime.

`2` Includes the rise and fall capacitance ranges where both range limits are set to the rise/fall capacitance attribute values:

```
rise_capacitance_range="<rise_
    capacitance>, <rise_capacitance>"
fall_capacitance_range="<fall_
    capacitance>, <fall_capacitance>"
```

`-ccs` Writes the CCS VA format data into the output library.

A set of CCS libraries is written, one for nominal, <*libname*>.nom.lib, and two for each parameter (one for increasing the parameter, <*libname*>.<*param_name*>_P.lib, and one for decreasing the parameter, <*libname*>.<*param_name*>_N.lib). After the complete set of CCS libraries is created, the final compact CCS VA library is created using the Synopsys Library Compiler® by calling `lc_shell` with a configuration file created by Variety called <*library_name*>_lc_config.tcl.

`-ccs_parameter <`*name*`>`

Specifies a name for and writes out the CCS library. Default: Variety dumps all the CCS libraries and automatically calls `lc_shell` to merge the individual libraries together.

If the `define_variation_group` command is used, the `-ccs_parameter <`*`name`*`>` must correspond to the variation group name.

Usage:

```
write_variation -ccs_parameter variationName
```

where

`variationName` outputs the *`libName_variationName`*`_N.lib` and *`libName_variationName`*`_P.lib` for that variation.

If `variationName` = `nominal`, Variety outputs the nominal library (*`libName`*`.nom.lib`) as well as the configuration files for running the LC merge of the individual libraries. In this case, you must run the `lc_shell` manually using

```
lc_shell -f "lib_name"_lc.tcl
```

`-ccs_va`

Outputs a single Primetime VX merged format file.

`-cells {`*`cell_names`*`}`

Specifies the cells to write into the output library. Default: all cells are written

This option supports the use of a wildcard.

`-driver_waveform_size`

Sets the number of voltage points in the normalized driver waveform index_2. Default: `500`

The normalized waveform uses an arbitrary number of voltage points uniformly distributed from gnd to vdd.

`-driver_waveform` Outputs normalized driver waveforms into the output library.

For the output to include the driver waveform, the LDB or VDB must contain the driver waveform data. If the Tcl contains multiple `write_variation` commands, the first command using this option enables the waveform output for all subsequent `write_variation` commands.

This option does not output normalized driver waveforms for user-defined PWLs that are incompletely specified or use wildcards.

`-ecsm` Writes ECSM format data into the output library.

`-exclude` Reverses the meaning of the `-cells` list, so that the specified list of cells are excluded.

`-extreme` Writes Extreme DA format data into the output library.

`-filename <`*`filename`*`>`

Specifies the output library filename. Default: the output library file is named *`libname`*`.lib`.

If the specified output library file already exists, a warning is given, and a unique file name is generated using the given name suffixed with a number.

`-format { "sensitivity" | "sensitivity_plus_nom" }`

Specifies the format of the output data file, that is, `sensitivity` or `sensitivity_plus_nom`.

If this option is set to `sensitivity`, the data file contains only the sensitivity data. However, when the option is set to `sensitivity_plus_nom`, the data file includes nominal data besides the sensitivity data. The nominal data is required when the `-sensitivity_normalize` option of the `add_margin` Liberate command is used.

The data file and the values contained in it can be loaded into Liberate to add margin to a corner library. For more information, see the following Liberate command options:

- `add_margin -sensitivity_file` and `-sensitivity_normalize`
- `write_library -sensitivity_file`

Note: The `sensitivity` format output file can be used by Liberate in the `add_margin` and `write_library` commands. While the two commands can be used as given below, it is not recommended to use both the commands in the same library creation because this can lead to double margining by the timer.

```
add_margin -sensitivity_file "xxx.lib"

write_library -sensitivity_file "yyy.lib"
```

`-gzip` Compresses the output file using gzip.

`-indent` Specifies the number of spaces to indent. Default: `2`

`-overwrite` Disables the automatic version control and overwrite any already existing output `.lib` library.

`-precision <`*`precision`*`>`

Specifies a format string that controls the precision used when writing out the library. The value for this option must conform to standard Tcl formatting. The default value is `%g`.

`-preserve_user_data_precision {`*`list_of_attributes`*`}`

Preserves the precision of the listed attributes in the `user_data` file and does not apply the `-precision` option to them.

`-rename` Renames the existing output library.

By default, Liberate checks whether an output library already exists. If it exists, a warning is printed and the output is written to a new library that is named based on the next available unused numerical index. However, when you specify the `-rename` option, the existing library is renamed using the next available unused numerical index and then the new library is written. This option is used to maintain the same output library filename for scripting purposes while also maintaining the history.

`-scan_output_dummy` Converts sequential cells (latches, flops) to scan dummy cells by removing all scan pins from a cells and writing out the reduced cells. For this to work, the `define_cell` command must include scan related information such as the scan pins.

`-sdf_edges` Enables the output of the *sdf_edges* attribute.

`-skip { constraint | delay | hold | mpw | nochange | setup }`

Lists the data types to be filtered from the LVF and extreme file. Default: none (do not skip any data)

`constraint` Variety skips all constraint timing types: `setup_rising`, `setup_falling`, `non_seq_setup_rising`, `non_seq_setup_falling`, `recovery_rising`, `recovery_falling`, `hold_rising`, `hold_falling`, `non_seq_hold_rising`, `non_seq_hold_falling`, `removal_rising`, and `removal_falling`.

`delay` Variety skips the following delay types: `cell_rise`, `cell_fall`, `rise_transition`, and `fall_transition`.

`hold` Variety skips `hold_rising`, `hold_falling`, `non_seq_hold_rising`, `non_seq_hold_falling`, `removal_rising`, and `removal_falling`.

`mpw`	Variety skips all `mpw` timing characterization. This includes attributes and `mpw` tables.
`nochange`	Variety skips the `nochange` arcs.
`setup`	Variety skips `setup_rising`, `setup_falling`, `non_seq_setup_rising`, `non_seq_setup_falling`, `recovery_rising`, and `recovery_falling`.

`-skip_weak_ecsm_sensitivity`

Reduces the size of the output library by removing ECSM waveform sensitivity data for parameters that contribute less than 20% to the delay impact of all of the parameters. This option applies only to the `ecsm` format.

`-spdf_file {`*`filename`*`}`

Outputs statistical parameter distribution format libraries (SPDF) for SECSM format libraries.

If you use this option, you must also use the `-ecsm` option.

`-unique_pin_data` Outputs unique data, such as timing and power, for each bus or bundle member. It specifies that the original pin names are to be used inside the `when` condition string without the pin names being processed to change them into bundle names.

`-user_data <`*`filename`*`>`

Specifies a user-provided library in Liberty format to be merged with the current library. This is useful to include non-characterized data models in the output library. After this user-data is merged into the current library, all subsequent `write_variation` commands output the merged constructs as part of the output library. If this is not desired, then separate runs of Variety consisting of `read_ldb` and `write_variation` must be executed. Any valid construct that is present in the user-provided library that is not present in the current library is copied to the output library with the following exceptions:

- Attribute *slew_derate_from_library* is not copied.
- Attributes *function*, *state_function*, and *area* overwrite values in the current library.
- Groups state_table, ff, and latch overwrite the equivalent groups in the current library.

<library_name> (Required positional option) Specifies the name of the output library.

Only one `write_variation` command is permitted in a single Variety™ run.

Examples

```
read_ldb variation.ldb.gz

# Output a library in Extreme format
write_variation -xt -user_data my.lib xt.lib

# Output a library in Cadence format
write_variation -ecsm -skip_weak_ecsm_sensitivity \
    -user_data my.lib secsm.lib

# Output a library in CCS VA format
write_variation -ccs ccs.lib

# Output a data file that contains sensitivity rules
write_variation -format sensitivity -filename MySensitivity.dat
```

write_variation_table

Outputs a one-stage POCV (derate) table to a file. Use this command along with `read_ldb` or `char_variation`.

Options

`-cells {`*`list`*`}` Specifies a list of cells to output.

`-chain_length` Specifies the length of the chain. Default: `1`

The `-chain_length` option is used to request a variation table with derate values for a chain of cells. Set this option to the number of instances to be characterized in a chain.

`-distance_list {list}` Specifies a list of distance to output to the AOCV table.

`-exclude` Reverses the meaning of the `-cells` list, so that the specified cells are excluded.

`-filename <`*`filename`*`>`

Specifies the output library file name to write to the variation table. Default: `${libname}.socv`

`-format <cadence | csv | synopsys>`

Specifies the output format to use. Default: `cadence`

`cadence` Uses the Cadence format.

`csv` Uses the comma-separated value (CSV) format. For example:

```
# Cell, depth, rise_or_fall,
early_or_late, derate, mean_delay,
sigma_delay, clock_slew, min_pulse_width
arc_info
a/DFFX1, 1, rise, early, 0.979766,
0.615234, -0.0124488, 0.25 "CK -> CK rise
when [D]"
```

`synopsys` Uses the Synopsys format.

`-load_index {list}` Specifies a list of output load index. The load index starts from `0`. Default: all load index

`-loads {list}` Specifies a list of output loads. Default: all loads from the library

`-sigma_factor {<name value>...}`

Specifies a list of name-value pairs for sigma scale factor for `delay`, `hold`, `recovery`, `removal`, and `setup`. If this is not specified, it defaults to `1.0` for all supported types.

`-slew_index {list}` Specifies a list of input slew index. The slew index starts from `0`. Default: all slew index

`-slews {list}` Specifies a list of input slews. Default: all slews from the library

`-type {data_types}` Specifies a list of data types. Supported types are: `delay`, `hold`, `min_pulse_width`, `recovery`, `removal`, and `setup`. Default: `delay`

- `delay` is supported for all formats.
- `hold`, `min_pulse_width`, `recovery`, `removal`, and `setup` are supported only in CVS format.

`-version` Specifies the version of AOCV format. The valid values are `1.0` or `2.0`.

`-wire_cap <auto | min | mid | max>`

Sets the wire capacitance between cells to a value in Farads, `min`, `mid`, or `max` from the delay table for the arc under test. Default: `min`

`min`	Sets the capacitance to the min load from the delay table.
`mid`	Sets the capacitance to the mid load from the delay table.
`max`	Sets the capacitance to the max load from the delay table.

`-wire_cap_index <index_point>`

Specifies a load value to use between cells based on the position in the `index_2` template. The value starts from `0`. If the specified value is not in the range of `index_2`, use the `-wire_cap` value instead. Default: `-1` (Use the value specified by `-wire_cap`.)

`-wire_cap_subtract_next_input`

Subtracts the next input capacitance from the `-wire_cap` value and uses that as a net wire cap.

`<library_name>` (Required positional option) Specifies the name of the output library.

Example

The following example illustrates the CSV format.

```
write_variation_table -cells $cells -load_index 2 -wire_cap_index 2
-chain_length 3 -filename new_data/wirecapidx -format cadence aaa
write_variation_table -cells $cells -load_index 2 -wire_cap 0.010203
-chain_length 3 -filename new_data/wirecap -format cadence aaa
```

write_vdb

Creates a Vector library DataBase (VDB) file for the current library.

Options

`-auto_index`

Instructs Variety to create the table indices for all constructs (except `si_immunity`) overriding the values specified in the given templates.

The number of entries for each index is taken from the appropriate pre-defined template. This feature uses the `max_transition` variable to determine the range of output loads for each cell. To automatically generate `si_immunity` indices set the `max_noise_width` variable.

`-extsim <hspice | spectre>`

Instructs Variety to use a specified external SPICE simulator. Default: Use the internal SKI simulator.

When the `-extsim` option is specified, temporary run directories called (for example) `altos0` or `altos1` are created to store the external simulation run-time files. These temporary run directories are created in the directory specified by the `TMPDIR` environment variable or in the initial run directory if `TMPDIR` is not set. If distributed processing is requested using the `set_client` command, these temporary run directories are created in the directory specified by the `-dir` option of the `set_client` command. The license for the external simulator must be available. Using an external SPICE simulator typically increases the characterization run time by a factor of two over using SKI.

`hspice` Uses the HSPICE simulator.

`spectre` Uses the Spectre simulator.

<vdb_filename> (Required positional option) Specifies the name of the output VDB file.

The VDB file includes vector data that is created during the preprocessing stage in Variety. This file can be used to speed up the preprocessing stage by storing the processed vector data and library structure into the VDB file. This file can be loaded using the `read_vdb` command. When loaded, the following `char_variation` command uses the vector and structure information stored in the VDB file and does not rerun the vector processing. This holds true for both the server and the client processes.

The `write_vdb` command should replace the `char_variation` command in a Tcl command file that has been fully set up to characterize a library. The `char_variation` command should not be executed in the same run as the `write_vdb` command.

This command must be specified after a database has been loaded.

4

Variety Variables

This chapter describes the following Variety-specific variables that impact process variation characterization.

Note: Variety-specific variables are set using the `set_var` command.

a...	
adjust_tristate_load	aocv_enable_clock_gater
aocv_chain_termination_mode	aocv_extra_driver
aocv_derate_method	aocv_sigma_factor
aocv_derate_mode	
b...	
binning_detail	bundle_mem_limit
c...	
ccs_abs_tol	constraint_clock_gater
ccs_base_curve_points	constraint_delay_degrade
ccs_base_curve_share_mode	constraint_delay_degrade_abstol
ccs_cap_hidden_pin	constraint_failed_value
ccs_cap_use_input_transition	constraint_glitch_hold
ccs_current_model_pin_load	constraint_glitch_peak
ccs_force_grid_delay	constraint_hold_probe
ccs_init_voltage_comp_thresh	constraint_info
ccs_max_current_thresh	constraint_linear_waveform
ccs_max_pts	constraint_merge_state
ccs_rel_tol	constraint_output_load
ccs_voltage_tail_tol	constraint_output_load_factor

ccs_voltage_tail_tol_mode	constraint_output_pin
char_effort_systematic_variation	constraint_random_variation_search_time_abstol
combinational_risefall	constraint_slew_degrade
conditional_cap_hidden_pin	constraint_search_bound
conditional_cap_hidden_pin_thresh	constraint_search_bound_estimation_mode
conditional_constraint	constraint_search_time_abstol
conditional_expression	constraint_snap_to_bound
conditional_include_constant	constraint_tran_end_extend
constraint_check_rebound	constraint_vector_mode
d...	
debug_flow	delay_inp_rise
default_capacitance	delay_out_fall
default_group_method	delay_out_rise
default_unateness	derate_comment_start_str
define_arc_merge_state	derate_comment_end_str
delay_inp_fall	disable_method
e...	
extsim_ccs_option	extsim_option
extsim_cmd	extsim_option_presim
extsim_cmd_option	extsim_reuse_ic
extsim_deck_dir	extsim_sanitize_param_name
extsim_deck_header	extsim_save_driver
extsim_deck_style	extsim_save_failed
extsim_exclusive	extsim_save_passed
extsim_lic_keep	extsim_tar_cmd
extsim_mc_append	extsim_timestep
extsim_model_include	extsim_tran_append
extsim_model_include_mode	extsim_use_node_name

extsim_monte_option	extsim_variation
f...	
force_condition	force_edge_timing_type
force_default_group	force_leakage_if_no_pg_pin
h...	
heartbeat_initial_timeout	heartbeat_timeout
i...	
init_constraint_period	
l...	
library_revision	logic_or
lic_max_timeout	ldb_checkpoint_dir
lic_queue_timeout	library_copyright
logic_and	lvf_constraint_early_late_mode
logic_not	lvf_delay_early_late_mode
m...	
mark_failed_data	measure_slew_upper_fall
mark_failed_data_replacement	measure_slew_upper_rise
max_capacitance_attr_limit	merge_related_preset_clear
max_capacitance_factor	min_capacitance_for_outputs
max_capacitance_limit	min_output_cap
max_transition	min_transition
max_transition_attr_limit	mpw_glitch_peak
max_transition_factor	mpw_input_threshold
measure_cap_lower_fall	mpw_search_bound
measure_cap_lower_rise	mpw_skew_factor
measure_cap_upper_fall	mpw_slew
measure_cap_upper_rise	mpw_slew_clock_factor
measure_output_range	mpw_table
measure_slew_lower_fall	mpw_variation

measure_slew_lower_rise	msg_level
n...	
non_linear_random_variation	nonseq_as_recrem
o...	
output_internal_pin	
p...	
packet_arc_notification_interval	packet_log_filename
packet_arc_notification_limit	packet_mode
packet_arc_notification_list	packet_rdb_count
packet_client_idle_count	packet_rsh_mode
packet_client_resubmit_count	parenthesize_not
packet_client_timeout	parse_auto_define_leafcell
packet_client_timeout_action	predriver_waveform
packet_clients	predriver_waveform_ccs_variation_mode
r...	
rcp_cmd	reset_negative_constraint
rdb_exit_if_source_differ	reset_negative_delay
removal_glitch_peak	resolve_collision
report_detail_variation	rsh_cmd
s...	
scale_load_by_template	skip_nfs_sync
scale_tran_by_template	slew_lower_fall
sdf_cond_prefix	slew_lower_rise
sdf_cond_style	slew_upper_fall
sdf_logic_and	slew_upper_rise
sdf_logic_not	sort_cells
sdf_logic_or	spice_delimiter
set_var_failure_action	supply_define_mode

t...	
tristate_disable_transition	
u...	
user_data_override	
v...	
variation_constraint_path_delta	variation_path_delta_slew_interp_factor
variation_dominant_xtr_ccc_abstol	variation_random_delay_mode
variation_early_late_adjust_mode	variation_random_search_filter_mode_size
variation_ecsm_cap_input_pin	variation_sds_early_late_mode
variation_enable_non_zero_mean	variation_sds_mode
variation_flatten_netlist_mode	variation_sds_samples
variation_mean_nominal_cross_params	variation_sds_sims
variation_mean_nominal_mode	variation_sigma
variation_mean_nominal_include_voltage	variation_static_partition_info
variation_mean_nominal_model_mean_shift	variation_static_partition_mode
variation_mean_nominal_model_mode	variation_static_partition_state_incr
variation_mean_nominal_model_skewness	variation_static_partition_state_max
variation_normalized_ecsm_mode	variation_target_sigma
variation_onesided_voltage	variation_voltage_adjust_pin_supply
variation_onesided_voltage_zero_mode	variation_voltage_variation_use_percent
variation_parallel_mos_mode	variety_netlist_mode
variation_path_delta_no_toggle_probe_marg in_mode	variety_pin_cap_match_liberate
w...	
write_library_mode	

adjust_tristate_load

`<0 | 1 | 2 | 21 | 22>`

Controls whether pin capacitance is added to the load indices on tri-state pins.
Default: `1`

`0`	Turns off these adjustments, that is, the library and template do not add or subtract the tri-state pin capacitance.
`1`	Adds the pin capacitance of the tri-state pin to each of the load indices when outputting the library. The rise index_2 is added to the rise_capacitance and the fall index_2 is added to the fall_capacitance. In addition, when using the `write_template` command to create a Variety Tcl command file, the tri-state pin rise/fall_capacitance is subtracted from the load indices specified in the input library to create the appropriate `define_template` commands for tri-state pins.
`2`	Enables functionality similar to `1` with the addition that instead of adding the rise_capacitance or fall_capacitance, the pin attribute capacitance is added to the load indices for the index_2 values. When using the `write_template` command, the pin capacitance is subtracted from the load indices specified in the input library to create the appropriate `define_template` commands for tri-state pins. The value of the capacitance attribute can be modified using the `set_pin_capacitance` command.
`21`	Provides an effect that is the same as a setting of `1`, but power arc loads are not adjusted.
`22`	Provides an effect that is the same as `2`, but power arcs are not adjusted.

The `adjust_tristate_load` variable can adjust timing arcs, but not power arcs.

Example

```
# Disable adjusting tristate pin load indices
set_var adjust_tristate_load 0
```

aocv_chain_termination_mode

`<0 | 1>` Determines whether to add an extra termination to the last cell in the chain.
Default: `0`

`0`	Does not add an extra termination.
`1`	Adds an extra termination cell to the last cell in the chain.

This variable must be set before the `char_variation` command is run.

aocv_derate_method

`<0 | 1 | 2>` Specifies the algorithm used to compute the early and late derate values. The supported settings are: `0`, `1`, and `2`.
Default: `0`

`0`	`late = ( Tdg + sigma_p ) / Td` `early = 2 - ( Tdg - sigma_n ) / Td`
`1`	`late = ( Tdg + sigma_p) / Td` `early = 2 - late`
`2`	`late = 2 - early` `early = ( Tdg + sigma_n) / Td`

Here:

- Tdg is the delay time at one corner with random variation.
- Td is the delay time at the nominal corner to which the derate will be normalized.
- sigma_p is the positive sigma incremental delay.
- sigma_n is the negative sigma incremental delay.

This variable must be set before the `char_variation` command is run.

aocv_derate_mode

`<"off" | "avg">` Specifies how to merge derates for a cell.
Default: `off`

`off`	Uses min/max to merge different arc derates.
`avg`	Uses average to merge different arc derates for the cell.

This variable must be set before the `char_variation` command is run.

aocv_enable_clock_gater

`<0 | 1>` Enables AOCV analysis for clock-gating cells.
Default: `1` (on)

`0`	Disables AOCV for clock-gating cells.
`1`	Enables AOCV analysis for clock-gating cells (Default).

Important

To enable AOCV generation for clock-gating cells in a library, the cells must include the Liberty attribute `clock_gating_integrated_cell`.

Example

```
library (MyLib) {
    ...
    cell ( MyCell ) {
    clock_gating_integrated_cell : "latch_posedge"; ...
    }
}
```

aocv_extra_driver

<number> Specifies the number of extra drivers to use before the stage where measurement begins.
Default: `0`

The delay calculation starts from the specified stage (for example, the 2nd stage, if *number* = 2) and extends to the Nth stage.

This variable must be set before the char_variation command is run.

aocv_sigma_factor

<*number*>	Specifies a multiplicative factor to apply to the variation. Default: 1

Use this variable to specify a multiplicative factor to apply to the AOCV table data.

If the define_variation commands are specified for a 1-sigma variation, but a 3-sigma variation is required, then set aocv_sigma_factor to 3 to output an AOCV table with 3-sigma variations.

This variable must be set before the create_aocv_table command is run.

binning_detail

<low | medium | high>

Sets the level of detail for state dependency.
Default: high

low	Results in a progressive reduction of the size of the resulting library.
medium	Uses detailed state dependency.
high	Results in a progressive increase of the size of the resulting library.

This variable is used to set the criteria for determining how individual state-dependent groups are merged. This variable affects the characterization and should be set before the char_variation command is run.

Example

```
# Enable low binning detail
set_var binning_detail low
```

bundle_mem_limit

<value>	Specifies the estimated memory-size above which the bundle mode is enabled. Default: `-1` (never use the bundle mode)

This variable is used to enable bundle mode. In this mode, Variety attempts to estimate the required memory to characterize all of the loaded cells. If the estimate exceeds this limit, then Variety separates the loaded cells into multiple Variety runs. Each run includes a number of cells that require less than the `bundle_mem_limit` amount of memory. Use this variable to enable characterization of more cells than can be run in a limited amount of memory, such as in a 32-bit environment. By default, the bundle mode is not enabled.

ccs_abs_tol

<value>	Specifies the CCS absolute tolerance. Default: `1e-13`

Use this control variable to set the CCS absolute tolerance (in seconds). When determining how many points are needed to reproduce the original SPICE waveform, Liberate stops adding points to the CCS data when the absolute error between the reduced CCS waveform and the original SPICE waveform is less than the specified tolerance.

This variable must be set before the `char_variation` command is run.

ccs_base_curve_points

<value>	Specifies the number of base curve points. Default: `15`

Use this control variable to specify the number of base curve points used when generating compact CCS natively in Variety. To output compact CCS format data, use the `write_variation -ccs_compact` option.

This variable can be used after the `char_variation` command.

ccs_base_curve_share_mode

`<0 | 1 | 2>` Determines which algorithm is selected for reusing CCS base curves.
Default: `2`

- `0` Set this variable to `0` to revert to release 2.3p2 and prior release behavior.
- `1` Selects an algorithm that uses a more aggressive base curve re-use rate without impacting accuracy. There is no significant impact on accuracy when using either mode `1` or `2`.
- `2` Selects an algorithm that uses a more aggressive CCS compaction algorithm. There is no significant impact on accuracy when using either mode `1` or `2`.

This variable can be used after the `char_variation` command.

ccs_cap_hidden_pin

`<0 | 1 |2>` Controls the output of the CCS receiver pin capacitance on inputs with hidden power arcs.
Default: `2`

- `0` Uses the behavior of release 2.3p1 and prior where Variety only saved the CCS receiver capacitance on "hidden" pins such as the D pin of a flip-flop.
- `1` Outputs CCS receiver capacitance on input pins that have "hidden" transitions such as clock, clear, preset, combinational_rise, combinational_fall, tristate_enable, and tristate_disable pins.
- `2` Output CCS receiver capacitance on all input pins that have potential "hidden" conditions; *any* pin that has a hidden power arc will also have CCS receiver capacitance. (Default)

This variable must be set before the `char_variation` command is run.

ccs_cap_use_input_transition

`<0 | 1>` Determines the pin transition direction followed to determine the CCS receiver capacitance.
Default: `1`

- `0` Reverts to the old behavior where the CCS receiver capacitance follows the *output* pin direction. Set this variable to `0` after `read_ldb` or `char_variation` and before `write_variation`.
- `1` Instructs Variety to use the *input* pin transition direction for CCS receiver capacitance.

The `read_library` command resets this variable to `0`.

ccs_current_model_pin_load

`<0 | 1>` Instructs Variety to characterize and model the CCS current (`output_current_rise` and `output_current_fall`) tables even when the output has a `pin_load` applied. See also the define_pin_load command and the `-pin_load` and `-load_dir` options of the define_arc command.
Default: `1`

- `0` Instructs Variety to not to characterize and model CCS current if the output has a parasitic load applied.
- `1` Instructs Variety to measure and report the CCS current even if a `define_pin_load` command has been run.

ccs_force_grid_delay

`<0 | 1 | 2>` Controls the accuracy algorithms for checking CCS waveforms.
Default and recommended: `1`

`0`	Does not force CCS waveforms to use the existing grid, so the size of the grid might change if the waveforms require it.
`1`	Forces CCS waveforms to use the existing grid and checks delay accuracy and transition accuracy when segmenting the waveform.
`2`	Forces CCS waveforms to use the existing grid but does not check delay accuracy when segmenting the waveform.

This variable must be set before the `char_variation` command is run.

ccs_init_voltage_comp_thresh

<value>	Sets a voltage threshold to enable a proprietary initial voltage offset. Default: `1` (1V) Set this variable to `-1` to disable the compensation.

The Synopsys CCS standard requires that the CCS waveform start from an initial rail voltage. If the SPICE output waveform starts from a non-rail value (usually due to a relatively large leakage), Variety compensates for the non-rail initial voltage.

Set this variable to a voltage that enables this proprietary compensation algorithm. This algorithm is enabled by default. The Variety CCS init compensation algorithm can adjust the CCS waveform from the initial time up to the time when the output waveform crosses the `ccs_init_voltage_comp_thresh` threshold. By default, the maximum time for the compensation is controlled by the `measure_slew_lower_rise` and `measure_slew_upper_fall` variables. This change can represent a significant change to the CCS waveform.

This variable must be set before the `char_variation` command is run.

ccs_max_current_thresh

<value>	Checks for CCS currents greater than the specified threshold. Default: `0.2A` (200mA) Use MKS units for *<value>*. If the absolute CCS current is larger than the threshold, Variety issues a warning.

This variable must be set before the `char_variation` command is run.

ccs_max_pts

<value>	Sets the maximum number of CCS points that are allowed in the CCS waveform data. Default: `50`

When determining how many points are needed to reproduce the original SPICE waveform, Variety stops adding points to the CCS data when the number of points reaches the limit specified by this variable.

This variable must be set before the `char_variation` command is run.

ccs_rel_tol

<value>	Sets the CCS relative tolerance. Default: `0.001`

When determining how many points are needed to reproduce the original SPICE waveform, Variety stops adding points to the composite current source (CCS) data when the relative error between the reduced CCS waveform and the original SPICE waveform is less than this relative tolerance.

This variable must be set before the `char_variation` command is run.

ccs_voltage_tail_tol

<value>	The stopping point as a ratio of supply of the CCS waveform for modeling purposes. Default: `0.955` (Output must swing to within 4.9% of the supply.)

For CCS timing waveform data, if the tail of the integrated `v(t)` obtained from the sampled `i(t)` does not reach `${ccs_voltage_tail_tol} x supply_swing`, then `i(t)` is padded to ensure that the integrated voltage reaches the supply rail.

As this parameter can cause Liberate to pad the current waveform, it is recommended to set this value lower than `ccs_voltage_tail_trim_tol`, but higher than the requirement of downstream tools (for example, 5% in Synopsys' Library Compiler).

This variable must be set before the `char_variation` command is run.

Example

```
set_var ccs_voltage_tail_tol 0.951
```

ccs_voltage_tail_tol_mode

`<0 | 1 | 2>`	Controls the padding for the CCS "tail". Default: `2`

`0`	Extends the tail as long as possible to reach the `ccs_voltage_tail_tol` value with the last current (`I`) close to 0.
`1`	Extends the tail with limited step. The last current might have a small spike to reach the `ccs_voltage_tail_tol` value.
`2`	Pads the current to ensure the integrated voltage reaches the supply rail tolerance. If the selected pad time is small, this results in a large spike in the current.

This variable determines how to pad the CCS tail if the tail of the integrated `V(t)` obtained from the sampled `I(t)` does not reach `${ccs_voltage_tail_tol} * supply_swing`.

char_effort_systematic_variation

`<0 | 1 | 2 | 3 | 4>`

Controls the effort versus run-time trade-off when characterizing CCS systematic variations.
Default: `3`

The larger the value, the slower and more accurate the characterization. For `char_variation` without `-ccs`, the values `0` and `2` are the same, and `1` and `3` are the same.

combinational_risefall

`<0 | 1>`

Enables characterization of combinational_rise and combinational_fall timing type arcs for preset and clear pins. These arcs occur when both the preset and clear pins are active on a sequential cell and one of them turns off, that is, changes from its active state to its inactive state. See the `merge_related_preset_clear` variable to specify how to model these combinational arcs.
Default: `1` (enabled)

- `0` — Instructs the *Inside View* to characterize only the active arcs on async pins in sequential cells.
- `1` — Characterizes the async arcs for the active edge of the async pins and the combinational arcs on sequential cells.

Note: If combinational arcs for async pins are specified using the `define_arc` command, these arcs are characterized without considering the value of this variable.

This variable must be set before the `char_variation` command is run.

Example

```
# Disable combinational_rise or combinational_fall arcs for preset and clear pins
set_var combinational_risefall 0
```

conditional_cap_hidden_pin

`<0 | 1>`

Enables to characterize and model conditional (state-dependent) CCS/ECSM pin capacitance. This type of pin capacitance is measured during hidden power arc characterization. To get state-dependent pin capacitance, state-dependent hidden_power arcs must be characterized.
Default: `1`

- `0` — Outputs unconditional (state-independent) CCS/ECSM `receiver_cap` for each input pin if the capacitance data was characterized and stored. The variable `ccs_cap_hidden_pin` or `ecsm_cap_hidden_pin` determines if the capacitance is characterized and saved for the hidden arcs.
- `1` — Characterizes and saves all conditional (state-dependent) pin-based receiver capacitance associated with each characterized hidden power arc.

This variable must be set before the `char_variation` command is run.

conditional_cap_hidden_pin_thresh

<value>

Controls the modeling of conditional (state-dependent) receiver capacitance under the input pin. When the number of input pin capacitance with "when" conditions (states) exceeds the specified value, Liberate writes into the `.lib` file two CCS/ECSM `receiver_capacitance` groups, where one is the minimum capacitance and the other is the maximum capacitance. Also, each group will have a `char_when` attribute (but no `when` condition). This is done to minimize the size of the output library.
Default: `1024`

The minimum or maximum capacitance table is determined by finding the capacitance value from all conditional receiver capacitance tables and selecting the table that contains the capacitance. If the minimum and maximum capacitance groups are equal, then only one capacitance group is output.

This variable must be set before the `char_variation` command is run.

conditional_constraint

`< 0 | 1 | 2 >` Controls the conditional constraint characterization for timing constraints.
Default: `1`

- `0` Timing constraints are characterized under worst-case conditions.
- `1` Characterize and model each unique when condition.

  Note: Using this value increases the run time.
- `2` The worst-case constraint condition is characterized similar to 0, but a combined `when` condition is generated in the library. If `conditional_arc` is set to `0`, `conditional_constraint` will affect characterization, but only one state-independent timing group will be stored.

This variable is used to enable characterization of conditional (state-dependent) timing constraints including setup, hold, recovery, removal and minimum pulse width.

This variable must be set before the `char_variation` command is run.

Example

```
# Enable conditional arcs for timing constraints
set_var conditional_constraint 1
```

conditional_expression

`<merge | separate>` Controls whether conditional groups are merged when they contain sub-expressions that are OR'd together.
Default: `merge`

`merge`	Merges all conditions that result in identical values by ORing each of the sub-expressions.
`separate`	Splits conditions containing OR sub-expressions into separate groups.

conditional_include_constant

`<0 | 1>` Controls whether constant nets are included in conditional `when` statements.
Default: `1` (on)

`0`	Constant nets are not included in conditional `when` statements.
`1`	Constant nets are included in conditional `when` statements.

Examples

```
# Enable conditional arcs for timing constraints
set_var conditional_constraint 1
# Enable conditional arcs for timing constraints
set_var conditional_expression separate
```

constraint_check_rebound

`<0 | 1>` Enables the output rebound checks.
Default and recommended: `0`

`0`	Does not enable the output rebound check.
`1`	Enables checking, during constraint search for degradation, for the first and last threshold crossing to see if they are different by more than 1ps. If they are, that is considered a failure.

This variable must be set before the `char_variation` command is run.

constraint_clock_gater

`<0 | 1>` Controls the use of special clock gater circuit constraints.
Default: `1`

- `0` Disables special handling for clock gater constraints.
- `1` Enables special handling for clock gater constraints.

Clock gating circuits require special techniques when measuring constraints such as setup and hold. These techniques are documented in the *Timing Constraints* section of the *Virtuoso Liberate Reference Manual*.

constraint_delay_degrade

<value> Specifies the percentage of delay degradation permitted in the clock-to-constraint-output-pin delay (flip-flop) or the data-to-constraint-output-pin delay (latch) before an arriving signal is deemed to fail a timing constraint (setup, hold, recovery, removal).
Default: `0.1` (10%)

constraint_delay_degrade_abstol

<delay> Specifies the minimum delay degradation value (in seconds) permitted in the clock-to-constraint-output-pin delay (flip-flop) or the data-to-constraint-output-pin delay (latch). The maximum of the `constraint_delay_degrade` percentage of the clock-or-data-to-output delay and the `constraint_delay_degrade_abstol` is used as the delay degradation criteria.
Default: `2e-12` (2ps)

constraint_failed_value

<value> Inserts the value of this variable into the LDB and resulting library.
Default: `1` (`1e9` in Liberate LDB units of nS)

`-1`	Reverts to the 2.5 (and prior) behavior where, for example, if the constraint arc search simulation failed with a “too close to search bounds” error, the resulting LDB and library contained the failing bound of the constraint search. In rare cases, this could be a large negative value, which resulted in an optimistic constraint value in the library.
`<value>`	Specifies a value to be inserted into the LDB and resulting library when a constraint characterization fails (as when the result is too close to search bounds). The units of this value is in LDB time units of `1e-9S`.

This variable must be set before the `char_variation` command is run.

constraint_glitch_hold

`<0 | 1>` Enables glitch height as the failure criteria for characterizing hold time constraints.
Default: `0`

`0`	Use delay degradation as the failure criteria.
`1`	Use glitch height as the failure criteria.

constraint_glitch_peak

`<value>` Specifies the maximum size of logic glitch permitted on the constraint output pin before an arriving signal is deemed to fail a timing constraint (setup, hold, recovery, removal).
Default: `0.1` (10%)

constraint_hold_probe

`<value>`

Specifies the name of the cell probe pin to use for hold timing constraint characterization.
Default: `''` (Use the name specified by the `constraint_output_pin` variable.)

The `<value>` can also be an internal node specified using the form `<transistor_name>:[S|D|G]` where `S` is the transistor source, `D` is the transistor drain, and `G` is the transistor gate terminal.

Example

```
# Set the output pin name for constraints
set_var constraint_hold_probe ProbePin
```

constraint_info

`< 0 | 1 | 2 >`

Enables printing of constraint measurement details.
Default: `1`
Recommended: `2`

`0` — Does not print any constraint characterization details.

`1` — Turns on printing of constraint characterization details (including the probe point) into the log file. The information printed includes the cell name, timing type, constraint type, pin, related pin, probe node, and criteria. The information is printed prior to the actual characterization and is therefore based on the characterization plan.

`2` — Turns on printing of constraint mapping information after a cell is finished with characterization. Constraint criteria can be updated during simulation. This is the recommended setting.

Example

The following is an example of the log file output:

```
constraint_map: Cell: CLKGater; Type: setup_rising; Constraint: rise_constraint;
Pin: EN; Related: CLK; probe:EN->I2:26; criteria: degradation
```

constraint_linear_waveform

`<value>` Controls whether a linear waveform is used to drive the inputs during constraint characterization.
Default: `0`

`0`	Uses the same driver that delay and transition use.
`1`	Requests a linear waveform for constraint characterization.

constraint_merge_state

`<0 | 1>` Controls whether timing groups are merged when the `when` states are the same. This can occur when there are multiple user-defined constraint arcs with the same `when` to be characterized using different metrics or probes.
Default: `1`

`0`	Uses the release 2.4 and prior behavior.
`1`	Merges these arcs using the bitwise worst case among the values.

This command must be set before the `char_variation` command is run.

constraint_output_load

`<min | max>` Sets the type of load on the output pin used for constraint characterization.
Default: `min`

`min`	Uses the minimum load index value.
`max`	Uses the maximum load index value.

constraint_output_load_factor

<value> — Specifies the factor to be used to scale the output load on the output pin used for constraint characterization.
Default: `1.0`

constraint_output_pin

<pin> — Specifies the name of the output pin used for timing constraint characterization (determining setup, hold, recovery, and removal values).
Default: `""` (the alphabetic first output pin name)

The *<pin>* can also be an internal node specified using the form *<transistor_name>*`:[S|D|G]` where `S` is the transistor source, `D` the transistor drain, and `G` the transistor gate terminal.

When calculating constraints, a search is performed by switching the relevant data signal with respect to the clock signal and determining when there is significant delay or voltage impact at a particular pin or node of the cell. This variable defines which of the pins of the sequential cell to monitor.

Examples

```
# Set the output pin name for constraints
set_var constraint_output_pin Q
define_cell \
    -input {D} \
    -output {Q QN} \
    -clock {CK} \
    -delay delay_5x5 \
    -constraint constraint_3x3 \
    DFFX1
# Set a pin name for constraints
set_var constraint_output_pin M1:D
```

constraint_random_variation_search_time_abstol

<value> — Specifies the minimum desired search time for random variation characterization.
Default: `1e-12` (in seconds = `1pS`)

Set this variable to the time in seconds, representing the minimum constraint binary search window for random sensitivity characterization. A smaller minimum search absolute tolerance limit will increase the characterization resolution at the expense of run time. An increase in resolution might not represent a significant increase in accuracy. This variable should be set with care to ensure that the accuracy improves to an acceptable level without an unacceptable increase in the run time.

The recommended setting for 16nm or smaller processes is `0.1e-12`.

This variable must be set before the `char_variation` command is run.

constraint_slew_degrade

<value>	Specifies the value (0.0 to 1.0) that represents the percentage of delay degradation. Includes slew degradation percentage along with delay degradation as a criteria for calculating timing constraints. Default: use only delay degradation

This variable is used to include slew degradation when determining timing constraints. By setting the slew criteria both delay degradation and slew degradation are checked and the first criteria to fail determines the setup and hold values. The slew degradation is measured using the `measure_slew_lower_fall`, `measure_slew_lower_rise`, `measure_slew_upper_fall`, and `measure_slew_upper_rise` variables.

Examples

```
# Set the setup time delay degradation criteria to 12%
set_var constraint_delay_degrade 0.12

# Set the minimum delay degradation criteria to 10ps
set_var constraint_delay_degrade_abstol 1e-11

# Set the slew degradation criteria to 50%
set_var constraint_slew_degrade 0.5

# Set the hold time glitch peak criteria to 15% of Vdd
set_var constraint_glitch_peak 0.15
set_var constraint_glitch_hold 1

# Set the output load to max and scale by 0.5
set_var constraint_output_load max
set_var constraint_output_load_factor 0.5
```

constraint_search_bound

<min_time>	Specifies the minimum search bound. Default: -1

Use this variable to set the initial constraint search bounds. The constraint bisection search will start at a maximum of +/- the search bound value. By default, Liberate automatically determines the search bound for optimal run time. When this variable is set to a time (in seconds), Liberate starts the bisection search using +/- the constraint_search_bound value that will override the automatically-determined bound value.

When constraint estimation is not done (see constraint_search_bound_estimation_mode). This variable determines the range of constraint values to search. The default is 10ns. The search range is +/- the minimum of constraint_search_bound (or 10ns) and 0.5*sim_duration, adjusted to allow for the constrained and related pin slews.

This variable must be set before the char_variation command is run.

constraint_search_bound_estimation_mode

<0 | 1 | 2 | 3>	Controls the method used to determine the constraint search bound. Default: 2

0	Use the method prior to release 3.1.
1	Use the method prior to release 12.1.4.
2	Use a method that is consistent for all settings of extsim_model_include and extsim_exclusive.
3	Disable search bound estimation (see constraint_search_bound).

Note: It is recommended not to change this parameter except for backwards-compatibility purposes.

This variable must be set before the char_variation command is run.

constraint_search_time_abstol

<min_time>	Set this variable to the minimum constraint binary search window. Default: `2e-12` (seconds)

This variable affects characterization and should be set prior to the `char_variation` command. If *<min_time>* is set too small, this variable can have a significant impact on the run time.

constraint_snap_to_bound

`< 0 | 1 >`	Determines which constraint characterization algorithm to use to snap to the passing bound. Default: `1` (Report the last passing bound)

`0`	Uses a proprietary algorithm to determine the final constraint value.
`1`	Enables Variety bisection-based constraint characterization to snap to the last passing bound while characterizing constraints when processing a binary search. This can result in a bit more pessimism and run time. By reporting the last passing bound, you can be assured that the reported constraint value was simulated by Variety.

This variable must be set before the `char_variation` command is run.

constraint_tran_end_extend

<value>	Specifies an absolute incremental increase (in seconds) in the transient simulation end time for constraints. This allows you to increase the `.tran` end time by an arbitrary time if the automated `.tran` end time is not sufficient for the simulation measurements to return proper values. Default: `500e-12` seconds

This variable specifies a duration in seconds to add to the transient simulation end time (see `.tran` in a SPICE format simulation deck). Variety checks if a transition on the constraint

probe occurs within `constraint_tran_end_extend/2` of the transient simulation end time. This check is performed to detect the possibility that a transition degrading beyond the `.tran` end could be mistaken for a failed constraint bisection iteration. If such a case is detected, a warning is generated that suggests increasing the value of this variable. It is important that the transient simulation does not terminate early because for glitch-based constraints, the reported constraint value might be optimistic while for delay degradation, the reported constraint might result in a pessimistic value.

This variable must be set before the `char_variation` command is run.

constraint_vector_mode

`< 0 | 1 >`

Specifies the mode for controlling recovery, removal, and MPW vectors when using the `define_arc` command.
Default and recommended: `2`

`0`	Liberate uses the pre-3.0p3 algorithm for vector generation.
`1`	If a `define_arc` is supplied for recovery, removal, or min_pulse_width types, Liberate uses an enhanced algorithm that increases the vector generation effort.
2	Enables the probing of internal nodes for all types of constraints/mpw when default probing fails to find a vector. This provides more automation in a recharacterization flow (especially for PMK) where certain arcs cannot be acquired without specifying the internal probe node. Examples include MPW on SAVE pin, and non_seq constraints between RDN and RETN pins.

This variable must be set before the `char_variation` command is run.

debug_flow

<1x1 | 2x2>	Speeds up the characterization by reducing the template as defined by the `define_template` command to either a `1x1` or a `2x2` data matrix. This can significantly decrease the run time for a quick debug flow. You can also check the selected points in a slew or load matrix by shrinking the template to a `1x1` or a `2x2` data matrix.
	`1×1` Uses the first value in each index.
	`2×2` Uses the first and last values in each index.

This variable must be set before the first `define_template` command is run.

Example

```
set_var debug_flow 2x2
```

default_capacitance

<min | avg | max>	Determines which rise/fall capacitance value is used as the default pin rise/fall capacitance. Default: `max`
	`min` Uses the minimum rise/fall capacitance value.
	`avg` Uses the average rise/fall capacitance value.
	`max` Uses the maximum rise/fall capacitance value.

default_group_method

<0 | 1>	Determines which method to use for creating the default groups. Default: `0`
	`0` Uses the worst value (min or max) from all relevant tables.
	`1` Uses the table that has the worst value for the first table entry (e.g. min slew/min load).

default_unateness

`<merge | separate>` Controls whether default *positive_unate* and *negative_unate* timing groups are merged into a single *non_unate* default timing group.
Default: `merge`

- `merge` — If both *positive_unate* and *negative_unate* timing groups exist, they are merged into a single non_unate default group.
- `separate` — Keeps *positive_unate* and *negative_unate* timing groups from being merged in the default group.

Note: If all timing sense attributes are identical, then the original timing sense remains unchanged during merging. If there are both *positive_unate* and *negative_unate* timing sense group data, then the merged timing sense is *non_unate*.

This variable typically has an impact on complex combination cells such as adders.

Example

```
# Disable merging for binate default groups
set_var default_unateness separate
```

define_arc_merge_state

`<0 | 2 | 3>` Improves the performance of Variety by recognizing the electrically-equivalent `when` conditions. This is especially useful in the verbose template flow.
Default: `0` (off)

- `0` — Disables recognition of electrically-equivalent `when` conditions.
- `2` — Enables recognition of electrically-equivalent `when` conditions. This results in improved performance.

`3`	Merges all define_arc commands on an arc basis effectively reducing the user-defined `when` conditions and possibly reducing the accuracy. This is the fastest and least accurate setting.

When this variable is set to `2`, Variety automatically sets `ccsn_compatibility_mode` to `0` and `conditional_expression` to `separate`. If this represents a change in the value of `conditional_expression`, the structure of the output library might change.

This variable must be set before the `char_variation` command is run.

delay_inp_fall

<value>	Specifies the % point on the cell input falling waveform from which to measure delays. Default: `0.5` (50% of supply)

delay_inp_rise

<value>	Specifies the % point on the cell input rising waveform from which to measure delays. Default: `0.5` (50% of supply)

delay_out_fall

<value>	Specifies the % point on the cell output falling waveform from which to measure delays. Default: `0.5` (50% of supply)

delay_out_rise

<value>	Specifies the % point on the cell output rising waveform from which to measure delays. Default: `0.5` (50% of supply)

These variables set the input and output rising and falling transition crossing points for measuring delays.

Examples

```
# Set the delay measurement from 45% to 55%
set_var delay_inp_fall 0.45
set_var delay_inp_rise 0.45
set_var delay_out_fall 0.55
set_var delay_out_rise 0.55
```

derate_comment_start_str

`"string"`	Specifies the string that demarcates the beginning of a comment containing details about the characterization results of the AOCV derate data. Default: `""` (Do not output the details)

derate_comment_end_str

`"string"`	Specifies the string that demarcates the end of the comment containing details about the characterization results of the AOCV derate data. Default: `""` (Do not output the details)

The `derate_comment_start_str` variable specifies the characters needed to begin a string in the AOCV table. When this is set, Variety begins a comment and adds to it the details that describe the characterization results of the AOCV derate data. The `derate_comment_end_str` variable specifies how the comment is terminated.

These variables must be set before the `write_variation_table` or `create_aocv_table` command is run.

Example:

```
read_ldb char.ldb.gz
set_var derate_comment_start_str "/*"
set_var derate_comment_end_str "*/"
write_variation_table -chain_length 3 -cells $cells -filename
new_data/read4.lib myLibrary
```

The output file will have data as illustrated below.

```
Cell MUX2
Early-Rise
Stage  1  2  3
```

```
distance
/* Mean_delay: 0.0328576, Sigma_delay: -0.0310776, load: 0.003429, slew: 0.37949,
Arc: B r -> Z r [When: ""] */
1.94583  1.6688  1.54607
.......
```

disable_method

`<0 | 1>` — Controls how disabled timing arcs are calculated, using the threshold voltage or Vdd/2.
Default: `0`

`0`	Calculates the disable timing arcs (three_state_disable) by measuring the delay time from the input disable pin to the time the input signal to the last transistor stage (channel connected block) crosses the lowest threshold voltage (V_{th}) of the impacted transistors.
`1`	Considers the output disabled when the input voltage to the last stage reaches half the supply voltage (Vdd/2).

Example

```
# Merge disable method to Vdd/2
set_var disable_method 1
```

extsim_ccs_option

`"options"` — Specifies the list of options to be used by the external SPICE simulator when characterizing CCS timing.
Default: set to the `<options>` set by the `extsim_option` variable

This variable applies only when the `-ccs` option is used with the `char_variation` command. For accurate CCS current measurements, make sure you are using an accurate simulator setting (e.g. `"runlvl=6"` for HSPICE). This variable also impacts NLDM delay, transition, and ECSM waveform values. The *options* string is passed as a `.option` line in the SPICE decks that Variety creates for characterization.

extsim_cmd

<string>	Specifies a command string to be used to call the external SPICE simulator. This variable can be used to override the default command used by Variety to call the external simulator. Default: see the information given below

This variable can be used to override the default command used by Variety to call the external simulator. The default settings are:

- For HSPICE: `extsim_cmd="hspice"`

```
$extsim_cmd $extsim_cmd_option -i $spicedir/sim.sp \
    -o $spicedir/sim >& /dev/null
```

- For Spectre: `extsim_cmd="spectre"`

```
$extsim_cmd $extsim_cmd_option sim.sp >& sim.lis
```

Examples

```
set_var extsim_cmd "spectre2"
set_var extsim_cmd_option "+log mylogfile"
```

File `spectre2` contains:

```
#!/bin/sh
exec spectre $*
```

extsim_cmd_option

"*string*"	Specifies options to be passed to the external simulator.

This variable can be used to override the default command arguments used by Variety to call the external simulator. The default settings are:

- For HSPICE:

```
extsim_cmd_option=""
```

- For Spectre:

```
extsim_cmd_option=""
```

extsim_deck_dir

<directory_name> Specifies a directory in which to save all of the SPICE decks created when using an external simulator for characterization.

The directory must be visible to all of the server and client machines being used. If the directory does not exist, it is created.

All of the decks are written to a sub-directory based on cell names within the named directory. The files in the directory use the following naming convention:

*<cell>*_*<pin>*_*<dir>*_*<relPin>*_*<relPinDir>*_*<type>*_*<iter#>*.sp

Because the number of simulation decks is very large it is advisable to use this variable only when characterizing a small number of cells.

Example

```
# Set the directory for storing SPICE decks
set_var extsim_deck_dir "/home/char/decks"
char_variation -extsim hspice -cells {INVX1 DFFX1}
```

extsim_deck_header

"*options*" Specifies a string of SPICE commands that are used to overwrite the headers of the external SPICE simulation decks. Default: for HSPICE, `".protect"`

extsim_deck_style

<merge | separate> Controls whether the netlist and models from the external simulator SPICE decks are saved into a separate file. Default: `merge`

`merge` Keeps the netlist and models in-line.

`separate`	Separates the netlist and the models from the external simulator SPICE decks. The netlist and models are saved into a separate file and loaded using the `.include` SPICE command.

extsim_exclusive

`<0 | 1>` Controls the SPICE engine used for pre-simulation.
Default and recommended: `1`

`0`	Uses Alspice for pre-simulation measurements.
`1`	Uses the external simulation engine for all simulations when `char_variation -extsim` is selected.

This variable is intended for use in cases where Liberate Alspice cannot support the process model, netlists, or both (such as when encrypted netlists or models, or Verilog-A models are used). In these cases, the `extsim_model_include` variable and the `define_leafcell` command should first be set properly. Note that using the `define_leafcell` command sets `extsim_exclusive` to 1. In cases where the netlist cannot be processed, the `extsim_flatten_netlist` variable might also need to be reset to 0.

This variable must be set before the `char_variation` command is run.

extsim_lic_keep

`<0 | 1>` Enables the HSPICE `-cc` mode of operation.
Default: `0`
Recommended: Set this variable is `1` if you are using a 2008.09 or newer version of HSPICE. If you are using a version of HSPICE that does not support keeping a license checked out, do not use this variable.

`0`	Disables the HSPICE `-cc` mode of operation.

`1`	Enables the HSPICE `-cc` mode of operation. This setting can significantly improve the run time when using HSPICE as the external simulator.

The external simulator must be enabled with the `char_variation -extsim` option.

This variable must be set before the `char_variation` command is run.

Example

```
set_var extsim_lic_keep 1
```

extsim_mc_append

"*string*"	Adds the specified string to the `.tran` simulation command in the extsim deck. Default: ""

Set this variable to a Monte Carlo compatible transient analysis option. This variable can be used when the Monte Carlo simulation algorithm is requested (see `char_variation -monte`) to add an arbitrary string to the `.tran` simulation command. It is currently supported only when Spectre is used as the external simulator.

This variable must be set before the `char_variation` command is run.

Example

```
set_var extsim_mc_append "dcop1 dc"
```

extsim_mc_type

"*string*"	Sets the Monte Carlo analysis type. Default: ""

This variable is currently required and supported only when you are using Spectre. For more information, see `char_variation -extsim spectre`.

Currently, the only supported value is `"mismatch"`. For information on enabling Monte Carlo analysis, see `char_variation -monte`.

This variable must be set before the `char_variation` command is run.

Example

```
set_var extsim_mc_type "mismatch"
```

extsim_model_include

<value>

Specifies a full path to a file that loads the SPICE models when using an external SPICE simulator.
Default: Uses flattened models in the external simulation input decks

When this variable is used, Variety uses the file specified instead of the flattened models in the external simulation input deck. Variety places a statement in the extsim SPICE decks such as:

.include <*extsim_model_include_file*>

This variable must be set before the `char_variation` command is run.

Example

```
set_var extsim_model_include "/home/user1/models/include_ff"
```

Where `include_ff` contains:

```
.include '/home/user1/models/models.l' ff
```

extsim_model_include_mode

`<0 | 1 | 2>`

Instructs the tool to add a1e-5 Ohm resistor to all internal cell nodes.
Default: `1`
Recommended: `2`

`0` Adds a 1e-5 Ohm resistor to all internal nodes of a cell. Uses the behavior of 2.4p1 and earlier releases of the tool.

`1`	Does not add a resistor for leakage. The `.nodeset` and `.ic` statements in the extsim SPICE decks for *leakage* reference internal nodes directly instead of through a 1e-5 Ohm resistor.
`2`	The `.nodeset` and `.ic` statements in *all* extsim SPICE decks reference internal nodes directly.

To assist the extsim SPICE engine (see `char_variation -extsim`) with DC convergence, Liberate adds a 1e-5 Ohm resistor to internal nodes of a cell. With some versions of SPICE, this can cause DC convergence issues with leakage simulations.

extsim_monte_option

<string>	Specifies the string that is passed as a `.option` line in the external MONTE decks that Variety creates.

Examples

```
# Set the sampling methodology for Monte Carlo
set_var extsim_monte_option "sampling=lhs"
```

extsim_option

"options"	Specifies the list of options to be used by external SPICE characterization for delay or timing constraint characterization. Default: for HSPICE, `"runlvl=5"`; and for Spectre, `"save=none"`

The `options` string is passed as a `.option` line in the external SPICE decks that Variety creates.

Examples

```
# Set the .options for external SPICE and CCS
set_var extsim_option "runlvl=5"
set_var extsim_ccs_option "runlvl=6 bypass=0"
```

extsim_option_presim

"*options*"	Specifies a list of options to be used for characterization with external SPICE. Default: "" (null)

When checking a user-defined arc (UDA) with an external simulator (`char_variation -extsim`), Liberate creates a deck used for checking if the arc can be simulated. This deck is called a “CHK” deck and uses faster simulator option settings. Sometimes, different options should be used to help DC convergence. Use this variable to specify simulator options to be used during the presim stage.

Example

```
set_var extsim_option_presim "ITL1=300"
```

This variable must be set before the `char_variation` command is run.

extsim_reuse_ic

<0 | 1 | 2 | 3>	Controls whether the first set of DC solutions in a group of `.alters` are reused during simulations when using Spectre as the external simulator. Default: 3

0	Do not reuse DC solutions during `.alter` simulations.
1	Reuses DC solution by adding `write/readns` to the `.tran` statement.
2	Reuses DC solution by adding the `restart=no` option to the `.tran` statement. This allows the previous solution to be uses as the IC for `.alter` simulations. However, you may not want this because if you run two subsequent transients, the last time point of the first transient would be the IC for the second transient. Therefore, this setting is not recommended.

`3` — Reuses DC solution by adding the `skipdc=useprevic` option to the `.tran` statement. (Recommended and the only supported option for Spectre-SKI.)

This variable must be set before the `char_variation` command is run.

extsim_sanitize_param_name

`<0 | 1>` — Enables the cleaning of net or port names that use the characters < and > in the output SPICE deck.
Default: `0`

`0` — Does not clean node names. This might lead to early external SPICE termination.

`1` — Sanitizes net or port names containing the characters < and > before using the name in the SPICE deck in a constructed `.param` name, in `.data`, or in a PWL list.

The < and > characters could cause the port or net to be interpreted incorrectly as an equation when using an external simulator.

This variable must be set before the `char_variation` command is run.

extsim_save_driver

`<0 | 1>` — Saves SPICE decks for failing active driver simulations.
Default: `1`

This variable is used to enable the saving of simulation decks used by the `set_driver_cell` command and characterize the active driver output waveform simulation decks.

`0` — Prevents active driver simulation decks from being saved. This option is available for backward compatibility.

`1`	Saves the simulation decks used for the final driver waveforms. This option works in combination with the `extsim_deck_dir`, `extsim_save_passed`, and `extsim_save_failed` variables. If the saving of decks is not enabled using these variables, no decks are saved.

This variable must be set before the `char_variation` command is run.

extsim_save_failed

`<none | deck | all>` Controls the saving of input SPICE decks and output listings for failing simulations.
Default: `all`

`none`	Neither decks nor listings are saved.
`deck`	Saves only the input SPICE deck.
`all`	Saves both the input deck and the output listings.

The data is saved in the directory defined by the `extsim_deck_dir` variable.

extsim_save_passed

`<none | deck | all>` Controls the saving of input SPICE decks and output listings for successful simulations.
Default: `none`

`none`	Neither decks nor listings are saved.
`deck`	Saves only the input SPICE deck.
`all`	Saves both the input deck and the output listings.

The data is saved in the directory defined by the `extsim_deck_dir` variable.

extsim_tar_cmd

"*string*"	Specifies the command to be used to compress the output SPICE decks. Default: `tar zcf` Set this variable to the NULL string (`""`) to disable compression.

Example

```
# Disable SPICE deck compression
set_var extsim_tar_cmd ""
```

extsim_timestep

<*value*>	Specifies the time step to be used for the external SPICE simulation. Default: `1e-12s` (1ps)

Example

```
# Set the time step for external SPICE
set_var extsim_timestep 2e-12
```

extsim_tran_append

"*options*"	Specifies a list of additional options to append to the `.tran` statement.

Example

```
# Set conservative mode for Spectre
set_var extsim_tran_append "errpreset=conservative"
```

extsim_use_node_name

<0 | 1>	Controls whether node names should be mapped to numbers in extsim SPICE decks. Default and recommended: `1`

`0`	Maps node names to numbers in the extsim SPICE decks.
`1`	Disables node names to number mapping in the extsim SPICE decks and requests real node names to be used instead.

extsim_variation

`<0 | 1>` Controls whether the external SPICE simulation engine is used for characterizing sensitivity.
Default: `1`

`0`	Sensitivity variation characterization uses the internal Alspice engine.
`1`	Sensitivity characterization follows the setting of the `char_variation -extsim` option.

force_condition

`<0 | 1 | 3>` Controls whether to include conditional `when` and `sdf_cond` attributes for single or binate timing groups.
Default: `1`

`0`	Turns off the `when` and `sdf_cond` attributes for arcs with a single or binate timing group.
`1`	Outputs `when` or `sdf_cond` conditions on timing groups even if there is only a single timing group for that arc. This enables less pessimistic timing simulation with back annotated delays (SDF).
`3`	Outputs conditional delays for clock to output arcs. With many sequential cells, the clock to Q path contains many states. Some of these states contain only rise or fall arcs while others contain both rise and fall arcs. Variety models this arc as a single worst-case unconditional arc.

This variable must be set before the `char_variation` command is run.

Example

```
# Turn off conditions on single timing groups
set_var force_condition 0
```

force_default_group

`<0 | 1>` Controls whether to always include a default group.
Default: `0`

`0`	Does not always output a default group.
`1`	Always outputs a default group, even if the group is redundant. This can occur when all states are exhaustively enumerated.

This variable may be used after the `char_variation` command.

Example

```
# Require output of default group
set_var force_default_group 1
```

force_edge_timing_type

`<0 | 1>` Controls whether single-sided edge triggered arcs are modeled as rising_edge, falling_edge or as combinational_rise, combinational_fall.
Default: `1`
Recommended: Set this variable to `1` when using 2009 and later versions of LC, the Synopsys Library Compiler.

`0`	Outputs edge timing types (rising_edge or falling_edge) only when the timing group has both rising and falling data. If only one direction of data exists, the timing type is converted to a timing type of combinational_rise or combinational_fall.

`1`	Outputs a timing type of rising_edge or falling_edge for single-sided edge triggered arcs (for example, a clock to output arc in a clock gating cell).

This timing type change was implemented because early versions of Synopsys Library Compiler did not permit one-sided "edge" timing types, but the more recent versions support this.

This variable must be set before the `char_variation` command is run.

force_leakage_if_no_pg_pin

`<0 | 1>` Controls whether cells, such as antenna cells that do not have related_pg_pins for leakage power, are forced to generate a leakage_power_group whenever pin_based_leakage is set. Default: `1`

`0`	Leakage groups are not output if there is no related_pg_pin, which is the pre-3.0 behavior. The resultant library might not be compliant with Synopsys Library Compiler.
`1`	Forces cells, such as antenna cells, that do not have related_pg_pins for leakage power to generate a leakage_power_group whenever pin_based_leakage is set.

The `force_leakage_if_no_pg_pin` variable only works with pin_based_leakage and it only affects cells without related_pg_pins for leakage power.

To be compliant with Synopsys Library Compiler, the `cell_leakage_power` global attribute for a cell is output to the library, without regard for the value of the `cell_leakage_power` variable. The default leakage power for that cell is not output to the library, without regard for the value of the `keep_default_leakage_group` variable.

This variable must be set before the `write_variation` command is run.

heartbeat_initial_timeout

<time>	Specifies the time, in seconds, that the server waits for the first client to communicate back to the server. Default: `3600` (1 hour)

When *time* is exceeded, the Variety server issues a warning that the client has failed to start and then restarts the `heartbeat_initial_timeout` timer. This situation could occur, for example, due to network problems.

Example

```
# Set the heartbeat initial timeout to 2 hours
set_var heartbeat_initial_timeout 7200
```

heartbeat_timeout

<time>	Specifies the time, in seconds, that a client machine can be inactive before being released by the server machine. Default: `300` (5 minutes)

This variable enables recovery from machine failures during distributed validation. It controls how long the server machine waits for a response from a client before releasing that client. If a client hangs, it is not used for the remainder of the validation run. In addition, the task (a collection of arc simulations) being performed by the failing client is re-submitted to another client. If the re-submission of this task causes another client to hang then this task is skipped. At the end of the validation run, any cells that are not fully simulated are reported.

Example

```
# Set the heartbeat timeout to 10 minutes
set_var heartbeat_timeout 600
```

init_constraint_period

<value>	Specifies the clock period, in seconds, used for circuit initialization during constraint characterization. Default: `-1` (no initial clock pulse is used)

A clock pulse with a 50% duty cycle is used to initialize the circuit before constraint circuit simulation.

Example

```
set_var init_clock_period 5e-9
```

ldb_checkpoint_dir

<directory>	Specifies the directory where the LDB checkpoint file is to be stored. Default: "." (the directory where the initial run was started)

Variety stores the characterization data in a temporary LDB (library database) file called `altos.ldb.<`*PID*`>.gz` where *PID* is the process ID. This variable is used to specify the directory where the temporary LDB checkpoint file is stored.

Example

```
set_var ldb_checkpoint_dir /home/work/rundir
```

library_copyright

<"copyright">	Specifies the string to denote copyright in the output library.

Example

```
# Set the copyright line
set_var library_copyright "Cadence, 2010-2013"
```

library_revision

<"revision">	Specifies the string to use for denoting the library revision in the output library.

Example

```
# Set the revision to 2.0
set_var library_revision "2.0"
```

lic_max_timeout

<value>	Specifies the amount of time, in seconds, to wait for the required licenses to be acquired. Default: `86400` (seconds)

When starting up, Variety attempts to check out all the licenses that are needed. For a server, 1 server license is needed. For a client, Variety needs 1 client license for each thread (see `char_variation -thread`). If `ALTOS_QUEUE` is set and only one license is needed, Variety waits until a license is available and then starts running. If `ALTOS_QUEUE` is set and more than 1 license is needed, Variety waits until the timeout or until it has all of the licenses it needs for all threads.

When the timeout ends, if Variety has at least 1 license, it stops waiting and starts execution with however many licenses it has. If Variety has no licenses at the end of the timeout, it resets the timeout clock and begins waiting again for licenses. After execution begins, Variety stops looking for additional licenses.

For example, if `ALTOS_QUEUE` is set to 1 along with `char_variation -thread 4`, and if there are only 2 (mix-and-matched `Liberate_Client` and `Variety_LX_Client` licenses available at the beginning, then Variety remains in a wait-and-check queue for an additional 2 licenses. As soon as the additional 2 client licenses are checked out successfully, Variety starts execution with 4 simulation threads. If, however, there are no additional licenses checked out at the end of the timeout, Variety starts execution with only 2 simulation threads.

If `ALTOS_QUEUE` is set to 0 or is not set, Variety does not wait for licenses. Instead, It checks out as many licenses as it can (not exceeding the number it needs) and begins execution. If no licenses are available, Variety terminates.

The shell environment variable ALTOS_LIC_MAX_TIMEOUT will override the value set by this variable in the Tcl file. For more information, see ALTOS_LIC_MAX_TIMEOUT.

lic_queue_timeout

`<value>`	Specifies the amount of time, in seconds, to wait for the required licenses to be acquired. Default: `60` (seconds)

The ALTOS_LIC_CHECK_ALT_TIMEOUT shell environment variable can be used to override the value set by this variable in the Tcl file.

logic_and

"*string*" — Specifies the characters to use for denoting logic AND in library attributes that contain logical functions such as `when` conditions.
Default: " * " (*Notice the space on both sides of* *)

Example

```
# Set the logic AND to &&.
set_var logic_and "&&"
```

logic_not

"*string*" — Specifies the characters to use for denoting logic NOT in library attributes that contain logical functions such as `when` conditions.
Default: "!"

Example

```
# Set the logic NOT string to ~
set_var logic_not "~"
```

logic_or

"*string*" — Specifies the characters to use for denoting logic OR in library attributes that contain logical functions such as `when` conditions.
Default: " + " (*Notice the space on both sides of +*)

Example

```
# Set the logic OR string to |
set_var logic_or "|"
```

lvf_constraint_early_late_mode

`<0 | 1 | 2>`

Controls how the constraint OCV sigma tables should be written to the LVF file.
Default: `0`
Recommended: `2`

`0`

Writes one OCV sigma constraint group without `sigma_type` to the output LVF file. First compute the effective sigma of `early` and `late` across all parameters. Then, average the `early` and `late` sigma data. The `early` and `late` sigma data must be characterized (see non_linear_random_variation). Assume there are two characterized parameters `p1` and `p2` each with `early` and `late` data. Then:

1. Compute the effective `early` and effective `late` sigma across `p1` and `p2`. There is now a single `early` and a single `late` sigma.
2. Next, compute a single sigma from the `early` and `late` values from *step 1* above.

`1`

Separates `early` and `late` constraint OCV sigma tables are written to the output LVF file. The corresponding `sigma_type` value of `early` or `late` is included in each table.

`2`

Writes one OCV sigma constraint group without `sigma_type` to the output LVF file. The effective sigma of each parameter is computed from its `early` and `late` data. Then, the effective sigma across all parameters is computed. The `early` and `late` sigma data must be characterized (see non_linear_random_variation). Assume that there are two characterized parameters `p1` and `p2` each with `early` and `late` data, Then:

1. Compute the effective sigma for each parameter. There is now a single sigma associated with each parameter.
2. Compute a single sigma from the parameter-based sigma values computed in *step 1* above.

Note: See the define_variation_average command to tune to algorithm for averaging the data.

Example:

```
define_variation_average \
    -min_factor 0.0 \
    -max_factor 1.0 \
    -random simple_pos_is_max
```

Use this variable to select the method to be used to model the OCV constraint sigma tables in the output library.

Note: Although the 2014.09 Liberty User's Guide allows for inclusion of both early and late tables for constraints, downstream tools do not recommend using separate early and late tables.

This variable must be set before the `write_variation` command is run.

lvf_delay_early_late_mode

`<0 | 1 | 2>` Controls how the delay OCV sigma type data will be written to the LVF file.
Default and recommended: `1`

- `0` Only one OCV delay table with early and late sigma type is written to the LVF file.
- `1` Separate early and late sigma type OCV delay tables will be written to the LVF file. The corresponding `sigma_type` of `early` or `late` will be included in each table.
- `2` Similar to `0`, but uses the root-sum-square of the maximum of `early_and_late` values. Use this value with the following command setting to get the correct behavior:

```
define_variation_average
    -min_factor 0.0
    -max_factor 1.0
    -random abs
    -systematic abs
```

Use this variable to select the method to be used to model the OCV delay sigma tables in the output library.

This variable must be set before the `write_variation` command is run.

mark_failed_data

`<0 | 1>` When Variety determines that a characterization has failed, it sets the `altos_error_flag` inside the LDB. This variable determines how Variety should deal with the presence of this flag in the LDB.

- `0` Does not issue any warnings. The output library contains the value as stored in the LDB. This value can be a legal number and can pass the library compilation without generating any warnings. In most cases, the constraint value in `.lib` is an acceptionally large value.

1	Issues a warning when `read_ldb` reads in an LDB that contains the `altos_error_flag`. In addition, all children groups of the group containing the failed data have their values set to `DBL_MAX/3` (`INF` in `.lib`).

This variable must be set before the `write_variation` command is run.

mark_failed_data_replacement

<value>	When mark_failed_data is enabled, this variable specifies the value to be plugged into the library for the failed data.

This variable must be set before the `write_variation` command is run.

max_capacitance_attr_limit

<value>	Specifies the maximum allowed *max_capacitance* attribute value. Default: unlimited

This limit is applied after the `max_capacitance_factor` variable is applied.

Example

```
# Set max capacitance attribute limit
set_var max_capacitance_attr_limit 1000e-15
```

max_capacitance_factor

<value>	Specifies the multiplication factor to be applied to all *max_capacitance* attributes in the library. Default: `1` This factor is not applied to indices.

Example

```
# Set max capacitance factor
set_var max_capacitance_factor 0.66
```

max_capacitance_limit

<value>	Specifies the maximum allowable load index value. Default: `1F`

This variable controls a global limit that only becomes effective when `char_variation -auto_index` is enabled. It specifies the maximum output pin load capacitance, in Farads, that can be assigned to index_2. If the max load capacitance calculated by `-auto_index` exceeds this limit, the max load capacitance is reset to the value stored in this variable.

Example

```
# Set maximum allowed capacitance load
set_var max_capacitance_limit 1e-9
```

max_transition

<value>	Specifies the maximum allowable delay transition time (in seconds). Default: `3.0e-9` (3ns)

This variable is used to limit the maximum allowable output transition for a cell. Set this variable when using the `char_variation -auto_index` option.

Example

```
# Maximum output transition time allowed
set_var max_transition 1e-9
```

max_transition_attr_limit

<value>	Specifies the maximum allowed for all *max_transition* attribute value. Default: unlimited

This limit is applied after the `max_transition_factor` variable is applied.

Example

```
# Set max transition attribute limit
set_var max_transition_attr_limit 1000e-12
```

max_transition_factor

<value> — Specifies the multiplication factor applied to all *max_transition* attributes in the library.
Default: `1`

This factor is not applied to indices.

Example

```
# Set max transition factor
set_var max_transition_factor 0.66
```

measure_cap_lower_fall

<value> — The % point on the cell input waveform from which to measure falling input capacitance.
Default: `0.01` (1%)

Examples

```
# Set the capacitance measurement to 10%
set_var measure_cap_lower_fall 0.1
```

measure_cap_lower_rise

<value> — The % point on the cell input waveform from which to measure rising input capacitance.
Default: `0.01` (1%)

Examples

```
# Set the capacitance measurement to 10%
set_var measure_cap_lower_rise 0.1
```

measure_cap_upper_fall

<value> — The % point on the cell input waveform from which to measure falling input capacitance.
Default: `0.99` (99%)

Examples

```
# Set the capacitance measurements to 90%
set_var measure_cap_upper_fall 0.9
```

measure_cap_upper_rise

<value> — The % point on the cell input waveform from which to measure rising input capacitance.
Default: `0.99` (99%)

The above variables are used to control how input capacitance is measured after SPICE simulation.

Examples

```
# Set the capacitance measurements to 90%
set_var measure_cap_upper_rise 0.9
```

measure_output_range

`<0 | 1>` — Enables the measurement of the initial and final voltages of a pin when measuring the output transition range.
Default: `0`

- `0` — The outputs are assumed to swing full rail.
- `1` — A user-specified relative measurement threshold is applied to the actual final voltage of an output transition.

  Note: The initial and final voltages of the output must be different by more than 100mV, or the actual rail voltage is used.

measure_slew_lower_fall

<value> — Controls how output transition times are measured after SPICE simulation. The *<value>* is the % point on the cell output waveform to measure falling output transition times *to*.
Default: `0.2` (20%)

Examples

```
# Set the transition measurement to 10%
set_var measure_slew_lower_fall 0.1
```

measure_slew_lower_rise

<value> — Controls how output transition times are measured after SPICE simulation. The *<value>* is the % point on the cell output waveform to measure rising output transition times *from*.
Default: `0.2` (20%)

Examples

```
# Set the transition measurement to 30%
set_var measure_slew_lower_rise 0.3
```

measure_slew_upper_fall

<value> — Controls how output transition times are measured after SPICE simulation. The *<value>* is the % point on the cell output waveform to measure falling output transition times *from*.
Default: `0.8` (80%)

Examples

```
# Set the transition measurement to 70%
set_var measure_slew_upper_fall 0.7
```

measure_slew_upper_rise

`<value>` — Controls how output transition times are measured after SPICE simulation. The `<value>` is the % point on the cell output waveform to measure rising output transition times *to*.
Default: `0.8` (80%)

Examples

```
# Set the transition measurement to 90%
set_var measure_slew_upper_rise 0.9
```

merge_related_preset_clear

`<0 | 1 | 2>` — The arc caused by the de-assertion of a preset/clear signal may be considered a combinational_fall/rise or a clear/preset type. This variable enables the merging of related preset and clear states.
Default: `2`

- `0` — The preset/clear generates a rise/fall transition, and the timing arc from the de-assertion of the preset/clear is omitted. If combinational_risefall is set to **1**, both the preset/clear arc and its associated deassertion arc are output in separate timing groups.
- `1` — The rise and fall tables from an asynchronous preset/clear related pin are merged into a single timing group with the clear/preset type.
- 2 — Both the clear and preset timing_type groups are output.

Example

```
# Allow the merging of combinational timing types with
# preset/clear timing types.
set_var combinational_resefall 1
set_var merge_related_preset_clear 1
```

min_capacitance_for_outputs

`<0 | 1>` Enables the *min_capacitance* attribute for output pins.
Default: `0`

- `0` Does not output the *min_capacitance* attribute on output pins.
- `1` Outputs the *min_capacitance* attribute on output pins. A setting of `1` is useful to match legacy libraries.

This variable must be set before the `char_variation` command is run.

min_output_cap

<value> Specifies the minimum allowable output capacitance load (in Farads) when using the `-auto_index` option to `char_variation`.
Default: the minimum input pin capacitance found in the library is used.

Example

```
# Set the minimum load index
set_var min_output_cap 5e-16

# Set the minimum output transition time index
set_var min_transition 1e-11
char_variation -auto_index
```

min_transition

<value> Specifies the minimum allowable output transition time (in seconds) when using the `-auto_index` option to `char_variation`.
Default: automatically calculates the minimum transition index

mpw_glitch_peak

<value>

Specifies the minimum pulse width (MPW) glitch peak failure threshold.
Default: `0.95` (95% of Vdd)

Set this variable to the desired measurement threshold when the MPW measurement uses the glitch criteria (see `mpw_criteria`).

Note: This variable has no effect if the `mpw_criteria` variable is set to `delay`.

This variable must be set before the `char_variation` command is run.

mpw_input_threshold

<value>

Specifies the minimum height allowed when the clock pulse becomes triangular. The `min_pulse_width` constraint is deemed to be violated if this occurs before-hand.
Default: `0.9` (90% of vdd)

mpw_search_bound

<value>

Specifies the initial search bound for MPW characterization.
Default: `5e-9`

mpw_skew_factor

<value>

Specifies the ratio of the input fall slew over the input rise slew.
Default: `1.0` (rise and fall input slews are identical)

With this variable, you can skew the ratio of the input rise versus input fall slew for calculating `min_pulse_width` constraints.

mpw_slew

`<`*value* | min | mid | max`>`

Specifies the value for the input slew to be used when determining the minimum pulse width timing constraint of a clock or async signal.
Default: `min`

`value`	Specifies the value for the input slew to be used. The value can be any floating point number, in seconds.
`min`	The minimum input slew value is taken from the input slew indices of the first delay arc where this clock signal is a related pin (for example, a clock to Q delay arc on a flip-flop).
`mid`	The middle input slew value is taken from the input slew indices of the first delay arc where this clock signal is a related pin (for example, a clock to Q delay arc on a flip-flop).
`max`	The maximum input slew value is taken from the input slew indices of the first delay arc where this clock signal is a related pin (for example, a clock to Q delay arc on a flip-flop).

Examples

```
# Set the MPW glitch height criteria to 30% of Vdd
set_var mpw_glitch_height 0.3

# Set the slew for MPW to 200ps
set_var mpw_slew 200e-12

# Set fall to rise slew ratio to be 0.8
set_var mpw_skew_factor 0.8

# Set the MPW input threshold to 70% of Vdd
set_var mpw_input_threshold 0.7
```

mpw_slew_clock_factor

<value>

Specifies the ratio to apply to the `mpw_slew` when characterizing the minimum pulse width value of clock nets. Default: `1` (use the `mpw_slew` for clock nets.)

When this variable is set, the slew applied to clock nets is: `mpw_slew_clock_factor * mpw_slew`

Examples

```
# Set the slew for MPW for clocks to 100ps when
# other nets are using 200ps
set_var mpw_slew 200e-12
set_var mpw_slew_clock_factor 0.5
```

mpw_table

<0 | 1>

Enables the generation of a one-dimensional table of minimum pulse width (MPW) values.
Default: `1`

`0` Generates only high and low MPW attributes. Does not generate a table.

`1` Generates an MPW table using the index_1 table as specified by the constraint_template associated with the `define_cell` command for the cell. Characterizing MPW tables increases the run time.

mpw_variation

<0 | 1>

Enables table based MPW sensitivity. Default: `0`

`0` Does not characterize MPW for sensitivity, meaning that the min_pulse_width sensitivity is all zeros (“0”).

`1` Enables table-based min_pulse_width sensitivity characterization.

msg_level

`<0 | 1>` Controls the verbosity of error and warning messages.
Default: `0`

- `0` Outputs error messages and useful warning and informational messages.
- `1` Outputs all messages.

  This setting can output a lot of messages (some of which may not be helpful) making it difficult to determine which messages are important.

non_linear_random_variation

`<0 | 1 | 2 | 3>` Specifies how to characterize random variation.
Default and recommended: `3` (use and store both positive and negative variations)

- `0` Characterizes using the positive variation for constraint arcs while assuming a symmetric variation. Characterizes both positive and negative variation simultaneously for non-constraint arcs.

  Note: This option gives the best run time.
- `1` Characterizes using both positive and negative variations independently.
- `2` Characterizes using both positive and negative variations simultaneously and stores the average into the LDB.
- `3` Characterizes using both positive and negative variations simultaneously and stores both positive and negative variations into the LDB.

Note: Options `1`, `2`, and `3` have a performance penalty over `0`.

This variable must be set before the `char_variation` command is run.

nonseq_as_recrem

`<0 | 1>` Controls whether arcs with `timing_type`, `nonseq_setup`, and `nonseq_hold` are converted into recovery and removal.
Default: `0`

- `0` The arcs are not converted.
- `1` All arcs with the timing type of nonseq_setup and nonseq_hold are converted into recovery and removal.

Example

```
# Convert non_seq_setup and non_seq_hold timing types
# to 'recovery and removal'
set_var nonseq_as_recrem 1
```

output_internal_pin

`<0 | 1>` Controls whether internal probe pins are included in the `.lib` file.
Default: `0`

- `0` Internal probe pins are not output into the `.lib` file.
- `1` Internal probe pins are output in the `.lib` file with the type `internal_pin`.

packet_arc_notification_interval

`<value>` Specifies the minimum time interval between two informational notifications (see `packet_arc_notification_list`). The range of value is between 0 to 72000.
Default: `600` (in Seconds = 10 minutes)

This variable must be used before the `char_ams` command.

Example

```
set_var packet_arc_notification_interval 3600
```

The above example requests no more than one informational notification per hour.

packet_arc_notification_limit

<value>	Specifies the maximum number of informational notifications per run. This variable is effective when the `packet_arc_notification_list` has been set. The specified value should be between 0 to 100. Default: `10`

This variable must be used before the `char_ams` command.

Example

```
set_var packet_arc_notification_limit 5
```

The above example limits the notifications to no more than 5.

packet_arc_notification_list

<string>	Sets the e-mail addresses or SMS equivalent e-mail addresses that can receive notifications. Multiple e-mails or SMS numbers can be specified by using a comma-separated list. By default, no notifications are sent. You can set this variable to a valid e-mail address to enable notifications to that address. Default: `""` (empty list)

Requirements:

- The main Liberate AMS job must run on a machine that is able to send e-mails.
- Any SMS numbers provided for notifications should be able to receive messages by e-mail. Some carriers block this ability to prevent spam messages.

This variable must be used before the `char_ams` command.

Example

```
set_var packet_arc_notification_list "1111111111@mms.att.net,\
2222222222@messaging.sprintpcs.com,3333333333@tmomail.net,abc@def.com"
```

The above example has three SMS numbers (ATT/Sprints PCS/TMobile) and one e-mail address.

packet_client_idle_count

<value> — Specifies the permissible number of idle clients.
Default and recommended: `-1` (Number controlled by internal heuristics)

Set this variable to reduce the number of clients available for characterization near the end of the run. The default and recommended value of `-1` allows the tool to internally control the client release mechanism.

In case of characterizing statistical delay, setting this variable to `2` helps to release the idle clients in a more timely manner.

This variable must be set before the `char_variation` command is run.

packet_client_resubmit_count

<value> — Specifies the number of times a failed LSF job should be resubmitted for simulation. Liberate will also check the LDB to make sure it contains data from the job, and will resubmit if necessary.

This variable must be set before the `char_variation` command is run.

packet_client_timeout

<value> — Specifies a timeout limit (in seconds) for client machines on the network. If a packet client log file has not been updated for more than the number of seconds specified, Liberate will assume that packet has died. (This can occur because of a machine crash, or a signal such as "`kill -9`" that can't be trapped.) If a packet client is determined to be “dead”, then the server will not wait and move on to the next client.
Default: `86400` (1 day in seconds)

This variable must be set before the `char_variation` command is run.

packet_client_timeout_action

`<value>` Specifies the timeout value. This variable has no effect unless you set the packet_client_timeout variable to a valid value. The two possible values are, `warning` and `error`.
Default: `warning`

- `error` The master marks the job and the associated cell as *failed*. If you are using LSF, the client job is killed using `bkill`.
- `warning` Output a warning into the log file when the `packet_client_timeout` value is exceeded. The timer is reset and the warning is generated repeatedly after the timeout is exceeded.

This variable must be set before the `char_variation` command is run.

packet_clients

`<0 | integer>` Enables Parallel Packets mode and specifies the number of machines to be used for distributed processing.
Default: `0`

- `0` Disables Parallel Packet mode.
- *`integer`* Enables Parallel Packet mode *and* sets the number of machines to be used.

Important

If your flow uses write_vdb, you must set this to `0`.

This variable must be set before the `char_variation` command is run.

packet_log_filename

`<file_name>` — Name of log file. You must set this to match the log file specified in the `rsh_cmd` to report characterization statistics.

Note: It is recommended that you should use the default name "`log`" and setting `rsh_cmd` to use "`/log`" as stdout and stderr filenames.

Important

"`%L`" is not allowed as a string in `packet_log_filename`.

This variable must be set before the `char_variation` command is run.

packet_mode

`<cell | arc>` — This controls the Parallel Packet Distribution Mode. Liberate can distribute Parallel Packets in cell-based mode or arc-based mode.
Default and recommended: `arc`

`cell`	Enables distribution of Parallel Packets in cell-based mode.
`arc`	Enables distribution of Parallel Packets in arc-based mode.

This variable must be set before the `char_variation` command is run.

packet_rdb_count

`<0 | 1>` — Enable RDB in parallel packets.

`0`	Disables RDB in parallel packets
`1`	Enables RDB in parallel packets. (Default)

Important

Set to `0` to disable RDB in parallel packets, otherwise recovery flow (`read_ldb`/`char_variation`) may not work.

This variable must be set before the `char_variation` command is run.

packet_rsh_mode

`<lsf | ns | custom>` Instructs the server to automatically kill client jobs if the server job is interrupted during the arc packet flow (see packet_mode).
Default: `custom`

This variable is automatically set for the following batch submission commands: `bsub` and `nc`. (see rsh_cmd and `set_rsh_cmd`.) You might need to explicitly set this variable to get the correct value if the `rsh_cmd` command points to a wrapper script instead of using the batch submission commands.

This variable must be set before the `char_variation` command is run.

parenthesize_not

`<0 | 1>` Controls whether parentheses are placed around variable names that are negated using "`!`"
Default: `0`

- `0` The parentheses are not added, resulting in: `!A`.
- `1` Parentheses are placed around negated variable names in logic functions, resulting in, for example: `!(A)`.

Example

```
# Remove () from variable names
set_var parenthesize_not 0
```

parse_auto_define_leafcell

`<0 | 1 | 2>` Controls whether Variety should recognize leafcells when parsing a netlist; if yes, then how should it happen.
Default: `1`

- `0` Do not generate define_leafcell commands. This setting is provided for backward compatibility to LIBERATE 14.1 ISR3 and prior releases.

`1`	Recognize leafcells automatically and prints `define_leafcell` commands into the log file. All of the user-provide `define_leafcell` commands are preserved and honored.
`2`	Detect leafcells automatically, but do not apply the detected leafcells to the analysis. Print `define_leafcell` commands into the log file so that you can examine and fill in the missing information. For example: `l`, `w`, `nfin`, `cjsw`, and so on.

A leafcell is the lower most instance that is found when flattening a netlist that is outside the model file. The define_cell command can be used to manually specify leafcells. Also, the auto-detection of leafcells is supported only when using the Spectre Front End (SFE) parser (see read_spice `-format spectre`) and the extsim_model_include variable is set.

This variable must be set before the `char_variation` command is run.

predriver_waveform

`<0 | 1 | 2>` Controls whether a linear ramp or a waveform based on averaging a linear ramp and the equivalent exponential response from an RC network, are used as the input driver. Default: `0`

`0`	Uses a linear ramp as the input slew.
`1`	Sets a PWL waveform as the input driver based on averaging a linear ramp and the equivalent impulse response from an RC network. Using this analytical waveform gives a good approximation for real waveforms over a large variety of different input driver/receiver combinations including fast slews on short wires and slow slews on long wires.
`2`	The linear ramp used will not be limited by the supply rail, but will continue in a linear fashion. This setting is recommended when characterizing CCS format data.

Setting `predriver_waveform` to `1` overrides any `set_driver_cell` commands.

Example

```
# Use a PWL pre-driver derived from an RC network
set_var predriver_waveform 1
```

predriver_waveform_ccs_variation_mode

`<0 | 1>` Controls whether pre-driver waveforms are written to each individual variation library.
Default: `0`

- `0` Does not write out pre-driver waveforms.
- `1` Writes out the pre-driver waveforms to each individual variation library generated by `write_variation -ccs`.

rcp_cmd

`<scp | rcp | cp>` Controls which file-copy command is used for copying files from the host to the client machine when using distributed parallel processing.
Default: `scp`

- `cp` Uses the `cp` (copy) command.
- `rcp` Uses the `rcp` (remote copy) command.
- `scp` Uses the `scp` (secure copy) command.

rdb_exit_if_source_differ

`<0 | 1>` Controls whether the tool exits if mismatched PVT corners are found.
Default and recommended: `1`

- `0` Continues when mismatched PVT corners are found and does not exit.
- `1` Exits if mismatched PVT corners are found.

The RDB flow ensures that exactly the same script/settings are used before restoring characterized values from RDB. This ensures data consistency is maintained from run to run even if the user erroneously sets the `rdb_checkpoint_dir` variable to the same location.

```
# To disable this check:
set_var rdb_exit_if_source_differ 0
```

removal_glitch_peak

<value>

Specifies the maximum size of a voltage glitch permitted on the constraint output pin before an arriving signal is deemed to fail a removal constraint.
Default: `-1` (use the same value as `constraint_glitch_peak`)

Example

```
# Set removal glitch peak to 5%
set_var removal_glitch_peak 0.05
```

report_detail_variation

<0 | 1 | 2>

Reports the sensitivities for each parameter.
Default: `2`

- `0` — Disables reporting of sensitivities.
- `1` — Reports time-stamped per-parameter simulation status and average sensitivities for the entire characterization.
- `2` — Reports a table of per-cell sensitivities in addition to the results reported when the variable is set to `1`.

This variable must be set before the `char_variation` command is run to get the full benefits.

reset_negative_constraint

<0 | 1>

Controls whether negative constraint values are replaced with zeros.
Default: `0`

- `0` — Outputs a negative constraint (setup, hold, recovery, and removal) value if the simulator measurement returns a negative value.

`1`	Causes the output library to have the negative values replaced by zeros.

This reset is applied after `set_constraint -margin`.

This variable is not applied to MPW constraints.

reset_negative_delay

`<0 | 1>` Controls whether negative delay or transition values are replaced with zeros.
Default: `0`

`0`	Outputs a negative delay and transition value if the simulator measurement returns a negative value.
`1`	Causes the output library to have the negative values replaced by zeros.

Examples

```
# Turn all negative delays and constraints to zeros
set_var reset_negative_delay 0
set_var reset_negative_constraint 0
```

resolve_collision

`<0 | 1 |2>` Controls how Variety resolves collision vectors with the requested SPICE engine.
Default: 2

`0`	Disables collision resolution using simulation.
`1`	Uses Alspice to resolve collisions even when `extsim_model_include` is enabled.
`2`	Resolves the collisions by using the external simulation engine if `-extsim` is used in `char_variation` or `extsim_model_include` is defined.

Collision vectors occur when a node has a path to vdd and to gnd. In these cases, Variety can simulate the vector in SPICE to determine if the output should result in a digital behavior.

This variable must be set before the `char_variation` command is run.

rsh_cmd

<ssh_cmd_str | rsh_cmd_str>

Controls which shell command is used for accessing a remote client when using distributed parallel processing.
Default: `ssh`

rsh_cmd_str Specifies an `rsh` (remote shell) command.

ssh_cmd_str Specifies an `ssh` (secure shell) command.

The `rcp_cmd` and `rsh_cmd` variables are used to control the interface to remote clients when using distributed parallel processing. Before using parallel processing, make sure that the server machine (the machine from which Liberate LV is run) can perform an `rsh` or `ssh,` and an `rcp` or a `scp`, to each client machine without requiring a password or passphrase.

The `rsh_cmd` command string can reference the current client and the command by which to invoke Variety by using `%M` (machine) and `%C` (command). In addition, command line arguments that appear after the Variety Tcl file name can be passed into the `rsh_cmd` string by using `%O` (options). These substitutions can be useful if your system is using a job queuing system.

Examples

```
# Set the shell and copy variables for distributed runs
# Set remote file copy to rcp instead of scp
set_var rcp_cmd rcp
# Set remote shell to rsh instead of ssh
set_var rsh_cmd rsh
```

scale_load_by_template

<0 | 1>

Determines whether the load indices are scaled using the values defined in the template when generating indices using the `-auto_index` option of `char_variation`.
Default: `0`

`0` Does not scale the load indices.

`1` — Turns on scaling of the load indices created by the `-auto_index` option of `char_variation`. The indices are scaled using the following formula:

```
scaled_load_index_value(i)=
    (max_load-min_load)*
    template_load_index_value(i) +
    min_load
```

scale_tran_by_template

`<0 | 1>` — Controls whether the transition indices are scaled using the values defined in the template when generating indices using the `-auto_index` option of `char_variation`.
Default: `0`

`0` — Does not scale the transition indices.

`1` — Turns on scaling of transition indices created by the `-auto_index` option of `char_variation`. The indices are scaled using the following formula:

```
scaled_tran_index_value(i)=
    (max_tran-min_tran)
    *template_tran_index_value(i)
    + min_tran
```

When using `scale_tran_by_template`, the template index_1 and index_2 values should be from 0-1.

Example

```
define_template -type delay \
    -index_1 {0 0.1 0.2 0.4 0.7 0.9 1} \
    -index_2 {0 0.2 0.4 0.8 1} \
delay_template_7x7

define_cell \
    -input { A } -output { X } \
    -delay delay_template_7x7 INV_1

set_var min_transition 6.6e-12
set_var max_transition 0.6e-9
set_var scale_load_by_template 1
set_var scale_tran_by_template 1
char_variation -auto_index
```

sdf_cond_prefix

"*prefix*"	Specifies the prefix to use for naming complex conditional `sdf_cond` attributes on sequential cells (flip-flop or latch). Default: `"adacond"`

If an `sdf_cond` attribute is complex (has two or more operands) it is replaced by a name of the form *`sdf_cond_prefix#`* where the `#` is unique for each `sdf_cond` within the cell. The `sdf_cond` attributes are used for writing conditional timing arcs in Liberty format (see `write_variation`).

Example

```
# Set the sdf_cond prefix
set_var sdf_cond_prefix "int_cond"
char_variation
write_variation my.lib
```

sdf_cond_style

`<0 | 1>`	Determines the `sdf_cond` style. Default: `0`

`0` — Generates a unique variable for each different condition as follows: `"${`*`sdf_cond_prefix`*`}#"` where the *`sdf_cond_prefix`* is set using the `sdf_cond_prefix` variable. Default: `adacond`.

This style applies only to complex `when` conditions that have multiple operands.

1	Creates a variable name from the `when` condition such that:
	Logic *and* operators are replaced with `AND`
	Logic *or* operators are replaced with `OR`
	Logic *not* operators are replaced with `NOT`
	`(` is replaced with `OP`
	`)` is replaced with `CP`
	These new operands are separated from the operators by underscores. This constructed variable name is prefixed with the value of the `sdf_cond_prefix` variable.
	This style applies to all conditional constraints.

The `sdf_cond` attribute for complex conditional constraint arcs must be represented as a "variable" that is the functional equivalent to the `when` condition.

This variable must be set before the `char_variation` command is run.

Example

```
"A * (B | !C)" becomes "adacond_A_AND_OP_B_OR_NOT_C_CP"
```

sdf_logic_and

"*string*"	Specifies the characters to use for denoting logic AND in `sdf_cond` attributes. Default: " `&` " (*Notice the space on both sides of* &)

Examples

```
# Set the logic AND to &&
set_var sdf_logic_and "&&"
```

sdf_logic_not

"*string*"	Specifies the characters to use for denoting logic NOT in `sdf_cond` attributes. Default: "~"

Examples

```
# Set the logic OR to ||
set_var sdf_logic_or "||"
```

sdf_logic_or

"*string*"	Specifies the characters to use for denoting logic OR in `sdf_cond` attributes. Default: " | " (*Notice the space on both sides of* |)

Examples

```
# Set the logic NOT to !
set_var sdf_logic_not "!"
```

set_var_failure_action

<warning | error>	Notifies the `set_var` command how to consider a failure. Default: `warning`	
	`error`	Displays an error message when a `set_var` command fails and suppresses execution of any subsequent commands like `char_variation` that might result in characterization or library generation. However, subsequent `set_var` commands are still allowed to check those for correctness.
	`warning`	Displays a warning when a `set_var` command fails. The failed `set_var` command is ignored and the execution of subsequent commands continues.

This variable must be set before any other `set_var` command.

skip_nfs_sync

`<0 | 1>` Forces an NFS file system synchronization by including multiple commands separated by a semicolon (`;`) on the command line.
Default: `0`

- `0` Submit runs to the batch tool using:
  `ls -l dirname; variety`
- `1` Submit runs to the batch tool using:
  `variety`

This variable might be needed when using a batch system that cannot accept a batch command containing multiple commands separated by a semicolon.

This variable must be set before the `char_variation` command is run.

slew_lower_fall

`<value>` Specifies the % point on the cell output waveform to output falling output transition times *to*.
Default: `0.2` (20%)

This variable is used to control how output transition times are stored in the delay tables. This variable can be set differently from the equivalent measurement threshold, in which case the measurement is normalized to fit the appropriate output transition slew threshold.

If the `slew_lower_fall`, `slew_upper_fall`, `slew_lower_rise`, and `slew_upper_rise` variables are set symmetrically, the output library that is generated includes the slew_derate_from_library attribute and the slew_*_threshold_pct_* attributes are set to the equivalent `measure_slew_*` variables (x 100). If they are not symmetrical, the slew_derate_from_library attribute is omitted and the slew_*_threshold_pct_* attributes are set to the equivalent `slew_*` variables (x 100).

Examples

```
# Set the output transition thresholds to 10%
set_var slew_lower_fall 0.1
```

slew_lower_rise

<value> — Specifies the % point on the cell output waveform to output rising output transition times *from*.
Default: `0.2` (20%)

This variable is used to control how output transition times are stored in the delay tables. This variable can be set differently from the equivalent measurement threshold, in which case the measurement is normalized to fit the appropriate output transition slew threshold.

If the `slew_lower_fall`, `slew_upper_fall`, `slew_lower_rise`, and `slew_upper_rise` variables are set symmetrically, the output library that is generated includes the slew_derate_from_library attribute and the slew_*_threshold_pct_* attributes are set to the equivalent `measure_slew_*` variables (x 100). If they are not symmetrical, the slew_derate_from_library attribute is omitted and the slew_*_threshold_pct_* attributes are set to the equivalent `slew_*` variables (x 100).

Examples

```
# Set the output transition thresholds to 10%
set_var slew_lower_rise 0.1
```

slew_upper_fall

<value> — Specifies the % point on the cell output waveform to output falling output transition times *from*.
Default: `0.8` (80%)

This variable is used to control how output transition times are stored in the delay tables. This variable can be set differently from the equivalent measurement threshold, in which case the measurement is normalized to fit the appropriate output transition slew threshold.

If the `slew_lower_fall`, `slew_upper_fall`, `slew_lower_rise`, and `slew_upper_rise` variables are set symmetrically, the output library that is generated includes the slew_derate_from_library attribute and the slew_*_threshold_pct_* attributes are set to the equivalent `measure_slew_*` variables (x 100). If they are not symmetrical, the slew_derate_from_library attribute is omitted and the slew_*_threshold_pct_* attributes are set to the equivalent `slew_*` variables (x 100).

Examples

```
# Set the output transition thresholds to 90%
set_var slew_upper_fall 0.9
```

slew_upper_rise

<value>	Specifies the % point on the cell output waveform to output rising output transition times *to*. Default: `0.8` (80%)

This variable is used to control how output transition times are stored in the delay tables. This variable can be set differently from the equivalent measurement threshold, in which case the measurement is normalized to fit the appropriate output transition slew threshold.

If the `slew_lower_fall`, `slew_upper_fall`, `slew_lower_rise`, and `slew_upper_rise` variables are set symmetrically, the output library that is generated includes the slew_derate_from_library attribute and the slew_*_threshold_pct_* attributes are set to the equivalent `measure_slew_*` variables (x 100). If they are not symmetrical, the slew_derate_from_library attribute is omitted and the slew_*_threshold_pct_* attributes are set to the equivalent `slew_*` variables (x 100).

Examples

```
# Set the output transition thresholds to 90%
set_var slew_upper_rise 0.9
```

sort_cells

`<0 | 1>`	Controls the sorting of cells into the output library file. Default: `1`
`0`	Disables sorting cells alphabetically.
`1`	Sorts the cells alphabetically.

Example

```
# Do not sort cells
set_var sort_cells 0
```

spice_delimiter

`"string"`

Specifies the hierarchy delimiter in the SPICE format netlists loaded into **read_spice**. It can be a single- or multiple-character string.
Default: "."

Every character in this variable is treated as a hierarchical delimiter.

In the SPICE decks that are written out, the first character in this string will be used as the hierarchical delimiter.

Example

```
# Set the SPICE delimiter to /
set_var spice_delimiter "/"
```

supply_define_mode

`<0 | 1>`

Selects the algorithm used for determining supplies.
Default: `0`

`0` — Determines the supplies by tracing the transistor connectivity.

`1` — Determines the supplies from module wires, rather than by tracing transistor connectivity. This setting can be useful in matching legacy libraries.

This variable must be set before the `set_operating_condition` command is run.

tristate_disable_transition

`<0 | 1>`

Controls whether the output of transition values in the library should be enabled or disabled.
Default: `1` (Copy delay to transition)

`0` — Outputs a scalar value in the transition tables for the `three_state_disable` arcs.

1	Outputs full transition tables for the three_state_disable arcs. The transition disable arc data will be copied from the corresponding delay tables.

This variable can be used before the write_library command.

user_data_override

{max_capacitance | cell_leakage_power}

Specifies a list of items that, if found in the user_data file, override the corresponding data in the LDB when loaded by the write_variation -user_data option.
Default: {area function three_state state_function}, which is the built-in list of overrides.

The specified list appends values to the built-in list of overrides.

Only two elements are valid for the override list: max_capacitance (a simple attribute under the pin() group) and cell_leakage_power (a simple attribute under the cell() group)

Example

```
set_var user_data_override {max_capacitance \
    cell_leakage_power}
```

variation_constraint_path_delta

<0 | 1>

Enables the path-delay constraint methodology.
Default: 0
Recommended: 1

0	Uses a bisection search for constraint variation.
1	Uses a path-delay difference for constraint variation.

This variable is used to enable the acquisition of constraint variation using a much faster method than bisection search. In this method, the constraint is measured using the difference

in the data-path delay and clock-path delay, instead of the bisection search algorithm. The constraint sensitivity is determined using this path-delta algorithm to measure the constraints.

This variable must be set before the `char_variation` command is run.

variation_dominant_xtr_ccc_abstol

`<value>` — Specifies a dominant transistor abstol in seconds.
Default: `0.0` (all transistors are dominant)

If an active transistor has a small variation impact on the CCC delay, the impact to the entire cell should also be small. Variety attempts to identify the dominant transistors that must be characterized in full detail at all slew and load combinations while reducing the characterization work on less dominant transistor. The flow is as described below:

1. Determine the variation impact of each active transistor in a CCC. This is done by measuring the sensitivity at a reduced set of slew and load combinations.
2. If the sensitivity of a transistor is greater than this abstol, select the transistor as dominant and continue to characterize it in detail.
3. If the sensitivity is less than this abstol, no further characterization will be done on this transistor.

This variable must be set before the `char_variation` command is run.

variation_early_late_adjust_mode

`<0 | 1>` — Performs quadratic sampling based on timing sensitivity equation to improve `ocv_sigma` accuracy.
Default: `0`

Note: To use this variable, the moments algorithm must be enabled. (See variation_mean_nominal_mode.)

- `0` — Disables the adjustment of the `ocv_sigma` values using quadratic sampling.
- `1` — Enables the adjustment of the `ocv_sigma` values using quadratic sampling.

This variable must be set before the `char_variation` command is run.

Example

```
set_var variation_sds_samples 100000
set_var variation_target_sigma 3
set_var variation_parallel_mos_mode 0
set_var variation_sigma 1
set_var variation_mean_nominal_mode 4
set_var -type delay variation_mean_nominal_cross_params 10
set_var -type delay variation_early_late_adjust_mode 1
define_variation -type random "p1  1" p1
define_variation -type random "p2  1" p2
define_variation -type random "p3  1" p3

define_variation_factor { \
    constraint 2.5 \
    delay 2.5 \
    }

define_variation_group \
    -sigma_factor 1 \
    {p1 p2 p3} \
    LOCAL_VARIATION
```

variation_ecsm_cap_input_pin

<0 | 1>	Controls the characterization of ECSM capacitance for input pins. Default: 1

`0`	Reverts to the release 3.0p3 and prior release behavior where "hidden" arcs are not characterized and the more accurate ECSM capacitance is missing. Prior to release 3.1, Variety characterized ECSM capacitance for all input pins that have forward timing arcs and are characterized for variation effects. Input pins that did not have forward timing arcs and only had hidden power arcs were not characterized. (This included pins such as the "D" input and the inactive edge of a clock pin of a flip-flop.) Because these "hidden" arcs were not characterized for timing variation, they also did not have ECSM pin capacitance. All inputs pins always have the less accurate NLDM pin_capacitance.
`1`	Characterizes ECSM capacitance for all input pins (including bi-direction pins).

This variable must be set before the `char_variation` command is run.

variation_enable_non_zero_mean

`<0 | 1>` Controls whether Variety, for sensitivity simulation, applies an assumed value of `0` for the mean or whether it detects the mean and applies the real mean value.
Default: `1`

`0`	Only the value defined in the `define_variation` command and an assumed mean value of `0` are output in the library.

`1`	Detects the mean and applies the real mean value instead of `0`, for sensitivity simulation. If the variation is defined by using the `define_variation -sigma` option, the real sigma and mean value are put into the library.

This variable must be set before the `char_variation` command is run.

variation_flatten_netlist_mode

`<0 | 1>` Enables modified netlist handling for variation effect. Default: `1`

`0`	Enables modified netlist handling when characterizing variation. This setting can improve accuracy and increase run time. This setting can be applied to specific data types using `set_var -type`. To set this to `0` for specific data types, it must first be set globally to `1`.
`1`	Enables the default netlist handling mode. This setting gives the best run time.

This variable must be set before the `char_variation` command is run.

Example

```
# To apply a setting of 0 only to delay arcs:
set_var variation_flatten_netlist_mode 1
set_var -type delay variation_flatten_netlist_mode 0
```

variation_mean_nominal_cross_params

<integer>

Enables an algorithm that considers cross terminals for timing correlation for moments. Use this variable to specify the number of transistor and process parameter pairs to be considered. Default: `0` (disables this algorithm)

This variable can be selectively enabled by data type.

Important

Specifying a large value, such as greater than 10, might result in considerable performance penalty. In addition, to use this variable, the moments algorithm must be enabled. (See variation_mean_nominal_mode)

This variable must be set before the `char_variation` command is run.

Example

```
set_var -type delay variation_mean_nominal_cross_params 3
```

variation_mean_nominal_mode

`<0 | 4>`

Controls whether Variety should detect the mean values. Default: `0`

`0` Variety does not detects the mean, standard deviation, and skewness values. It writes only the nominal values into the library.

`4` Variety detects the mean, standard deviation, and skewness values along with the nominal values. It then stores all that data into the LDB.

Note: This option can be used with variation_mean_nominal_model_mode to output the OCV mean shift, standard deviation, and skewness into an LVF sensitivity file.

This variable must be set before the `char_variation` command is run.

variation_mean_nominal_include_voltage

`<0 | 1>` Specifies whether the mean nominal computation should include the characterized voltage variation. This variable affects the computation of the moments that are used by Tempus. See variation_mean_nominal_model_mode.
Default: `1`

- `0` Exludes the voltage variation when computing the moments.
- `1` Includes the voltage variation when computing the moments.

This variable must be set before the `char_variation` command is run.

variation_mean_nominal_model_mean_shift

`<0 | 1>` Controls the modeling of mean shift in the output library.
Default: `1`

Use this variable to control the modeling of mean shift in the output library. To model the mean shift, see the variables variation_mean_nominal_mode and variation_mean_nominal_model_mode.

- `0` Exludes the mean shift in output library.
- `1` Includes the mean shift in output library.

This variable must be set before the `write_library` command is run.

variation_mean_nominal_model_mode

`<0 | 1 | 2>` Enables modeling of mean shift, standard deviation, and skewness.
Default: `0` (not enabled)

- `0` Saves the characterized `ocv_sigma` data into the output file. The `ocv_sigma` data can contain delay, transition, and constraint arc sensitivities depending on the characterization setup.

`1` — Saves to the output file the mean shift, standard deviation, and skewness for delay and transition. This is in addition to the `ocv_sigma` data obtained by setting this variable to `0`. The additional data is output as new groups after the nominal group, but before the `ocv_sigma` groups. The additional groups are named as follows:

```
cell_rise_mean_shift,
cell_rise_std_dev, cell_rise_skewness
cell_fall_mean_shift,
cell_fall_std_dev, cell_fall_skewness
rise_transition_mean_shift,
rise_transition_std_dev,
rise_transition_skewness
fall_transition_mean_shift,
fall_transition_std_dev,
fall_transition_skewness
```

`2` — Writes only the Tempus moments into the output library. The `ocv_sigma_*` data are not included in the output library.

Use this variable to save additional data into the sensitivity file. Liberate uses this additional data (see `write_library -sensitivity_file`) to write additional data that is used by Tempus into the output library.

To add this additional data into the sensitivity_file, the characterization must save the additional data into the LDB (see variation_mean_nominal_mode).

This variable must be set before the `write_variation` command is run.

variation_mean_nominal_model_skewness

`<0 | 1>` — Controls the modeling of skewness in the output library. Default: `0`

Use this variable to control the modeling of skewness in the output library. To model skewness, see the variables variation_mean_nominal_mode and variation_mean_nominal_model_mode.

`0` — Exludes the skewness in output library.

`1` — Includes the skewness in output library.

This variable must be set before the `write_library` command is run.

variation_normalized_ecsm_mode

`<0 | 1>` Controls whether unitless SECMS sensitivity data is scaled or not.
Default: `0`

- `0` Scales only the SECSM sensitivity data for variation parameters with units; SECSM sensitivity data for unitless variation parameter data is not scaled.
- `1` Scales all the SECSM sensitivity data whether the parameter has units or not.

variation_onesided_voltage

`<0 | 1>` Selects one-sided or two-sided voltage variation.
Default and recommended: `0` (two-sided)

Variety can characterize the timing variation caused by changes in the supply voltage. Use this variable to control whether the supply voltage should be varied in both positive and negative directions, or only in the positive direction. For more information on enabling supply voltage variation characterization, see the define_variation command.

- `0` Characterizes both the positive and negative variation of the supply voltage.
- `1` Characterizes only the positive variation of the supply voltage and copies the positive sensitivity into the negative sensitivity.

This variable must be set before the `char_variation` command is run.

variation_onesided_voltage_zero_mode

`<off | early | late | auto>`

Specifies whether voltage variation sensitivity should be zeroed out for early or late timing.
Default: `0`

Note: For this variable to have any effect, voltage variation characterization must have been requested using the define_variation command.

`off`	Keep all the characterized voltage sensitivity data.
`early`	The early timing voltage variation sensitivity is zeroed out.
`late`	The late timing voltage variation sensitivity is zeroed out.
`auto`	The sensitivity is zeroed out depending on the measured effect based on the actual simulation result. If the voltage variation results in an early result, then zero out the late sensitivity; otherwise, zero out the early sensitivity.

This variable must be set before modeling using the `write_variation`, `write_variation_table`, and `create_aocv_table` commands.

variation_parallel_mos_mode

`<0 | 1 | 2>`

Enables improved handling of parallel MOS devices in the netlist. This results in improved accuracy and run time during variation characterization.
Default: `0`
Recommended: `2`

`0`	Ensures that the effect of each parallel MOS device is considered independently during variation characterization.

1	Enables the heuristics for improved accuracy and run time during variation characterization when parallel transistors are present in the netlist. Transistors are considered to be parallel if they have the same logical drain, gate, and source nodes. This algorithm requires that a leafcell-based flow be used. See also the extsim_model_include variable and define_leafcell command.
2	Enables the heuristics for improved accuracy and run time during variation characterization when parallel transistors are present in the netlist. In this mode, transistors with the same logical drain, gate, and source nodes as well as those with disconnected center nodes are considered parallel.

Note: This variable can be disabled (=0) on a local cell basis (see set_var). However to do this, the variable must be first enabled on a global basis.

This variable must be set before the `char_variation` command is run.

Example

The following example illustrates how a cell with parallel transistors and unconnected stack nodes is handled when `variation_parallel_mos_mode` is set to `2`:

```
.SUBCKT nand2_parallel2 A B Y VDD VSS
MP1 Y A VDD VDD pmos L='length' W='width1'
MP2 Y A VDD VDD pmos L='length' W='width1'
MP3 Y B VDD VDD pmos L='length' W='width2'
MP4 Y B VDD VDD pmos L='length' W='width2'

MN1 Y A mid1 VSS nmos L='length' W='width3'
MN2 mid1 B VSS  VSS nmos L='length' W='width4'
MN3 Y A mid2 VSS nmos L='length' W='width3'
MN4 mid2 B VSS  VSS nmos L='length' W='width4'
.ENDS
```

variation_path_delta_no_toggle_probe_margin_mode

`<0 | 1>` Controls addition of a margin to the sensitivity on the minimal slew for the path whose delay cannot be measured when the path delta algorithm is used (see variation_constraint_path_delta).
Default: `0` (Disabled)

- `0` Disables addition of a margin to the minimal slew sensitivity for no toggle when path delta algorithm is used.
- `1` Enables addition of a margin to the minimal slew sensitivity for no toggle when path delta algorithm is used.

This variable must be set before the `char_variation` command is run.

variation_path_delta_slew_interp_factor

<value> Specifies a factor to apply to the input slew of a path.
Default: `1`

The Path Delta constraint characterization algorithm computes a constraint as the difference between the clock path and the data path delays. For example, the reset removal time is computed by using the formula: `(clock path delay - reset path delay)`.

Variety cannot measure the turning off delay of the reset path because the internal probe node is not toggled by reset turning off. When using the bisection based result, it can be shown that the reset input slew can impact the sensitivity. The Path Delta constraint characterization algorithm applies a margin depending on the input pin slew for paths whose delays cannot be measured directly. `variation_path_delta_slew_interp_factor` specifies a factor to apply to the input slew when applying this margin. The value must be equal to or greater than `0`. A smaller value decreases pessimism and a larger value increases pessimism in the computed sensitivity values.

This variable must be set before the `char_variation` command is run.

variation_random_delay_mode

`<0 | 1>` Controls the methodology regarding accuracy and performance for delay and transition sensitivity tables.
Default: `1`

This variable affects the methodology for acquiring delay and transition sensitivities. If enabled, a proprietary methodology is used that improves the performance and accuracy when compared to Monte Carlo SPICE.

`0`	Use the original methodology for acquiring delay and transition sensitivities.
`1`	Use the proprietary methodology introduced in Liberate 14.1 ISR4.

This variable affects characterization and must be set before the `char_variation` command is run.

variation_random_search_filter_mode_size

`<1 | 2>` Controls the table characterization of constraint arcs when generating LVF data format.
Default: `2`

`1`	Enables full table characterization of constraint arcs when generating LVF data format.
`2`	Improves run time by applying an alternate algorithm for table characterization of constraint arcs when generating LVF data format.

This variable must be set before the `char_variation` command is run.

variation_sds_early_late_mode

`<-1 | 0 | 1 | 2>` Selects the constraint and delay arc sampling for early, late, or both in Sensitivity-Driven Sampling (SDS) mode. See also variation_sds_mode.
Default: `-1`

Set this variable to identify where to apply the SDS algorithm.

`-1` Select the default behavior for constraint and delay arcs as following:

- For constraint arcs, the default behavior is to process only the late side of the OCV data. This is same as setting `variation_sds_early_late_mode` to `1`.
- For delay arcs, the default behavior is to process both early and late side of the OCV data. This is same as setting `variation_sds_early_late_mode` to `0`.

`0` Simulate samples in both early and late sensitivity characterization.

`1` Simulate samples only in late sensitivity characterization.

`2` Simulate samples only in early sensitivity characterization.

This variable must be set before the `char_variation` command is run.

variation_sds_mode

`<0 | 1 | 2>` Enables Sensitivity-Driven Sampling (SDS) mode that enhances the Variety sensitivity flow to increase accuracy by applying real Monte Carlo simulation.
Default: `0`

Note: Using SDS mode can increase the run time over using only Variety sensitivity characterization.

`0` Do not use SDS mode.

`1`	Simulate a fixed number of samples (see variation_sds_sims) for early or late sensitivity.
`2`	Simulate a dynamic number of samples (see `variation_sds_sims`) for early or late sensitivity and converge by confidence.

This variable must be set before the `char_variation` command is run.

variation_sds_samples

<value>	Specifies the number of random samples used in SDS mode (see variation_sds_mode). The default behavior is to calculate this value automatically using variation_sigma. A value larger than `10000` is recommended.

This variable must be set before the `char_variation` command is run.

variation_sds_sims

<value>	Specifies the maximum number of simulations for early or late sensitivity. Default: `500`

This variable must be set before the `char_variation` command is run.

variation_sigma

<value>	Specifies the sigma value at which the variation variables are being characterized.

Use this variable to specify the sigma value at which the sensitivity variation variables (see the define_variation command) are characterized. All the variables must be characterized at the sigma value specified by this variable. It is up to the user to verify that this happens. For example, if all the parameters defined using `define_variation` are at `3` sigma, the `variation_sigma` variable must be set to `3.0` to enable the correct detection of the mean value.

This variable must be set before the `char_variation` command is run.

variation_static_partition_info

`<0 | 1>` Enables output of partition flow-related statistics.
Default: `0`

`0`	Disables writing out of partition-related statistics.
`1`	Enables writing out of partition-related statistics.

This variable must be set before the `char_variation` command is run.

variation_static_partition_mode

`<0 | 1 | 2 | 3>` Enables Logic Cone Analysis (LCA).
Default: `2`

`0`	Disables partitioning and uses the entire circuit.
`1`	Applies partitioning for statistical analysis.
`2`	Applies partitioning to predict statistical analysis. This setting reduces the simulation run time because the analysis is performed on a partitioned subcircuit.
`3`	Uses the partitioning information to predict statistical analysis for a whole subcircuit. Therefore, the results with this setting are more accurate, but the simulation run-time speed is slower than `2`.

Use this parameter to enable acquisition of sensitivities in a manner that speeds up the simulation run time by ignoring all transistors outside the cone of logic that ends at the pin for the arc being characterized.

This variable must be set before the `char_variation` command is run.

variation_static_partition_state_incr

`<0 | 1 | 2>` Specifies the levels of additional circuit structures to include in the LCA partition.
Default: `0` (Include standard partition)

`0`	Includes standard partition.
`1`	Includes everything from setting `0` plus latches.
`2`	Includes everything from setting `1` plus flip-flops.

Note: Setting variation_static_partition_state_incr to `1` and variation_static_partition_state_max to `2` is always recommended.

To accurately characterize a given arc, LCA (enabled with variation_static_partition_mode=`2`) improves simulation performance by including a specific portion of the subcircuit instead of the entire circuit. Normally, a single latch or flip-flop together with relevant surrounding combinatorial logic is sufficient. In certain special cases, the surrounding logic must also include more than combinatorial logic. Each level of expansion will increase simulation time.

This variable must be set before the `char_variation` command is run.

variation_static_partition_state_max

<value> Specifies an upper limit on the number of sequential elements, such as, latches and flip-flops, along a circuit path during LCA partition.
Default: `100` (This means unlimited.)

Use this variable to specify the maximum number of sequential elements the LCA algorithm (see variation_static_partition_mode) will allow in a path. The larger the number, the longer is the run time. The recommended setting is `2`.

This variable must be set before the `char_variation` command is run.

variation_target_sigma

<value> Specifies the target sigma for the characterization.
Default: `3.0`

Use this variable to set the target sigma value for the sensitivity-driven sampling flow (see variation_sds_mode) or the cross-term flow (see variation_early_late_adjust_mode).

For example, if the variable is set to `3.0`, the `99.865%` percentile point of the sorted samples will be selected for the calculation of the `late` part, and the `(1 - 99.865%)` percentile point of the sorted samples will be selected for the calculation of the `early` part.

This variable must be set before the `char_variation` command is run.

variation_voltage_adjust_pin_supply

`<0 | 1>` Specifies whether the cell pins should be varied along with the voltage supply or only the voltage supply should vary. Use this variable when you choose to characterize voltage variation sensitivity.

- `0` While characterizing the power supply for voltage variation sensitivity, the cell input voltages, related_pin transition, and measurement thresholds are not adjusted and are based on the original unmodified supply value instead.
- `1` While characterizing the power supply for voltage variation sensitivity, all cell pins are adjusted to match the adjusted voltage.

This variable must be set before the `char_variation` command is run.

variation_voltage_variation_use_percent

`<0 | 1>` Specifies the unit of the supply voltage variation in the `define_variation` command.

- `0` The value in the `define_variation` command for the supply voltage variation is in volts.

`1`	The value in the `define_variation` command for the supply voltage variation is in percent, specified as a ratio of the supply voltage. To specify a 10% variation, the value in the `define_variation` command will be `"0.10"`.

This variable must be set before characterization, `char_variation` and `create_aocv_table` commands are run.

variety_netlist_mode

`<0 | 1>` Sets the method that Variety uses to interpret the netlist when characterizing variations and using `define_leafcell`.
Default: `1`

`0`	Enables the method used in Liberate 12.1 ISR3 and prior releases.
`1`	Default setting.

This variable must be set before the `char_variation` command is run.

variety_pin_cap_match_liberate

`<0 | 1>` Controls whether Variety should use tighter accuracy checks when CCS characterization is requested with the `char_variation -ccs` option.
Default and recommended: `0`

`0`	Allows higher accuracy so that Variety can achieve better correlation for statistical input pin capacitance. This might result in tiny differences in input capacitance when compared to Liberate results.
`1`	Uses the same accuracy checks as Liberate, yielding identical input capacitance values at the expense of some accuracy.

write_library_mode

`<0 | 1>` Enables a faster method for writing a variation sensitivity format library in the arc packet flow.
Default: `0`

- `0` Writes a library on the server containing all characterized cells.
- `1` Distributes the write_variation command to each client in an arc-packet flow. See packet_mode.

When enabled, a directory is created in the LDB called `LIBS`. This directory contains a unique library file for each characterized cell. These individual libraries are appended together on the server. This distributes the creation of variation model across the clients.

For this flow, the `write_variation` command should be run both on the server and the client. Any existing script that restricts the `write_variation` command from running on the client (see `packet_slave_cells` and `ALAPI_active_cell`) must be updated. If the `write_variation` command is not run on the client, the variation model is written on the server like it is done when `write_library_mode` is set to `0`.

This variable must be set before the write_variation command is run. Writing a library with a setting of `1` followed by writing a library with a setting of `0` is not supported.

5

Parallel processing

This chapter describes how to use Variety across multiple CPUs.

To achieve good performance, Variety can use multi-threading across all available CPUs. Furthermore, Variety can use distributed processing with multi-threading across a network of machines. Parallel processing reduces the total turnaround time for characterization nearly linearly with the number of CPUs used.

Multi-threading

The simplest way to use parallel processing with Variety is to use multiple threads on a single computer. In multi-threading, multiple CPU cores residing on the same physical machine operate on the same memory image.

Variety automatically determines the optimal number of threads based on the hardware characteristics of the available CPUs. The `-thread` option of the `char_variation` command can be used to increase or decrease the number of parallel processes that Variety can use on a single machine.

Distributed Processing

To distribute Variety across multiple CPUs, `set_client` commands are used. The `set_client` commands specify either the names, or number of client machines to be used. For each machine, a directory in which Variety can temporarily store data must also be specified. If the `set_client -n` option is used, Variety submits the appropriate number of tasks to the named queuing system.

```
# Specify 10 client machines to use on lsf_queue
# Use /tmp/Variety_#N to store intermediate files
set_client -dir /tmp/Variety_%N -n 10 lsf_queue
```

In parallel processing mode, Variety partitions the characterization task into a group of related simulations (arc partitions) to be performed on each of the available CPUs. Consequently, the

characterization workload is well balanced across all of the machines and the improvement in turnaround time is nearly linear with the number of machines.

The `rsh_cmd` variable (default `ssh`) can be used to specify the shell to use for starting remote jobs on a client machine. The `rcp_cmd` variable (default `scp`) can be used to set the command for copying files from the host to client machines. Before starting a parallel processing job, make sure that the following commands can be performed without requiring any password or passphrase.

- `ssh` or `rsh` from the server to the client
- `scp` or `rcp` a file from the server to the client

The `rsh_cmd` command string can reference the current client and the command to invoke Variety by using `%M` (machine or queue name given to the `set_client` command) and `%C` (command). In addition, command-line options that appear after the Variety Tcl filename can be passed into the `rsh_cmd` string by using `%O` (options). These % overrides can be useful if your system is using load-balancing and scheduling software.

If using LSF to run remote jobs use:

```
# Use LSF bsub to invoke jobs on remote clients
# %M is replaced with the queue name "lsf_queue"
set_client -dir /tmp/Variety_%N -n 10 lsf_queue
set_var rsh_cmd "bsub -q %M %C"
```

If using Sun Grid to run remote jobs use:

```
# Use SunGrid qsub sub to invoke jobs on remote clients
# %M is replaced with the queue name "sungrid_queue"
set_client -dir /tmp/Variety_%N -n 10 sungrid_queue
set_var rsh_cmd "qsub -b y -q %M %C"
```

When using distributed mode it is important to make sure that each client machine can access the necessary external SPICE binaries and licenses. To ensure this, create an `altos_init` shell script (sh or bash) in your home directory that sets the path to the binaries and licenses. Each time Variety starts on a client machine, this script is sourced. Example script:

```
export PATH=/home/spice_vendor/bin:$PATH
export LM_LICENSE_FILE=2860@linux1:$LM_LICENSE_FILE
```

Variety requires that all the server and host machines are NFS-mounted and that all the files and directories use the same path. In addition, all the files referenced in the Variety Tcl file must use absolute path names. If using the `-n` option to `set_client`, then the Variety Tcl file should also be a full path name. References inside SPICE netlist files using `.include` or `.lib` commands are assumed to be relative to the top level SPICE netlist. For example, if a model file `model.sp` is in a `models` directory at the same level as the top level SPICE netlist, then use `.include ./models/model.sp` in the top level SPICE netlist.

Variety automatically recovers from any client failures that are caused by system failures, license failures, or clients dying. If a client is killed, the tasks for that client are re-assigned to another available client. If a client fails to communicate back to the server within a pre-defined heartbeat time (`heartbeat_timeout`) Variety removes that client from the list of available clients and re-assigns that client's tasks to another client. If the re-assigned tasks also result in a client failure, the tasks are skipped and the current cell is omitted from the library database (LDB).

If it is using a queuing system that permits pre-emption (stopping and re-scheduling of active jobs), Variety tries to re-schedule the stopped client's current tasks to another free client. If no clients are free, these tasks are put back on the list of characterization tasks to be performed. After the pre-empted client re-starts it is given a new collection of tasks to be performed. The characterization process continues until all of the tasks have been performed.

A

Deprecated and Legacy Variables

This appendix lists all the variables that are either deprecated, or included for backward compatibility. We would like to discourage users from relying on these variables, and instead find alternate variables and settings to achieve the best results from Cadence® Virtuoso® Variety tool.

Deprecated Variables

The variables covered in this section are being phased out, and have been replaced by either new commands, new variables, or new behaviors of the tool.

bundle_count

<*number*>	Specifies the number of packets to be used while validating a library. Default: `0` (Use `1` packet per cell.) If this variable is not set, Variety uses `1` packet per cell. If <*number*> is set to a non-zero number, Variety divides the number of cells in the library into that number of packets. (See packet_mode for more information.)

This variable must be set before the `char_variation` command is run.

conditional_variation

`<0 | 1 | 2>`	Controls whether detailed parameter random variation characterization is performed for each distinct `when` condition. Default: `1` (on)

`0`	Characterizes only the worst condition per arc for random sensitivity. The data from the worst case is then duplicated for all other conditions. This can speed up characterization time for cells with many side-input conditions such as AOI gates.
`1`	Calculates a unique set of sensitivity values for each state for every random parameter.
`2`	Characterizes the worst condition for both random and systematic variations. Then populate the characterized sensitivity to each condition.

Examples

```
# Disable conditional random variation characterization
set_var conditional_variation 0
```

default_timing

Important

You are recommended to use the set_default_group command in place of this variable.

`<off | min | max>` Controls the creation of default library group data for pin capacitance and timing.

`off`	Omits the default timing group.
`min`	Uses the minimum delay, transition, and constraint value as the value for the default timing group.
`max`	Uses the maximum delay, transition, and constraint value as the value for the default timing group.

Examples

```
# Set defaults for the fast corner
set_var default_timing min
set_var default_capacitance min
```

```
# Ignore default timing groups
set_var default_timing off
```

max_capacitance_auto_mode

`<0 | 1>` Selects the computation mode for TieHi and TieLo cells.
Default: `1`

`0` Uses a default algorithm for determining the maximum capacitance.

`1` The `char_variation` and `write_vdb` arguments `-auto_index` or `-auto_max_capacitance` calculate the *max_capacitance* pin attribute for TieHi and TieLo cells by dividing the max_transition by the resistance drain to source (Rds).

To measure the Rds, connect a 100k Ohm resistor to the opposite supply and compute the Rds as follows:

TieHi_Ion = (Vdd -Vth)/Rds = Vth/Rl
TieLo_Ion = (Vdd -Vtl)/Rl = Vtl/Rds
Max_cap = max_transition / Rds

Where:

`Vth` = TieHi steady-state output pin voltage,
`Vtl` = TieLo steady-state output pin voltage.

This variable must be set before the `char_variation` command is run.

mpw_delay_use_active_edge

`<0 | 1>` Controls whether MPW related delay degradation is measured from the active edge or the leading edge of the circuit clock.
Default and recommended: `1`

- `0` When `mpw_criteria = 1` (delay degradation), Variety measures the delay degradation from the leading edge of the circuit clock to the transition on the probe node. This can lead to incorrect delay pushout if the delay should be measured from the trailing (active) edge of the pulse.
- `1` Variety measures the delay degradation from the trailing (active) edge of the circuit clock.

This variable must be set before the `char_variation` command is run.

variation_search_mode

`<0 | 1>` Enables advanced heuristics that can improve the characterization run time.
Default and recommended: `1`

- `0` Uses standard algorithms.
- `1` Enables advanced heuristics that can reduce characterization run time.

variation_sign_mode

`<0 | 1>` Controls how random variation contributions are combined into the early and late sensitivity values.
Default: `1`

- `0` The direction of the shift is not monitored.

  Late = "pos" = delay shift is measured at `+value` from `define_variation`.

  Early = "neg" = delay shift is measured at `-value` from `define_variation`.

`1`	The `pos` (`neg`) is mapped to the `late` (`early`) as determined by the actual delay shift from nominal. This improves the correlation accuracy versus Monte Carlo.

Note: Setting this variable to `1` is effective only if the non_linear_random_variation variable is set to `1` or `3`.

This variable must be set before the `char_variation` command is run.

Backward Compatibility Variables

The variables covered in this section invoke older behaviors of the tool. We generally discourage using these variables because many of the older algorithms have since been corrected or improved.

lvf_enable_constraint

`<0 | 1>` Controls output of On-Chip Variation (OCV) constraint variation data to the Liberty Variation Format (LVF) in the Variety sensitivity file. When you specify the Liberate `write_library -sensitivity_file` command, the data included in the Variety sensitivity file will be written out in a Liberate-generated library in LVF format. Default: `1`

`0`	Disables the output of constraint variation data in the sensitivity file. This is provided for backward compatibility with Liberate13.1 ISR3 and earlier releases.
`1`	Enables dumping of OCV constraint variation data to the Variety sensitivity file. For more information, see the `write_variation -format "sensitivity"` command option.

This variable must be set before the `write_variation` and `write_library` commands are run.

Example

Run the following steps:

1. Run Variety:

```
char_variation -cells $cells
write_variation -format "sensitivity" -filename a.lib ff_2p6v_0c
```

2. Run Liberate:

```
write_library -sensitivity_file a.lib -filename b.lib ff_2p6v_0c
```

lvf_enable_transition

`<0 | 1>` — Controls output of On-Chip Variation (OCV) transition data to the Liberty Variation Format (LVF) in a Variety sensitivity file. When you specify the Liberate `write_library -sensitivity_file` command, the data included in the Variety sensitivity file will be written out in a Liberate-generated library in LVF format. Default: `1`

- `0` — Disables the output of transition data in the sensitivity file. This is provided for backward compatibility with Liberate13.1 ISR3 and earlier releases.
- `1` — Enables dumping OCV transition data into the Variety sensitivity file. For more information, see the `write_variation -format "sensitivity"` command option.

This variable must be set before the `write_variation` command is run.

Example

Run the following steps:

1. Run Variety:

```
char_variation -cells $cells
write_variation -format "sensitivity" -filename a.lib ff_1p5v_0c
```

2. Run Liberate:

```
write_library -sensitivity_file a.lib -filename b.lib ff_1p5v_0c
```